Improving Information Retrieval with Textual Analysis: Bayesian Models and Beyond

by

Jaime B. Teevan

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

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2001

© Jaime B. Teevan, MMI. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author ....
Department of Electrical Engineering and Computer Science
May 25, 2001

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

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

Improving Information Retrieval with Textual Analysis: Bayesian Models and Beyond

by

Jaime B. Teevan

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2001, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Information retrieval (IR) is a difficult problem. While many have attempted to model text documents and improve search results by doing so, the most successful text retrieval to date has been developed in an ad-hoc manner. One possible reason for this is that in developing these models very little focus has been placed on the actual properties of text. In this thesis, we discuss a principled Bayesian approach we take to information retrieval, which we base on the standard IR probabilistic model. Not surprisingly, we find this approach to be less successful than traditional ad-hoc retrieval. Using data analysis to highlight the discrepancies between our model and the actual properties of text documents, we hope to arrive at a better model for our corpus, and thus a better information retrieval strategy. Specifically, we believe we will find it is inaccurate to assume that whether a term occurs in a document is independent of whether it has already occurred, and we will suggest a way to improve upon this without adding complexity to the solution.

Thesis Supervisor: David Karger
Title: Professor of Computer Science

Acknowledgments

I would like to thank David Karger for his invaluable guidance throughout this process. I have particularly appreciated his theoretical background and enthusiasm for understanding the fundamentals of any problem. He has taught me to hack less and think more. In addition to David, Lynn Stein and everyone else in the Haystack group, have made the past two years stimulating and enjoyable. Tommi Jaakkola and Leslie Kaelbling were valuable resources during this research. I would also like to acknowledge the National Science Foundation for their financial backing. And, of course, thank you, Alexander Hehmeyer, for your love and support. You are a part of everything I do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis we will talk about the work we have done using textual analysis to improve information retrieval (IR). The purpose of this section is to motivate our work by first talking about why information retrieval is important and why the approach we have taken to improving IR is a good one. We will then discuss several reasons that textual models should be used for information retrieval, and motivate using textual analysis in the process of building a model. Finally, we will give a quick overview of exactly what we are going to discuss in this thesis.

1.1 Why is Information Retrieval Important?

Text retrieval is a difficult problem that has become both more difficult and more important in recent years. This is because of the increased amount of electronic information available and the greater demand for text search as a result of the World Wide Web. People are surrounded with large quantities of information, but unable to use that information effectively because of its overabundance. For example, the Web without a search engine would be a much less useful tool. By improving information retrieval, we are improving people's access to the information available to them.

1.2 How is Information Retrieval Done?

Since we will be discussing information retrieval, it is necessary first to understand something about how the retrieval process works. As pointed out by Robertson and Walker [36], there are two main approaches that we can take to build a system that retrieves documents for a given query. One is in an ad-hoc manner, and the other is a more principled, model-based manner. This is, of course, an over-generalization, but we will attempt to elucidate IR by discussing these two approaches separately. Let us first discuss the ad-hoc retrieval methods, as these were the first to be developed and are the most common retrieval methods used in practice. We will then talk about why one would use a model for information retrieval instead. This motivation is important since we will use a model in our investigations.

Keep in mind that while we discuss the two approaches separately, most approaches to IR fall somewhere between the two extremes. Even the most highly principled models have some elements that are chosen in an ad-hoc fashion because they "seem right". While retrieval may initially be developed in an ad-hoc manner, these systems can often be motivated by models as well [6]. This allows us to reap the benefits of a model from the ad-hoc methods.

1.2.1 Ad-hoc retrieval

What does it mean to retrieve documents in an ad-hoc fashion? It means that we do not approach the problem of IR in a principled manner, but rather continue to add new heuristics to whatever retrieval method we employ in hopes of improving our result quality. For example, let us say that we have a query and a set of documents. We could retrieve documents by returning those documents that contain the query terms. To improve upon this in an ad-hoc manner, we could decide to factor the number of times that a term appears in a document as well. Probably most of the searching that you do is done by an ad-hoc retrieval method, as most Internet search engines use these methods. The best example of such a system is Salton's early vector space model [40]. Note that while it may be difficult to understand what is happening with an ad-hoc retrieval system, they can be based on textual analysis and a good understanding of the written language. Examples of this include work done by Sparck Jones [19] when she suggests using the inverse document frequency for term weighting, as well as more recent work by Warren Greiff [12].

There are many benefits of ad-hoc information retrieval, and these benefits are reflected in the fact that the most popular retrieval methods are all ad-hoc. An ad-hoc retrieval system is quick to build. You come up with a new way you think can improve retrieval, and you add it. Many of the ad-hoc retrieval methods are also very fast, needing only to look at the occurrence of query terms in the documents they appear at query time. And, despite the fact that you might question the performance of your favorite Internet search engine sometimes, the search result quality is actually quite high as well.

However, there are also many draw backs of such a retrieval system. Every change that is made to an ad-hoc retrieval system effects the retrieval in unpredictable ways. Additionally, it is very difficult to understand exactly what is happening in these systems, and therefore understand what should be done to improve them. For this reason people have built models for information retrieval.

1.2.2 Retrieval using a model

An architect who is designing a house cannot test her different designs in the real world. The real world is just too complex and expensive for her to operate in. Instead she builds a model of the house she is creating. This model represents the important aspects of reality, and allows her to understand her design and eventually build the house. Like the architect, we will build a model. Rather than modeling physical objects, though, we will build a model of information, modeling our corpus, query, relevant and irrelevant documents. This will allow us to work with something we can understand and control. Our model, if it is built correctly, will capture the important aspects of a query and the documents needed for retrieval.

There are many benefits to working with a model. First, the assumptions inherent in the model are explicit. For the architect, it means that it is clear that the cardboard walls of her model represent wood and that the scale of her model is 1:50. For us as IR researchers, the assumptions will relate to the properties of the text. For example, we might assume that word order is not important, or that whether or not a term occurs in a document is independent of all the other terms in that document. Many of these same assumptions are implicit in the ad-hoc retrieval methods. However, since here these assumptions are explicit, we can better understand how closely our retrieval method matches reality, and clearly understand areas for improvement.

In addition, it is nice to work with a model because the trade-offs of making changes are clear. When the architect builds a wing on her model house, she can clearly see that she is blocking windows, and see how much new floor space she is adding. When, in IR, we change an assumption, such as deciding that whether a term occurs in a document is actually dependent on what other terms are in the document, we can clearly understand what sort of complexity that will add to our model, and predict the effects of the changes on performance.

Why, then, are models not used in practice? It is because currently there don't exist models that give as high quality search results as the standard ad-hoc methods at the fast speeds that are required. But through the process of working with a model, we should be able to develop a better system for all of the reasons described above that models are good.

1.3 Using Textual Analysis in Model Building

We will be working to improve model based information retrieval in a principled manner. We will discuss exactly how models are built in greater detail in Chapter 2. Here let us discuss briefly the process by which models are often build in IR. To build a model, the IR researcher first makes some simplifying assumptions about textual information. Then, using those assumptions, he builds a model. Once the model is built, he uses it to retrieve and rank the relevant documents.

All too often it is only at this last stage, the retrieval point, that an IR researcher interacts with textual data. Using the results from his model, he runs some tests that allow him to compare his model with other retrieval methods, and determines how good the results his model produces are. He then goes back to the simplifying assumptions his model makes and modifies his assumptions. From here he develops a new model, uses it for retrieval, tests is, and compares it. We will discuss the lack of interaction an IR researcher has with the text further in Section 2.3.

Instead, what we will do in order to improve upon model based retrieval, is to go directly to the text at the point where we are making the assumptions. This way we can know exactly what is happening at the assumption level, and by grounding the assumptions in reality, build a model with a strong foundation. We all have so much experience with language and text documents, given that we have been speaking and reading almost our whole lives, that we assume that we know what's going on in written documents. While we may have a good idea of whether an assumption is true or not, that does not mean that we know which assumptions are the least true, and therefore the most worth investigating. We want to break away from using our preconceived beliefs and understand what is actually present in the data in order to improve retrieval.

1.4 Thesis Overview

In this thesis, we will first explain in greater detail several different methods of information retrieval, as well as give some insight into what sort of data analysis has been done so far, in information retrieval and in related fields. We will then discuss the basic model that we use as our baseline, the multinomial model. The multinomial model is a probabilistic model for document generation that assumes that each term in a document is selected independently from a single probability distribution. We look at how this model performs, as well as what sort of corpus statistics we would expect from this model. We then go to the text, and look at how closely the text matches what we would expect from this model.

Once we have a better understanding of the text, we update our assumptions, and build a new model. One of the things that we find through our textual analysis is that term appearance tends to be more clustered than we would expect with the multinomial model. That is, once a term is used in a document, it is more likely to appear again. We change our model to expect terms to appear in groups, and find that we can get better performance by doing this.

Chapter 2

Background

Thinking about the problem of information retrieval, it may seem amazing that a computer can know about and understand an extremely large set of documents written in natural language, at least enough to be able to answer a person's queries over that set. There are all sorts of questions that arise: When a person issues a query, does the IR system have to fetch and read through each of the documents it knows about to find results? Does it have to have an understanding of what the query is asking and what the documents mean?

Since in this thesis we will discuss how the work we have done has improved upon current information retrieval techniques, in this section we will first discuss how these questions are currently answered and develop a kit of tools that we can use to answer other similar questions in IR. We will look at some of the most common methods of information retrieval by following a single query, a query for "Granny Smith apples". There are a number of good resources that summarize information retrieval models [9, 43, 7] if you are interested in delving further into the subject.

The set of documents, also called a corpus, that we start out with is very simple:

1. Apples are the most widely cultivated of all tree fruit.
2. An apple a day keeps the doctor away. Eat lots of apples.
3. Granny Smith apples are green and Washington apples are red.

With this simple corpus, it seems reasonable that our search for "Granny Smith apples" should return Document 3 as the most relevant document. After all, Document 3 is the only document that contains any content about Granny Smith apples. To understand how we can conduct a search that ranks this document first, let's discuss some basic information retrieval models. First we will talk about the basic assumptions about the corpus that we apply to all of the models that we will discuss. Then we will describe these models, showing how they perform for this query, and finally, we will discuss how these models relate to actual textual data.

2.1 Basic IR Assumptions

First, let us look a little bit at the assumptions common to all of the information retrieval models we will be discussing here. Any information retrieval model assumes that there exist some features that belong to each document in a document set, and that these features are sufficient to determine if a document is relevant or irrelevant. While the features of a document may include its length, title, author, or any other additional information, we will look specifically at the subset of document features that we will call terms. We hope this smaller feature set will be sufficient to determine relevance.

Since we are dealing with text, a term can be defined as a word, a semantic meaning, a pairs of words, a phrase, or even an entire sentence. In our examples, and in the testing we do, we assume that a term is word, but it is important to remember that this is just an additional assumption we make. We make this assumption because we want our terms to appear often enough throughout the corpus to tell us something meaningful about their distribution. In doing retrieval, we will be drawing generalizations about documents from these terms. Without enough data to draw good generalizations, we will over fit our model to the data. Since even the occurrence of words throughout a corpus tends to be sparse, often with a typical term occurring in just 0.001% of the documents, just imagine how sparse the occurrence of something more complex like sentences is.

In information retrieval, we hope these terms are sufficient to determine if a document is relevant or not. Only the presence or absence of a term in a document matters, and not the sentences, word order, or any other structure of the document. While not all IR systems make these assumptions, each system we discuss will.

With the assumption that everything in a document is unimportant except the presence or absence of a term, and with the assumption that each term is a word, we can view the documents in our corpus as what is called in information retrieval a "bag of words". Our "Granny Smith" corpus as a bag of words looks like this:

1. all Apples are cultivated fruit most of the tree widely
2. a An apple apples away day doctor Eat keeps lots of the
3. and apples apples are are Granny green red Smith Washington

This processing clearly loses some information about the content of each documents. For example, the document Granny Smith apples are red and Washington apples are green. would look just like Document 3 (Granny Smith apples are green and Washington apples are red.), even though the documents are different. Things get even more confusing if we consider a document such My grandpa is a smith and my granny fixes Apple computers. Nothing that treats a document as a simple bag of words will have any hope of distinguishing the "Apple" in this document from the fruit "apple" in the other documents. Something like this would require a more complicated model, such as the semantic networks discussed by Fuhr [9], or a natural language system such as START [21].

As a side note, you should notice that while the proposed document Granny Smith apples are red and Washington apples are green. may be as informative about Granny Smith apples as Document 3 is, it is, in fact, wrong. That it is incorrect highlights yet another problem with information retrieval. There is a lot of information available that just isn't correct, and it would take a very sophisticated information retrieval system to understand the difference.

Given that we are working with bag of word documents, however, there are a number of trick that we could use to make the collection more useful. One is the removal of stop words. Stop words are common words whose presence provides no new information about a document. Because they appear so often in every document when they appear in a particular document it is not meaningful. Stop words include words like "the", "and", and "a". While removing stop words is a generally useful practice, it also runs the risk of removing possibly meaningful words. For example, try searching for "The Who" on Google. You get no results, because they have removed "the" and "who" as stop words. Without stop words, our corpus looks like this:

1. all Apples cultivated fruit most tree widely
2. apple apples away day doctor Eat keeps lots
3. apples apples Granny green red Smith Washington

Another way to make the terms in a document more useful is to try to merge terms together that are similar in meaning, but contain slight differences that cause them to be treated as separate terms. For example, the term "Apples" in Document 1 seems like it should match the "apple" in Document 2. We can arrive at this matching through two steps. First, we remove the case from the terms. Second, we save only the stem of the word. This means removing endings such as "-ing", and "-ed", and "-s", and sometimes more complicated changes, like mapping "ate" to "eat". We could even merge terms in a more complex manner, mapping words with the same semantic meaning. For example, the term "cultivate" and the term "farm" could be merged. Porter provides a common algorithm used for stemming [33].

Again, as in the case of stop words, stemming is a simplification that, while it often helps, can also be a problem. It could cause us to confuse two terms with different meanings. You wouldn't want your search for "number" to retrieve documents on "numbness" because the two words stem to the same thing. Church discusses some of these trade-offs in his paper [5]. With stemming and case removal, we now have a corpus that looks like this:

1. all apple cultivate fruit most tree wide
2. apple apple away day doctor eat keep lot
3. apple apple granny green red smith washington

Note that when performing retrieval, we will perform the same preprocessing to our query "Granny Smith apples". As a case-less, stop-word-less, stemmed bag of words, the query becomes "apple granny smith". Now that we understand our corpus and our query, processed in a manner typical of all retrieval systems, we will look at what different models for retrieval do with them.

2.2 Information Retrieval Models

Here we will discuss the three most common types of information retrieval, Boolean retrieval, vector space retrieval, and retrieval using probabilistic models. The viewpoint of each model is fairly obvious from the model's name. A Boolean model views each document in the corpus as a Boolean statement, so that the problem of resolving a query becomes a problem of finding which documents make the query true. A vector space model views documents and queries as vectors in Euclidean space, reducing search to finding the closest documents to a query. And probabilistic models model each document as having been created by sampling from a probability distribution. Given that we can describe the probability of certain features occurring in a document, documents can be retrieved by looking at the probability of relevance, the probability a document generated the query, or by using some other measure.

We will describe these models in some detail, continuing with our "Granny Smith apples" example. Keep in mind that each "model" we discuss here is actually a class of models that encompasses much variation. We will also briefly discuss some of the real world implications of using of these different models. Fuhr [9] discusses each of these models in greater detail.

2.2.1 Boolean retrieval

As we mentioned earlier, Boolean information retrieval takes a Boolean view of the data. A document is seen as a logical statement. The components of a document are the presence and absence of a term. Each term in the corpus is represented by a boolean variable. Let us call the ith term in the corpus t_i. A document looks like this: $t_1 \wedge t_2 \wedge \cdots \wedge t_n$. Note that a closed world is assumed. This means that terms not occurring within a document are assumed to be negated. So if term 2 does not appear in a document, but all other terms do, it would actually look like this: $t_1 \wedge \neg t_2 \wedge \cdots \wedge t_n$. From a Boolean perspective, our corpus looks like this:

1. $\mathsf{all} \wedge \mathsf{apple} \wedge\neg \mathsf{away} \wedge \mathsf{cultivate} \wedge\neg \mathsf{day} \wedge\neg \mathsf{doctor} \wedge\neg \mathsf{eat} \wedge \mathsf{fruit} \wedge\neg \mathsf{granny} \wedge\neg \mathsf{green} \wedge\neg \mathsf{keep} \wedge\neg \mathsf{lot} \wedge \mathsf{most} \wedge\neg \mathsf{red} \wedge\neg \mathsf{smith} \wedge \mathsf{tree} \wedge\neg \mathsf{washington} \wedge \mathsf{wide}$
2. $\neg \mathsf{all} \wedge \mathsf{apple} \wedge \mathsf{away} \wedge\neg \mathsf{cultivate} \wedge \mathsf{day} \wedge \mathsf{doctor} \wedge \mathsf{eat} \wedge\neg \mathsf{fruit} \wedge\neg \mathsf{granny} \wedge\neg \mathsf{green} \wedge \mathsf{keep} \wedge \mathsf{lot} \wedge\neg \mathsf{most} \wedge\neg \mathsf{red} \wedge\neg \mathsf{smith} \wedge\neg \mathsf{tree} \wedge\neg \mathsf{washington} \wedge\neg \mathsf{wide}$
3. $\neg \mathsf{all} \wedge \mathsf{apple} \wedge\neg \mathsf{away} \wedge\neg \mathsf{cultivate} \wedge\neg \mathsf{day} \wedge\neg \mathsf{doctor} \wedge\neg \mathsf{eat} \wedge\neg \mathsf{fruit} \wedge \mathsf{granny} \wedge \mathsf{green} \wedge\neg \mathsf{keep} \wedge\neg \mathsf{lot} \wedge\neg \mathsf{most} \wedge \mathsf{red} \wedge \mathsf{smith} \wedge\neg \mathsf{tree} \wedge \mathsf{washington} \wedge\neg \mathsf{wide}$

Be aware that this is a slight abuse of notation. The word isn't actually a truth value like we are using it to represent. Note also that the number of occurrences of a term is not important, since $t_1 \wedge t_1 \rightarrow t_1$.

When retrieving, the query is treated as a boolean statement consisting of the terms it contains. The goal is to find the set of documents that could be true given the query. Only Document 3 would be consistent with the query ($\mathsf{apple} \wedge \mathsf{granny} \wedge \mathsf{smith}$).

You should note several things about this model. Boolean retrieval is very powerful for a user who knows how to specify exactly what they want. For example, we could issue a very complicated query: $((\mathsf{granny} \wedge \mathsf{smith} \wedge \mathsf{apple}) \vee (\mathsf{apple} \wedge \mathsf{tree})) \wedge\neg \mathsf{washington}$. In this case we would get Document 1 as a result, but not Document 3.

Formulating such complex queries can be difficult for users. We know that the standard Web user of Internet search engines typically uses very short and simple queries. She tends not to put the effort into modifying her query in order to obtain better results, and would probably never formulate complicated Boolean queries [41].

Another problem with Boolean retrieval is that it always yields a set of documents as a result, without any further ranking. A document either satisfies the query, or it doesn't. If the result set is very large, presenting the relevant documents to the user could pose a problem.

2.2.2 Vector space retrieval

The vector space model was originally introduced by Salton [39, 40], and also developed by Sparck Jones [19] and others. It is one of the most common forms of retrieval used in practice [25]. Vector space models represent a broad class of models that view text documents and queries as term vectors. Once a document is represented as a term vector, the similarity of that document to a query can be represented as a function of Euclidean distance of the document from the query. This class of models is popular because it performs well, and the results are computationally efficient.

To represent a document or a query as a vector, we create a vector that has an element for each term in the corpus. As you can surely imagine, these document and query vectors can be very large, because they have to have the dimension equal to the number of unique terms in our corpus. This number can be huge in a real corpus, and you can see that it is even fairly large (18) given our small example corpus. Each coordinate in the vector represents information corresponding to the term's occurrence in the document or query the vector represents. The entire corpus can be represented as a matrix, where each row is a document vector.

The value representing a term in a document vector can take on any value we define it to. For example, it could be binary. This would mean that it simply represents the presence or absence of the term in the document. It could also be more complex. We will discuss complex term representations later in this section. For now, let us take the simple binary case where each term is 0 or 1, depending on whether it appears in the document or not. 1 will indicate the presence of the term in the document, and 0 will represent that the term is not present. Under these conditions, our corpus and query look like this:

$$D = \begin{array}{c} \begin{array}{cccccccccccccccccc} \text{all} & \text{apple} & \text{away} & \text{cultivate} & \text{day} & \text{doctor} & \text{eat} & \text{fruit} & \text{granny} & \text{green} & \text{keep} & \text{lot} & \text{most} & \text{red} & \text{smith} & \text{tree} & \text{wash} & \text{wide} \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$q = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

To find the closest document to the query, we will chose the document that has the highest dot product with the query. If the document vectors have the same magnitude, the document with the highest dot product with the query will be the document with the smallest angle between it and the query. In a binary model, if two documents have the same number of unique terms, the document vectors for those two documents have the same magnitude. This was more or less true of the early corpora that IR systems were tested with [18]. Therefore the dot product was a good measure of closeness during the development of early vector space models. For more complex coprora such as the TREC corpus we will discuss in Section 2.3.3, the assumption that document vectors have the same magnitude no longer holds. For this reason there has been considerable work with document length normalization [4, 18].

In our example, we will not worry about vector magnitude. The dot product of our document matrix with the query results in this:

$$D \cdot q\prime = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

Note that this scheme, unlike Boolean retrieval, actually produces a ranking. Document 3 is ranked first. Document 1 and Document 2 tie for second. However, like Boolean retrieval, this version of the vector space model does not account for term frequency. This does not mean, however, that the vector space model cannot account for term frequency. It can, in fact, incorporate what ever information about the term and its presence in a document that we chose it to, merely by modifying the value at the appropriate place in the vector.

Let us say that we get a new document in our corpus: Granny Smith apples are green and Granny Smith apples are wide. The document vector for this document looks like:

$$\mathbf{d^4} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Since $\mathbf{d^4}.q\prime = 3$ this new document ties for most relevant with Document 3. But you could argue that this document is more informative than Document 3 about Granny Smith apples, evidenced in part by the greater number of times that it says "Granny Smith apples". It does, after all, provide two facts about the apples rather than just one. It would be nice if our search results reflected this. One way to do this would be, instead of having our coordinates be binary values, they could be the term frequency ($\mathrm{TF}(t_i)$) of a term within a document. Modified in this way, and taking into account term frequency, our vector space model will compute the following scores for each document:

$$\begin{array}{c} \begin{array}{ccc} \text{apple} & \text{granny} & \text{smith} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 2 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \end{array} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}\prime = \begin{bmatrix} 1 \\ 1 \\ 3 \\ 6 \end{bmatrix}$$

Because the terms that don't appear in the query have a value of 0 in the query vector, only terms that actually appear in the query factor into the dot product. For this reason, to make notation simpler, we ignored non-query terms in our document representation. We have only shown the documents' and query's values for "granny", “smith”, and “apple”.

You can see that for our example, incorporating term frequency into our model works well. But imagine we get another document that says a lot about apples, but nothing about Granny Smith apples: Apples can be used to make apple cider, apple juice, apple sauce, apple pie and apple strüdel. If we were to include this new document (represented as the bottom row of the new document matrix), we would get results that look like:

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 0 \\ 2 & 1 & 1 \\ 2 & 2 & 2 \\ 6 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}' = \begin{bmatrix} 1 \\ 1 \\ 3 \\ 6 \\ 6 \end{bmatrix}$$

The new document, Document 5, actually receives the same score that Document 4 does, and a higher score than Document 3. But we could argue that it is actually less relevant since it doesn't contain any content relating to Granny Smith apples specifically. Many common vector space models account for this problem by factoring in a term's rarity.

In our corpus, “apple” is a very common word. Because it appears in every document, it is almost like a stop word in our corpus. Its presence doesn't mean very much. Therefore it is not likely to be a very important term in our query. On the other hand, “Granny Smith” doesn't appear quite as often. For this reason, we might say that it is a more informative part of our query. Using this rationale, we would like to boot the value of informative terms. To do this, we use the inverse of the number of different documents a term appears in (inverse document frequency, or IDF(t_i)) as a measure of rarity.

The following equation shows the document matrix with the term rarity taken into account. Each entry represents the term's frequency within a document divided by the term's inverse document frequency. For example, since IDF(granny) = 2, the value for the term "granny" in Document 3, where the term occurs only once, is $\frac{1}{2}$. On the other hand, even though the term "apple" occurs twice in the same document, its coordinate takes on a lower value ($\frac{2}{5}$). This is because "apple" has an inverse document frequency of 5.

$$\begin{bmatrix} \frac{1}{5} & 0 & 0 \\ \frac{2}{5} & 0 & 0 \\ \frac{2}{5} & \frac{1}{2} & \frac{1}{2} \\ \frac{2}{5} & \frac{2}{2} & \frac{2}{2} \\ \frac{6}{5} & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}' = \begin{bmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 1\frac{2}{5} \\ 2\frac{2}{5} \\ 1\frac{1}{5} \end{bmatrix}$$

Now we get the results we wanted, with Document 4 being judged most relevant, and Document 3 the next.

As you can see, the general vector space framework allows for a lot of variation. The coordinates in the vector can be represented as many different variations on what we have discussed here, and closeness measures can be effected in many different ways as well, depending on how we normalize for document length. While the initial vector space models were developed empirically, as we have done here, it should be noted that there exist probabilistic models that in their simple forms can be reduced down to a vector space model, and can be used to motivate vector space retrieval [18].

The most common vector space model is very similar to the method described above. Because it takes into account term frequency and the inverse document frequency of a term, it is often called tf.idf. We will be using tf.idf for comparison with our retrieval systems. The specific variation we will use is mentioned by McKeown et. al. [27]. We will set the ith coordinate of the vector for document j to be:

$$\mathrm{TF}(t_i) \cdot \log\left(\text{number of documents} \cdot \mathrm{IDF}(t_i)\right)$$

Recall from before that $\mathrm{TF}(t_i)$ is simply the number of times that a term occurs in a document, and IDF(t_i) is the term's inverse document frequency.

2.2.3 Probabilistic models

A number of principled models have been developed to improve upon the ad-hoc, yet successful, vector space method described above. The probabilistic retrieval model is one of the most common frameworks for building principled information retrieval models. vanRijsbergen gives a good introduction to the basics of the probabilistic model in his book [43]. There are a number of other treatments of probabilistic models if you are interested in pursuing the topic further [10, 24]. In this section we will describe the basic probabilistic framework.

Probabilistic models assume the existence of a query, and then model the generation of documents that are relevant and irrelevant to that query. Our discussion here will all be with respect to a given query. A probabilistic model further assumes that there exists a feature distribution from which each relevant document is created and a feature distribution from which each irrelevant document is created. Note that these feature distributions may be simple, such as assigning each term a probability of occurring in a document, or complex, including the selection between many other feature distributions. Documents are assumed to have been generated by sampling from these distributions.

Given a probabilistic generative model for the corpus, a probabilistic model must retrieve and rank documents. There are several ways to do it. Here we will highlight two. When we rank documents by the probability that a document would generate the query, it is called a language model. However, traditionally, when we refer to probabilistic models, we mean we will be ranking by the probability that a document is relevant. We will discuss both methods, but will focus on the later since it is what we use in our models. In our discussion, let us use r_j to represent whether or not document j is relevant ($r_j = 1$ if document j is relevant and $r_j = 0$ if it is irrelevant), and $\mathbf{d^j}$ represent document j.

Language models

A language model is interested in ranking documents by the probability that the query could have been generated if a particular document were relevant ($\Pr(q|\mathbf{d^j})$) [1, 32]. Language modeling is not only done specifically for the purpose of information retrieval, and Rosenfeld discusses language modeling in general further [37]. Ng takes a different approach to ranking, looking at the change in likelihood of a document from before the query is seen to after [28]. This approach reduces to a ranking that is very similar to that of a language model.

Probability ranking principle

On the other hand, we will be interested by ranking by the probability that a document is relevant, $\Pr(r_j|\mathbf{d^j})$. It can be shown that if the cost of incorrect retrieval is the same across documents, ranking documents in descending probability of relevance reduces cost, and is therefore optimal [10]. This idea is so important to probabilistic retrieval models that the concept of ranking by the probability that a document is relevant is given a name, the *probability ranking principle* [35].

Given the probability ranking principle, if we know the probability for a particular query that a document is relevant ($\Pr(r_j|\mathbf{d^j})$), then we can produce results. For example, if the probability that Document 3 is relevant is 0.67, while the probability that Document 1 is relevant is 0.12, then we can rank the two documents using those probabilities, placing 3 ahead of 1.

We don't know this value. But, given a generative model and the appropriate term distributions, we can find $\Pr(\mathbf{d^j}|r_j)$. If we know the relevant and irrelevant distributions that the documents are created from, as well as the sampling method used to generate the documents, it is straight forward to find the probability that a document was generated from each distribution. Unfortunately, what we're interested in is not the probability that a document was created from the relevant or irrelevant distributions, but rather, given a document, whether that document is relevant or not. Fortunately, Bayes rule transforms this problem from something we don't know into something we do.

$$\Pr(r_j|\mathbf{d^j}) = \frac{\Pr(\mathbf{d^j}|r_j)\Pr(r_j)}{\Pr(\mathbf{d^j})} \tag{2.1}$$

Using a probabilistic model for information retrieval thus becomes a matter of understanding the generative model for a document and accurately estimating the values for the distributions from which documents are generated.

Distribution of terms

We have seen how, given a probabilistic model, understanding the distribution of terms from which documents are generated is fundamental. We need some idea of what the distribution looks like in order to find the probability that a document was generated from the distribution.

There are a number of ways that terms could be distributed. Early probabilistic models were binary, and assumed terms either appeared in a document or they didn't. On the other hand, the 2-Poisson model proposed by Harter [16] and built upon by many [18, 36], assumes that terms follow a Poisson distribution. Further, 2-Poisson model assumes that only certain terms from each document, the terms the 2-Poisson model declares to be "elite", or particularly indicative of the document's subject are important. Later in this thesis we will discuss in more detail using a multinomial distribution as the term distribution. The multinomial distribution has also been explored by Kalt [20].

Finding the distribution

Once the model is specified, and we know what sort of distribution we are looking for, it becomes essential to make good estimates about the parameters of the relevant and irrelevant term distributions. There are many different paths that have been explored here as well. The distribution parameters can be estimated directly from the available information or learned, but neither can be done if we don't have some information, such as documents labeled relevant and irrelevant, with which to begin.

Depending on the system, the estimation process from the labeled can take a classical approach, or a Bayesian approach. A classical approach does not try to incorporate any prior knowledge that the system may have about the distributions, but rather assumes that all information is directly available in the labeled data. A Bayesian approach places priors on the distributions that are then updated with whatever new information that becomes available. Keim, et al. [22] provides an example of such a Bayesian approach. Because we will be taking this approach, we discuss the use of priors in greater detail in Section 3.2.3.

Here we will talk about how information about the term distribution parameters is acquired and used. Once a probabilistic retrieval system collects what it believes to be the information necessary for estimating the relevant and irrelevant distributions, as well as any prior knowledge it might have available to it, the distributions can either be estimated directly from that information, or learned using some sort of machine learning technique. We discuss how to gather labeled data as well as how machine learning is done to try to improve the distribution estimates.

Gathering labeled data Some implementations of probabilistic models assume that there already exist sufficient labeled documents to estimate the relevant and irrelevant term distributions [20]. Others realize there is a need to collect that information in some way. One way that the system might collect labeled information is called relevance feedback. The user is first presented with a set of results from some other retrieval system (for example, an ad-hoc retrieval system), and the user is then asked to label the documents as relevant or irrelevant. These documents can then be used to estimate the term distributions [43]. Another source of labeled information is the query that is issued, as well as any previous queries the user has run [22].

Learning the distribution In addition to using relevance judgments that are hand labeled by the user or some other process, the system could use its own relevance judgments to feedback into its results in order to find what it thinks are the most likely term distributions. One way to do this feedback is called Expectation Maximization

(EM). EM is a simple iterative hill climbing method commonly used for machine learning. EM finds a local maximum in the search space it is given. In the case of estimating term distributions, EM can be used to find the maximum likelihood distributions that are most likely to have created the available information.

For example, Nigam, et al., uses expectation maximization to estimate the most likely parameters for their probabilistic text classification model and use the estimates classify text documents [30, 29]. While they still rely on having some labeled training data, they are able to use the unlabeled data as well in a feedback loop.

In this thesis, we use EM to find the most likely term distribution for our retrieval model, and will discuss the details of how we do this further in Section 3.2.6. There are many good discussions that can be found on learning [17] and on EM specifically [3, 34], including several that deal specifically with Bayesian parameter estimation [31, 8]. For this reason, we will only go into the details of it as necessary when discussing the models that we use for retrieval.

2.2.4 Practical considerations

Despite the fact that there are many models for information retrieval, some of which we have just described, the most common form of IR in practice is the fairly simple vector space method described in Section 2.2.2. It is popular because it gives good performance and it can be optimized to run very fast. As we saw with the vector space model, only the query terms matter in retrieval. This being the case, we can build what is called a reverse index, which maps terms to documents in which they occur. At retrieval time we only need to look at the documents that contain the query terms. These we can quickly retrieve from the reverse index and perform the necessary dot product.

The results for some probabilistic models can similarly be computed very quickly, but most are fairly complex. The performance of a model is a function of how many parameters are required to be estimated during retrieval, and what the estimation methods used are. One of our goals in working with the multinomial model that we will discuss in Chapter 3, as well as in our development of new probabilistic models in Chapter 5 is to keep the models simple. This allows for result computation that performs in a manner comparable to a vector space model.

2.3 Data Analysis

We have just looked at various contrived examples, and tried to improve upon each model we discussed by coming up with new examples that brake our current model. In doing this, we demonstrated how our intimate interaction with natural language gives us the feeling that we can look at and improve IR models without actually understanding how well the models fit actual, real life data.

Unfortunately, this intuitive often how models for information retrieval are built. A researcher says that she has feeling that a certain assumption must be wrong, and then works to improve upon that. The intuitive feeling that an assumption is very broken may stem from goals expressed in previous work in information retrieval or statistical natural language processing, or it may just be a plain belief that something clearly is wrong.

Examples of models that are developed this way are easy to find [33, 23, 13, 32]. For example, Ponte and Croft, in discussing their language model approach to information retrieval, motivate what they do based on feelings: "[M]any of the current of the current models make assumptions about the data that we feel are unwarranted [32]." In each of the cases pointed out above, the only real interaction that the research has with the data is through the evaluation process. In contrast, we go directly to the data and improve upon the current probabilistic models by understanding which assumptions are most inaccurate, and therefore need to be relaxed.

Understanding the corpus from which we retrieve is essential in building a model for information retrieval. In this section we discuss issues involving the corpus. First we will discuss the most common interaction an IR researcher has with the data, which is result evaluation. Not all information retrieval has been developed solely by interacting with the data at retrieval time, however. We will discuss some work that has been done involving exploratory data analysis to improve retrieval, as well as touch upon exploratory data analysis outside of the field of IR. Finally, we will introduce the TREC corpus, a collection of documents available for testing and evaluation.

2.3.1 Evaluating results

Exactly what to evaluate in an information retrieval system and how to evaluate it can be contentious issues [42]. There are many different aspects of a retrieval system, and comparing all facets can be hard. However, we will use the reasonably standard and well established measures of *precision* and *recall* to evaluate our model. Further discussion of IR evaluation methods can be found in VanRijbsbergen's book [43].

Imagine there is a large document set that comprise our corpus. Some small subset of these documents are relevant, and some other small subset of the documents that have been retrieved by the system we are testing. It is our hope that these two subsets overlap. If the retrieval is perfect, they're the same. This imagined document space can be seen in Figure 2-1. $A \cup B$ is the set of retrieved documents, while $C \cup B$ is the set of relevant documents. B represents the intersection of the two sets.

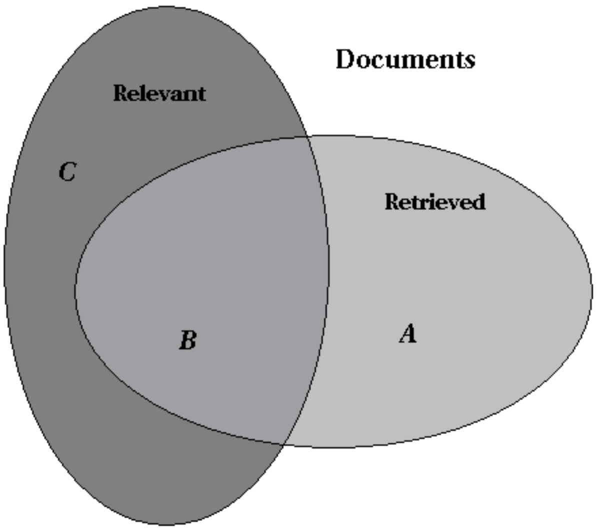


Figure 2-1: Graph of the conceptual document space. *Recall* measures the fraction of documents that are retrieved. *Precision* is the fraction of retrieved documents that are relevant.

Recall is defined as the fraction of relevant documents that are retrieved, or $\frac{|B|}{|B \cup C|}$. An example of a person who cares about high recall is an attorney doing a search for relevant cases to a case that she is handling. She wants to know about every relevant case that exists, so she wants high recall. One way to get perfect recall is to return every document in the corpus. If high recall were all that were required for good retrieval, our job would be easy.

However, we also care about *precision*. Precision is the portion of the retrieved documents that are actually relevant, or $\frac{|B|}{|A \cup B|}$. A person performing a Web search is an example of a person who cares about high precision. When he searches for "apples", rather than wanting to see all of the hundreds of thousands of Web pages on apples, he just cares that the top ten results retrieved are about apples. A good way to get perfect precision is to not return any documents at all.

As you can imagine, these two metrics are at odds with each other. You often get higher recall at the expense of precision, and vice versa. Retrieval effectiveness is measured as the ability of the system to retrieve relevant documents while holding back irrelevant ones. Because of this, it is standard in the IR community to graph both metrics in what is called a precision-recall curve. We are able to look at different precision and recall values easily when results are provided in a ranked list. Working our way down the list, we can look at the precision and recall values at for various document set sizes. When we only include documents from the top of the list, we should have high precision and low recall. As we include more documents the recall will go up while the precision goes down. Our goal when precision and recall are plotted against each other is to have the line move up and to the right, obtaining both the highest possible precision and the highest possible recall.

2.3.2 Evaluating assumptions

As we have mentioned before, we all have a lot of experience with text documents. We read the newspaper. We read books, journals and magazines. We know that when we write a document, we're not drawing words at random from some prior probability distribution. Because of this knowledge, it is often tempting when trying to improve upon a text retrieval model to make assumptions about how the data diverges from the model without actually understanding how it diverges. For example, we know that the word order in a document matters, so we might try to fix the model by removing the bag of words assumption. But this assumption may not be the most significant problem with our retrieval model. Without data analysis we cannot really know. As we mentioned earlier, this intuitive manner for improving IR models is how changes are often made. People often use the results as their closest interaction with the data.

That is not to say that there hasn't been any textual analysis. In areas other than information retrieval, there has been a lot of textual analysis done, such as in natural language processing. The results we have found are similar to what has been found in natural language processing. For example, an important topic in statistical natural language processing is Zipf's law, which says that the term distribution within a corpus obeys a power law distribution. We are interested in the term distribution for a subset of the corpus, namely for an individual document, and we find that the term distribution within a document obeys a power law distribution. Natural language processing also describes the inter-document term clustering we find, calling it "burstiness"[25]. Salva Katz applies some of what she found in here statistical natural language analysis to several problems, including information retrieval. The analysis that she performs is similar in many ways to what we describe in Chapter 4.

Some data analysis has been done with respect to information retrieval as well, and particularly with respect to ad-hoc retrieval methods, such as during the early development of vector space models [40]. More recently Warren Greiff revisits the origin of the vector space model, and uses data analysis of relevant and irrelevant document sets to understand the relationship between term frequency and the mutual information between relevance and term occurrence [12]. Through this analysis he finds a "flattening" of the typical term rarity indicator, the inverse document frequency, at document frequency extremes, and uses this to improve information retrieval. Textual analysis has also been used to look at the effect of length for document normalization [4], as well as to understand the usefulness of stemming [5]. In related areas to information retrieval, it has been used in text document summarization [11].

In our work we will similarly use data analysis in the hopes of improving retrieval, but with the goal of understanding what are the appropriate features for our Bayesian model, rather than improving the ad-hoc vector space method. In this way we are most similar to the work done by Harter in using textual analysis to develop his 2-Poisson model for retrieval [15].

2.3.3 TREC

A common set of documents for both testing and data analysis is important because documents in different corpora could conceivably be written in very different manners, and therefore have significantly different statistical properties. This is obvious when considering documents in different languages, but likely true for different topics as well. Retrieving over a collection of law review articles may be different from retrieving over Web pages.

Additionally, which documents are relevant to a person's search may be difficult to determine, especially because searches are often under specified. Although two people may both search for "Granny Smith apples", the correct result set could be very different for each person. A baker might be interested in the apple with respect to pies, while a farmer might instead be interested in Granny Smith apple trees. And if testing an individual system seems complicated, just imagine how much more complicated it gets when you want to be able to compare results across different systems.

Because of this, the information retrieval community has developed a common framework for testing. Each year at the Text REtrieval Conference (TREC), a new corpus, set of queries, and a result set for testing against, are published. Donna Harman gives a good introduction to TREC [14]. While there is still some room for variation within this framework, in general it greatly facilitates evaluation, both internally to one's experiments, and across different experiments. In addition to allowing for comparison across models, the existence of a large and common corpus for IR has fostered much of the data analysis that has begun in recent years [18]. We use the TREC framework for our data analysis and evaluation.

Chapter 3

Multinomial Model

The purpose of this thesis is to understand how to improve probabilistic models for information retrieval by using textual analysis. Because of this, let us first look more closely at a very simple probabilistic model, the multinomial model, in order to have a starting point for our textual analysis. The generative model we describe here for our version of the multinomial model is the same as the one proposed by Kalt [20], although the details of how these two models are used to perform retrieval differ.

We have already discussed the fundamentals of a probabilistic model in Section 2.2.3. Here we will focus only on the details of the multinomial model, discussing the specific implementation we use, as well as how we estimate the parameters for this model. We will conclude by discussing the results we find using the multinomial model. The differences in the statistical properties of the text from what is expected given this model will be used to help us develop a new model for information retrieval, and we will discuss that in the following chapter on textual analysis. The results we find here will be used as a baseline when testing the new models that we build based on our textual analysis.

Recall the "Granny Smith apples" example we followed throughout Chapter 2. We will use it again here to aid in our description of the details of the multinomial model. Specifically, recall that the "Granny Smith apples" corpus, separated into groups of relevant and irrelevant documents, looks like this:

Irrelevant:

1. all apple cultivate fruit most tree wide

2. apple apple away day doctor eat keep lot

5. apple apple apple apple apple apple cider juice make pie sauce strüdel use

Relevant:

3. apple apple granny green red smith washington

4. apple apple granny granny green smith smith wide

3.1 Description of Model

In this section we will elaborate on the probabilistic model framework we discussed in Section 2.2.3 to arrive at the multinomial model. In our discussion we will first talk about how the model assumes that the corpus in generated and then focus specifically on how we use these generative assumptions to rank documents for retrieval. Given this understanding of the model we will also look at the expected performance of a system.

3.1.1 Generative model

Let us describe the generative model corresponding to a multinomial probabilistic model. This model takes a very simple view of the world. While you may know that when you sit down and write a document, you are drawing on your understanding of natural language, and your knowledge of the topic about which you are writing, we're going to assert that what happens is actually much simpler. It is, after all, the purpose of a model to simplify the real world into something that we can work with.

Recall for a probabilistic model that, given a query, documents are assumed to be generated by sampling from either relevant or irrelevant distributions. Like we did during our discussion of the probabilistic model, here we again will assume that the query is a given, and talk only about the relevant and irrelevant term distributions relating to this query.

Let us look at one of the simplest ways we could generate a document from a relevant and irrelevant distribution. First, we choose a length ℓ for the document we are generating. Then we flip a coin to determine if we are going to generate a relevant or irrelevant document. If the coin comes up relevant, we independently sample terms from the relevant distribution ℓ times. If the coin comes up irrelevant, we similarly draw a term from the irrelevant distribution ℓ times. Each term in the document is found by sampling with replacement from the appropriate distribution. This means that the probability of selecting a term when sampling is always the same.

For example, if in our example corpus the term "apple" has a probability of being selected from the relevant term distribution of 0.25, then if our coin flip comes of "relevant", more or less one out of every four times we draw from the relevant term distribution, we get the term "apple". On the other hand, the probability of drawing the term "pie" when creating a relevant document may only be 0.0001. This would mean that only one out of every 10,000 times you sample from the relevant distribution do you select that term. It is no surprise, then, that we wouldn't see "pie" in a relevant document in our example corpus.

Over simplification

This model for generating a document is clearly a gross over simplification. For example, one clear over simplification is the assumption of term independence. We know that a term's appearance isn't really independent of what other terms have already been selected to appear in the document. When you create a document, the use of one word often implies the use of another word. For example, if we begin our document with the words "Granny Smith", the probability of drawing the word "apple" should be higher than "stethoscope". The advantage of assuming that the probability of generating a term is independent from what has already been drawn is that we do not need to worry about all of the interactions that terms may have. This means we don't need to make assumptions about all of those interactions, many of which we surely don't have enough data to fully understand. It also means that we don't need to account for all these interactions when retrieving, which improves retrieval efficiency. It is for this reason that such an assumption is very popular in probabilistic models.

Another objection to this over simplification that is specific to the multinomial model that we describe is that we know a term's probability of occurring in a document for a second time is probably higher than the term's probability of occurring in any random document. And yet the multinomial model says even if the the document says "apple, apple, apple", we're only as likely to draw the word "apple" again as we were when we first began the document.

3.1.2 Using the model for retrieval

Now that we have our generative model, let us understand how we will rank documents using it. Since the multinomial model is most easily described in mathematical notation, let us now quickly define the notation that we will be using. We will then use that notation to describe how we rank documents, and go through an example.

Notation

j An index into the jth document in the corpus. When we use $j = 3$, for example, we are referring to Document 3.

n The number of documents there are in our corpus. In our example case, $n = 5$.

i The value i is an index representing the ith unique term in the corpus. For example, we take all 25 unique terms above and give each term an index value. The term "all" might get an index of $i = 1$, the term "apple" an index of $i = 2$, and the term "granny" an index of $i = 21$.

m The number of unique terms that appear in the corpus. There are 25 unique terms in the "Granny Smith apples" corpus.

r_j A boolean value representing whether or not document j is relevant. If $r_j = 1$, document j is relevant. If $r_j = 0$, document j is irrelevant. All documents are are assumed to be either relevant or irrelevant. $\mathbf{r}$ is a vector representing all of the relevance values. The dimensions of $\mathbf{r}$ is equal to the number of documents in our corpus, n. For us, $\mathbf{r} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix}$.

d_i^j The value for d_i^j represents the number of times that the ith term occurs in document j. For example, d_{21}^4 counts term 21, the term "granny", in document 4 (Granny Smith apples are green and Granny Smith apples are wide.). It has the value 2. The vector $\mathbf{d^j}$ consists of all of the d_i^j values for a particular document j, and represents everything we know about that document. The vector $\mathbf{d^j}$ has as many elements as there are unique terms (m). All of the documents together can be represented in an $m \times n$ matrix $\mathbf{D}$ with rows $\mathbf{d^j}$.

$p^{\mathbf{rel}}$ The probability that a document is relevant. It is the value for $\Pr(r_j = 1)$. While we don't necessarily know the true probability of relevance, we can find the empirical probability of relevance. For our example corpus of 5 documents, since there are 2 relevant documents, the empirical probability is $p^{\mathrm{rel}} = \frac{2}{5} = 0.4$. Note that the probability that a document is irrelevant is (1-p^{rel}).

θ_i This value represents the probability of drawing term i from the relevant probability distribution. In the example we discussed it was 0.25 for the term "apple" and 0.0001 for the term "pie". The relevant term distribution can be expressed as a vector, θ.

$\hat{\theta}_i$ Similarly, the value $\hat{\theta}$ represents the probability of drawing term i from the irrelevant distribution. The irrelevant distribution can also be expressed as a vector, $\hat{\theta}$.

Ranking

Recall that in a probabilistic model, we are interested in the probability that a document is relevant, since we will rank our results by this value. In Chapter 2 we saw in Equation 2.1 that we can use Bayes rule to transform the problem into that of finding the probability a document was generated from the relevant distribution.

$$\Pr(r_j = 1|\mathbf{d^j}) = \frac{\Pr(\mathbf{d^j}|r_j = 1)\Pr(r_j)}{\Pr(\mathbf{d^j})}$$

Since we are interested in ranking specifically by the probability of relevance (as opposed to the probability of irrelevance), let us consider the case where $r_j = 1$. We know that $\Pr(r_j = 1) = p^{\text{rel}}$. We also know that $\Pr(\mathbf{d^j}) = \Pr(r_j = 1)\Pr(\mathbf{d^j}|r_j = 1) + \Pr(r_j = 0)\Pr(\mathbf{d^j}|r_j = 0)$. Thus

$$\Pr(r_j|\mathbf{d^j}) = \frac{p^{\text{rel}}\Pr(\mathbf{d^j}|r_j = 1)}{p^{\text{rel}}\Pr(\mathbf{d^j}|r_j = 1) + (1 - p^{\text{rel}})\Pr(\mathbf{d^j}|r_j = 0)}$$

Given all of this information, the only thing left for us in order to find $\Pr(r_j|\mathbf{d_i^j})$ is to understand the probability that a document was drawn from a certain distribution, $\Pr(\mathbf{d_i^j}|r_j = 1)$ and $\Pr(\mathbf{d_i^j}|r_j = 0)$.

Given the assumptions we have for how our corpus was created, by independently sampling from a relevant or irrelevant probability distribution, we can express the probability of seeing a document given it was generated from a particular probability distribution as merely the product of the probability of seeing each term it contains. As we mentioned earlier, this is one of the advantages of assuming term independence. What we find for the probability that a document was generated from the relevant distribution is:

$$\begin{aligned}\Pr(\mathbf{d^j}|r_j = 1) &\propto \prod_{i=1}^{m} \Pr(d_i^j|r_j = 1)\\ &\propto \prod_{i=1}^{m} \theta_i^{d_i^j}\end{aligned}$$

Not included in the above equation is the multinomial coefficient that should be included because we are treating the document as a bag of words where word order does not matter. However, it is a constant, so we will ignore it for now. The probability that a document was generated from the irrelevant distribution can similarly be expressed as $\prod_{i=1}^{m} \hat{\theta}_i^{d_i^j}$.

	Relevant	Irrelevant
all	0.02	0.05
apple	0.16	0.18
away	0.02	0.03
cider	0.03	0.01
cultivate	0.02	0.01
day	0.01	0.04
doctor	0.01	0.02
eat	0.02	0.06
fruit	0.05	0.04
granny	0.09	0.05
green	0.08	0.04
juice	0.02	0.01
keep	0.02	0.06
lot	0.01	0.04
make	0.03	0.03
most	0.03	0.04
pie	0.02	0.04
red	0.04	0.01
sauce	0.03	0.04
smith	0.11	0.05
strüdel	0.02	0.02
tree	0.05	0.04
use	0.01	0.02
washington	0.03	0.03
wide	0.07	0.04

Table 3.1: The underlying probability that a term will occur in relevant and irrelevant documents our "Granny Smith apples" corpus (θ). Given we know these values, finding the probability a document is relevant is straight forward. Estimating these values is fundamental to retrieving a document using the multinomial model.

Now it is simple to calculate the value we need for ranking within the multinomial model given that we know the relevant and irrelevant term distributions:

$$\Pr(r_j = 1|\mathbf{d^j}) = \frac{p^{\text{rel}} \prod_{i=1}^{m} \theta_i^{d_i^j}}{p^{\text{rel}} \prod_{i=1}^{m} \theta_i^{d_i^j} + (1 - p^{\text{rel}}) \prod_{i=1}^{m} \hat{\theta}_i^{d_i^j}} \tag{3.1}$$

Example

Let us say we do know the distributions that the relevant and irrelevant documents in our example "Granny Smith apple" corpus was generated from. The distributions we assume can be found in Table 3.1. Given the distributions, it is easy to find the posterior probability of relevance of each document in the corpus. For brevity, when finding the probability of relevance of a document we show only the terms that occur in that document. Terms not appearing in the document can safely be ignored. If a term that does not appear in a document it has a value $d_i^j = 0$. Since $\theta_i^0 = 1$, these terms will not factor into the product in Equation 3.2.

1. $\Pr(\mathbf{d^1}|r_1 = 1) = \frac{0.4(0.02 \cdot 0.16 \cdot 0.02 \cdot 0.05 \cdot 0.03 \cdot 0.05 \cdot 0.07)}{0.4(0.02 \cdot 0.16 \cdot 0.02 \cdot 0.05 \cdot 0.03 \cdot 0.05 \cdot 0.07)+0.6(0.05 \cdot 0.16 \cdot 0.01 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04)} = 0.49$

2. $\Pr(\mathbf{d^2}|r_2 = 1) = \frac{0.4(0.16^2 \cdot 0.02 \cdot 0.01 \cdot 0.01 \cdot 0.02 \cdot 0.02 \cdot 0.01)}{0.4(0.16^2 \cdot 0.02 \cdot 0.01 \cdot 0.01 \cdot 0.02 \cdot 0.02 \cdot 0.01)+0.6(0.16^2 \cdot 0.03 \cdot 0.04 \cdot 0.02 \cdot 0.06 \cdot 0.06 \cdot 0.04)} = 0.00$

3. $\Pr(\mathbf{d^3}|r_3 = 1) = \frac{0.4(0.16^2 \cdot 0.09 \cdot 0.08 \cdot 0.04 \cdot 0.11 \cdot 0.03)}{0.4(0.16^2 \cdot 0.09 \cdot 0.08 \cdot 0.04 \cdot 0.11 \cdot 0.03)+0.6(0.16^2 \cdot 0.05 \cdot 0.04 \cdot 0.01 \cdot 0.05 \cdot 0.03)} = 0.94$

4. $\Pr(\mathbf{d^4}|r_4 = 1) = \frac{0.4(0.16^2 \cdot 0.09^2 \cdot 0.08 \cdot 0.11^2 \cdot 0.07)}{0.4(0.16^2 \cdot 0.09^2 \cdot 0.08 \cdot 0.11^2 \cdot 0.07)+0.6(0.16^2 \cdot 0.05^2 \cdot 0.04 \cdot 0.05^2 \cdot 0.04)} = 0.97$

5. $\Pr(\mathbf{d^5}|r_5 = 1) = \frac{0.4(0.16^6 \cdot 0.03 \cdot 0.02 \cdot 0.03 \cdot 0.02 \cdot 0.03 \cdot 0.02 \cdot 0.01)}{0.4(0.16^6 \cdot 0.03 \cdot 0.02 \cdot 0.03 \cdot 0.02 \cdot 0.03 \cdot 0.02 \cdot 0.01)+0.6(0.16^6 \cdot 0.01 \cdot 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.04 \cdot 0.02 \cdot 0.02)} = 0.27$

We see that document 4 has the highest probability of being relevant, followed by document 3, and so on. The results seem reasonable.

3.1.3 Efficiency

Let us pause briefly to consider the efficiency with which we can expect to perform retrieval using this ranking function. We will consider for now the case where we know the appropriate distributions. Dividing Equation 3.1 through by $\prod_{i=1}^m \theta_i^{d_i^j}$ we get:

$$\Pr(r_j = 1|\mathbf{d_i^j}) = \frac{p^{\text{rel}}}{p^{\text{rel}} + (1 - p^{\text{rel}}) \prod_{i=1}^m \left(\frac{\hat{\theta}_i}{\theta_i}\right)^{d_i^j}} \tag{3.2}$$

This equation gives us an absolute value for the probability of relevance of a document, and enables us to provide a cardinal ranking. However, if we only care about the order that we rank documents, then we could consider an ordinal ranking to be sufficient. In this case we would only care about whether a document was more likely to be relevant than another document, and not by how much. You will note that in Equation 3.2 $\Pr(r_j = 1|\mathbf{d^j})$ is monotonically decreasing with respect to $\prod_{i=1}^{m}\left(\frac{\hat{\theta}_i}{\theta_i}\right)^{d_i^j}$. This means it is monotonically increasing with its inverse, $\prod_{i=1}^{m}\left(\frac{\theta_i}{\hat{\theta}_i}\right)^{d_i^j}$. Ranking by this value will give us the same results as ranking by the posterior probability of relevance, $\Pr(r_j = 1|\mathbf{d^j})$.

This ranking value highlights one of the advantages of the very simple model that we have described. A document's rank can be found by looking at only one parameter per term, per distribution. We can go further than that, however. As we saw in the example, terms not appearing in a document can safely be ignored. This means that to find the value by which we will rank a document, we need only consider two values for each term that appears in that document. In practice, we may even assume that for most terms, those which are not content terms, the probability of the term appearing in a relevant document is equal to the probability that it appears in irrelevant documents. This allows us to ignore these terms as well, since if $\theta_i = \hat{\theta}_i$ we know $\left(\frac{\hat{\theta}_i}{\theta_i}\right)^{d_i^j} = 1$. We will elaborate on how we can ignore most terms in practice further in Section 3.2.5.

3.2 Finding the Term Distributions

Given the equation that we will be ranking by is a function of θ and $\hat{\theta}$ you can see that finding the relevant and irrelevant term distributions is at the heart of being able to retrieve using this model. In this section we will explain how we estimate these values.

First we will give a straight forward example of one way to estimate the distributions using our "Granny Smith apples" corpus. We will go through an example of retrieving with these estimates. The example will highlight some of the problems that arise when trying to estimate the distribution. We will then discuss how we solve the problems that arise using a Bayesian technique called a prior. We will explain how we make our initial estimates of the term distributions given that we are not provided in advance with the relevance judgments that we have for our example corpus, and talk about how we use machine learning in an attempt to improve on those initial estimates.

3.2.1 Example

Let us consider the information we have available to us to estimate the term distribution for our example corpus. We know the empirical values for the probability that a term occurs in a relevant or irrelevant document in that corpus. One way we could estimate the relevant and irrelevant distribution would be to take the empirical distribution to represent the actual distribution.

Let us look at what these distributions look like given that we are estimating the distributions in this way. There are nine occurrences of the term “apple” in the irrelevant documents, and 28 term occurrences in total in the irrelevant documents. This means that our estimate for the probability that you will draw a term from the irrelevant distribution when creating the documents is $\frac{9}{28} = 0.32$. We can find the probability of drawing the term “apple” from the relevant distribution as well. There are four occurrences of the term in the relevant documents, and 15 terms total, meaning that we estimate the term has a probability of $\frac{4}{15} = 0.27$ of being selected to occur in relevant documents. The empirical distribution can be seen in Table 3.2.

On the other hand, we will estimate that “smith” has a probability of appearing in the irrelevant documents $\frac{0}{28} = 0.00$ and $\frac{3}{15} = 0.20$ in the relevant documents, which is a significant difference. Note that there is a much larger difference between the estimated probability that “smith” will be drawn from a particular distribution (0.00 versus 0.20) compared to the estimated probability that “apple” will be drawn (0.32 versus 0.27). This could be interpreted to reflect that “smith” is a more descriptive term than “apple”.

Let us look at the results we will get using the empirical term probability dis-

	Relevant	Irrelevant
all	0.00	0.04
apple	0.27	0.32
away	0.00	0.04
cider	0.00	0.04
cultivate	0.00	0.04
day	0.00	0.04
doctor	0.00	0.04
eat	0.00	0.04
fruit	0.00	0.04
granny	0.20	0.00
green	0.13	0.00
juice	0.00	0.04
keep	0.00	0.04
lot	0.00	0.04
make	0.00	0.04
most	0.00	0.04
pie	0.00	0.04
red	0.07	0.00
sauce	0.00	0.04
smith	0.20	0.00
strüdel	0.00	0.04
tree	0.00	0.04
use	0.00	0.04
washington	0.00	0.04
wide	0.07	0.04

Table 3.2: The empirical probability that a term will occur in relevant and irrelevant documents in our "Granny Smith apples" corpus. While using the empirical probability of term occurrence is one way to estimate the underlying probabilities, it requires large amounts of labelled data to do suitably, and has several other problems as well.

tribution we can estimate from our example corpus to retrieve documents from our example corpus. Again, for brevity, we will only show the terms that appear in each document.

1. $\Pr(\mathbf{d^1}|r_1 = 1) = \frac{0.4(0 \cdot 0.27 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)}{0.4(0 \cdot 0.27 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)+0.6(0.04 \cdot 0.32 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04)} = 0$
2. $\Pr(\mathbf{d^2}|r_2 = 1) = \frac{0.4(0.27^2 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)}{0.4(0.27^2 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)+0.6(0.32^2 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04)} = 0$
3. $\Pr(\mathbf{d^3}|r_3 = 1) = \frac{0.4(0.27^2 \cdot 0.20 \cdot 0.13 \cdot 0.07 \cdot 0.20 \cdot 0.07)}{0.4(0.27^2 \cdot 0.20 \cdot 0.13 \cdot 0.07 \cdot 0.20 \cdot 0.07)+0.6(0.32^2 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)} = 1$
4. $\Pr(\mathbf{d^4}|r_4 = 1) = \frac{0.4(0.27^2 \cdot 0.20^2 \cdot 0.13 \cdot 0.20^2 \cdot 0.07)}{0.4(0.27^2 \cdot 0.20^2 \cdot 0.13 \cdot 0.20^2 \cdot 0.07)+0.6(0.32^2 \cdot 0^2 \cdot 0 \cdot 0^2 \cdot 0)} = 1$
5. $\Pr(\mathbf{d^5}|r_5 = 1) = \frac{0.4(0.27^6 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)}{0.4(0.27^6 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0)+0.6(0.32^6 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04 \cdot 0.04)} = 0$

While this produces a ranking we like (documents 3 and 4 are ranked first, and then the rest of the documents), there are some obvious problems with these results.

3.2.2 Problems

The examples makes clear a couple of difficulties with the probabilistic model as we've described it on top of the obvious problem of where we get perfectly labeled documents from which to estimate our term distributions. First, note that since we are dealing with very small values the numbers can get very small, very quickly. To solve this problem, something that is often done is to take the log of the probability function. For example, taking the log of $Pr(\mathbf{d^j}|r_j)$ would give us $\sum_i d_i^j \log(\theta_i)$. Recall that if we are only performing an ordinal ranking, this is a fine thing to do, since the log function is monotonically increasing. Note that small numbers are only really a "problem" during implementation, but we also often find it convenient during analysis to use sums instead of products.

Additionally, you have probably noticed that because some terms never occur in the relevant document set, we estimate that they have a zero probability of occurring in relevant documents. This means that if we have not seen a term occurring in any of our labeled relevant information during the distribution estimation process, we will end up giving the lowest possible probability of relevance to any document that contains that term. We'll end up doing this even if the document has other strong evidence supporting its relevance. If a new document, Granny Smith apples are fruit. were introduced, it would get a very low ranking since "fruit" has not been seen in any relevant documents. By taking the empirical probabilities of terms occurring to be the actual probabilities, we have obviously made an error. We have over-fit our estimates to the data. For this reason this problem is called "over-fitting".

The method that we have just used to estimate the probability distribution is a classical approach. This means that we take only the information immediately available to us, and estimate the distribution from that. But that has the problem of giving some terms a zero probability of occurring when we know that must not be the case. While classical models overcome this problem with heuristics, we will take a more principled approach, the Bayesian approach. In a Bayesian model we would include the prior belief we have about term occurrence in addition to the information available to us. We would place a prior expectation of seeing any term to be somewhat greater than 0.

Another potential problem arises regarding document length. As we discussed, the probability of a document given that it was generated by the relevant term distribution ($\Pr(\mathbf{d^j}|r_j = 1)$), is the product of the probability of the occurrence of each term in the document. Since the probability of a term occurring in a document is less than one, this value will be significantly smaller the more terms there are in a document. That this value is smaller for longer documents shouldn't be a problem, since it is normalized by the probability that the document was generated by the irrelevant term distribution ($\Pr(\mathbf{d^j}|r_j = 0)$), which is also smaller.

However, the longer the document, the greater the variation you will see in the probability of a document being generated from either distribution. There are more terms in the document, so we are more sure about whether the document is relevant or not. Small differences in θ_i and $\hat{\theta}_i$ will accumulate to make for a large difference in the overall probability. This isn't necessarily a bad thing, but is something to consider. In more complex models, there is often much done to deal with length, such as document length normalization.

3.2.3 Prior

Let us understand what it means to have a prior belief about something. Imagine that you had a coin, and when you flipped it, it came up either heads or tails. When you get the coin, you think it is probably a fair coin, meaning that you expect to get heads with equal probability of getting tails. But if you flipped it 10 times, and each time it came up heads, you might start to think that the coin was biased. On the other hand, if you first flipped the coin two hundred times, of which only hundred of those flips came up heads, you might think that the ten heads in a row is just a statistical fluke. In fact, seven heads in a row is very likely in two hundred flips, and ten heads is not all that unlikely. Likewise, if your mother gave you the coin, and promised that it was fair, meaning you had a very strong prior belief in the coin being fair, then you might also be willing to accept the 10 heads as a statistical fluke, and not vary your expectation of whether the coin will come up heads or tails too drastically.

We can express our belief about the coin by saying that we have some prior probability distribution on what we expect from the coin. This distribution we will as a distribution over the random variable θ. We can tell you whether we expect the next coin flip to be heads or tails by telling you what is likely given the probability distribution over θ. We express the prior distribution θ mathematically with a Beta prior. The Beta prior is described by two parameters: α_h, which measures our prior belief that the coin will come up heads, and α_t represent our prior belief that the coin will come up tails. We also let $\alpha = \alpha_h + \alpha_t$. One way to understand these values is to view them as the imagined number of times that you've already seen the coin come up either heads or tails.

θ is a random variable that represents the probability that the coin will come up heads. However, we don't actually know what value θ is; we want to estimate it. What we do know is the probability distribution for θ, which is a Beta distribution:

$$p(\theta) = \mathrm{Beta}(\theta|\alpha_h, \alpha_t) = \frac{\Gamma(\alpha)}{\Gamma(\alpha_h)\Gamma(\alpha_t)}\theta^{\alpha_h-1}(1-\theta)^{\alpha_t-1}$$

In this equation, $\Gamma(\cdot)$ is the Gamma function which satisfies $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(1) = 1$.

To find what we believe our odds of getting heads on the next flip, we integrate over θ. By solving $\int \theta \text{Beta}(\theta|\alpha_h, \alpha_t)d\theta$ we find that the probability of the coin landing heads is equal to the expectation of θ, $E[\theta] = \frac{\alpha_h}{\alpha}$.

What is convenient about the Beta prior is that it can be updated easily with new observations. If we observe h heads and t tails after flipping the coin several times, and had a prior $\text{Beta}(\alpha_h, \alpha_t)$, then our posterior $\Pr(\theta) = \text{Beta}(\alpha_h + h, \alpha_t + t)$. This means that we can update our belief about the probability of flipping a head to be $\frac{\alpha_h+h}{\alpha+h+t}$.

Let's go over the example we gave earlier, this time using the specifics of the Beta prior to illustrate the example. Suppose we get a coin from a shady looking character, we might have a prior belief that the coin is fair, but we don't feel very strongly about it. In this case we might choose $\alpha_h = 4$ and $\alpha_t = 4$. If we flip the coin, and get ten heads in a row ($t = 0$, $h = 10$), then our expectation for getting a head the next time we flip the coin would be $\frac{4+10}{4+4+10+0} = 0.78$. The coin no longer looks very fair. But if we were to have flipped the coin 200 times first, and gotten 100 heads and tails each, with our new observations we only arrive at $t = 100$ and $h = 100 + 10$. In this case we would expect the probability of flipping a head to be $\frac{4+110}{4+4+110+100} = 0.52$, or still basically fair. Similarly, if our mother gave us the coin, we might assign a prior of $\alpha_h = 100$ and $\alpha_t = 100$. Now, even if we flip ten heads straight off the bat ($t = 0$, $h = 10$), we still think the coin is basically fair ($\frac{100+10}{100+100+10+0} = 0.52$).

The Beta prior can be generalized to the multinomial case, in which case it is called a Dirichlet prior. We place a Dirichlet prior on the relevant term distribution. For each unique term i we assign a prior belief of the weight that term carries in the relevant distribution to an associated variable, α_i. We make α_i proportional to the number of times that that term occurs in our corpus.

$$p(\theta) = \text{Dir}(\theta|\alpha_1, \alpha_2, \ldots, \alpha_r) = \frac{\Gamma(\alpha)}{\prod_{i=1}^{r} \Gamma(\alpha_i)} \prod_{i=1}^{r} \theta_i^{\alpha_i - 1}$$

Integrating this, we find our prior belief of drawing term i from the relevant probability distribution is $\frac{\alpha_k}{\alpha}$. As in the coin-flipping example, where we observed whether we rolled a head or a tail, as we perform learning over our corpus, we will observe the number of times that term i occurs in relevant documents to update our belief as to the probability of drawing term i from the distribution. Let us represent the number of times we see term i occurring in a relevant document as s_i ($\sum_i s_i = s$). Again, like the Beta distribution, the Dirichlet distribution has a nice update rule. After our observations of s_i, the expected value for the probability of drawing term i from the relevant distribution is simple to compute ($\frac{\alpha_i+s_i}{\alpha+s}$). For a more in depth discussion of Beta and Dirichlet priors, see Heckerman's tutorial [17]. In subsequent discussion we will refer to our expectation of θ_i as ϑ_i.

3.2.4 Initial distribution estimate

We want to make an estimate about the term distribution. The only information that we have available to us is the distribution of terms throughout the corpus and the query. We know nothing specifically about the relevant or irrelevant distributions, but that does not mean that we want to take the classical approach and say we know nothing about the probability of a term appearing. After all, we do know that the term "apple" is probably more likely to appear in general than the term "bailiwick". Using our knowledge of priors, let us first talk about how we will estimate the irrelevant distribution, and then talk about how we will estimate the relevant distribution.

Irrelevant distribution

We can begin our estimation of the irrelevant term distributions by making the assumption that by far most documents are irrelevant. This implies that the irrelevant term distribution should more or less equal to the corpus term distribution. Because of this, if the probability of seeing a term in the corpus at large is 0.0001, then we will take the probability of that term being selected when an irrelevant document is generated to also be 0.0001.

In our "Granny Smith apples" example, the probability of seeing the term "green" throughout the entire corpus is $\frac{2}{43} = 0.0465$, and this is what we would take as the probability of seeing the term "green" in an irrelevant document. This value, in our example case, is clearly very far from the 0 probability the term "green" empirically has of appearing in irrelevant documents. This is because the assumption that there are many more irrelevant documents than relevant documents doesn't hold, and also because the corpus is rather small to be making any sort of generalization with.

Relevant distribution

We have described how we estimate the irrelevant term distribution. We still need to find the relevant term distribution. In estimating the relevant term distribution, we will assume that the query, having been hand generated by the user, contains terms that are particularly representative of the relative term distribution. If we were to assert that the query term distribution represents the entire relevant term distribution, then we will encounter the problem we discussed in Section 3.1.2, namely that the probability of most terms appearing in relevant documents will be 0, causing the probability of relevance of every document to be 0.

To avoid this problem we will start with some non-zero probability of seeing each term in relevant documents. For example, one thing we could do is to also say that the relevant distribution is equal to the corpus distribution, with the exception that each query term's probability of appearing increased in some way. The corpus distribution could represents our prior belief, before seeing the query terms, about what the relevant term distribution looks like. We could then update this prior belief with our belief that the query terms are particularly likely in the relevant distribution. Once we have this starting point, we would be able to do retrieval with this initial distribution, or we could use it as a starting point for unsupervised learning of the distribution.

One way to incorporate the query is to directly alter the relevant term distribution. The query consists of hand selected words that have been determined by the user as being, we assume, highly representative of the relevant distribution. As we have said, we have a prior belief of the distribution based on the corpus distribution. By fiddling with the prior, such as, for example, by doubling the prior probability of the query terms appearing, these terms can be made more likely to appear in relevant documents. This allows for a very direct control of the way that the query effects the framework.

Another way to deal with the query is to treat it as a relevant document. We could assume that the user generates the query terms by sampling from the relevant distribution, and then the user tells the system that this "document" is relevant. While this fits nicely into the framework it does not express the importance of the query as clearly. This results in some problems relating to learning that we will discuss in Section 3.2.7.

The first approach is nice because it expresses clearly the true importance of the query terms. The second approach is nice because it somehow seems less arbitrary than fiddling with numbers. The system is currently implemented using the second approach. However, with the understanding that the terms a query contains are particularly important, we have experimented with treating it like several documents, instead of just one, and we will discus the effect that this has on the results later.

3.2.5 Efficiency

We saw earlier that using the multinomial model we can produce search results very quickly. We will now discuss the details of how this initial distribution estimate allows us to retrieve documents with performance time comparable to the search algorithms most commonly used in practice.

We saw earlier in Section 3.1.3 that an ordinal ranking can be obtained using $\prod_{i=1}^{m}\left(\frac{\hat{\theta}_i}{\theta_i}\right)^{d_i^j}$. Let us begin our investigation into this value by looking separately at how our estimate for the probability of a term i occurring in the relevant and irrelevant distributions, namely ϑ_i and $\hat{\vartheta}_i$. Initially, before we incorporate the query, $\vartheta_i = \hat{\vartheta}_i$. Note that $\hat{\vartheta}_i$ actually never changes, since we have such a strong prior belief in it, so it will always be equal to our initial estimate of ϑ_i.

On the other hand, our estimate for ϑ_i changes after we observe the query. Recall from our discussion of priors that we have an initial value for our prior belief of seeing term i, α_i, such that we believe the value for ϑ_i is $\frac{\alpha_i}{\alpha}$. Let q_i represent the number of times a term i appears in the query, and $q = \sum_i q_i$. Since the query is the only thing we will have observed, we know that

$$\vartheta_i = \frac{\alpha_i + q_i}{\alpha + q}$$

Let us return to the value from Equation 3.2 we said we were ranking by, and express it in terms of αs and qs.

$$\begin{aligned}
\prod_{i=1}^{m} \frac{\hat{\vartheta}_i^{d_i^j}}{\vartheta_i^{d_i^j}} &= \prod_{i=1}^{m} \frac{\left(\frac{\alpha_i+q_i}{\alpha+q}\right)^{d_i^j}}{\left(\frac{\alpha_i}{\alpha}\right)^{d_i^j}} \\
&= \prod_{i=1}^{m} 1 \left(\frac{\alpha}{\alpha+q}\right)^{d_i^j} \left(\frac{\alpha_i+q_i}{\alpha_i}\right)^{d_i^j}
\end{aligned}$$

The first part of this equation is constant for all documents, and can therefore safely be ignored for the purpose of ranking, leaving us with a simple ranking function:

$$\propto \prod_{i=1}^{m} \left(\frac{\alpha_i + q_i}{\alpha_i}\right)^{d_i^j}$$

Most terms aren't query terms. For these terms $q_i = 0$, and $\frac{\alpha_i+q_i}{\alpha_i} = 1$, meaning that these terms will not effect our results. Additionally, most query terms occur only once. With these assumptions, we find that we can rank documents with the following function:

$$\prod_{i \in q} \left(1 + \frac{1}{\alpha_i}\right)^{d_i^j} \tag{3.3}$$

Essentially, each term i appearing in a document contributes $\frac{1}{\alpha_i}$ to the document's score. Since α_i is a function of the corpus frequency of a document, this gives us a reasonable motivation to use the inverse corpus frequency for relevant documents. By taking the log of this value we get a vector space variation, where the weight of each term i is $\log(1 + \frac{1}{\alpha_i})$. A vector space model with this term weighting would produce the same ranking as Equation 3.3 does, both giving the document a ranking of $\sum_{i \in q} d_i^j \log(1 + \frac{1}{\alpha_i})$.

3.2.6 Learning the distribution

Let us now look at how we learn the relevant term distribution by updating the initial term distribution we discuss in Section 3.2.4. Learning is a relatively slow process, and if we attempt to learn the relevant distribution at retrieval time, we will no longer be able to find results as quickly as we do using a vector space model. However, we use learning to better understand our model. If our model is a good one, we should be able to improve performance by arriving at a better estimate for the term distributions.

We will use a simple machine learning technique called "Expectation Maximization" (EM). Expectation maximization is a greedy algorithm that performs simple iterative hill climbing from whatever starting point it is given to find a nearby local maximum. As we discussed in Section 2.2.3, we will use EM in this case to find the relevant probability distribution that makes our corpus most likely, given our multinomial generative model.

Here we discuss in detail how we use EM to find the relevant term distribution that makes the corpus most likely. During the "expectation" step we compute what we expect to see from the information we have. The "expectation" step represents the results we would compute if we didn't have any opportunity for learning. During the "maximization" step we force the likelihood of the corpus to increase by optimizing the values for the relevant term distribution. The learning process involves iterating over these two steps until the likelihood of the corpus given the relevant distribution stops changing, or some other stopping condition is reached.

Expectation

In the expectation step, we find the best relevance judgments that we would expect given the term distribution we have at this point. This means finding the posterior probability of relevance for the document, which we describe in Equation 3.1. With the most current Dirichlet distribution over the θ_is, we can find the expected probability that we would see term i in a relevant or irrelevant document, ϑ and $\hat{\vartheta}$. Plugging these values into Equation 3.1 we can compute the posterior probability of relevance for each document as follows.

$$\Pr(r_j = 1|\mathbf{d^j}) = \frac{p^{\text{rel}} \prod_{i=1}^{m} \vartheta_i^{d_i^j}}{p^{\text{rel}} \prod_{i=1}^{m} \vartheta_i^{d_i^j} + (1 - p^{\text{rel}}) \prod_{i=1}^{m} \hat{\vartheta}_i^{d_i^j}}$$

Note that since we have a Dirichlet prior, we could update our belief about seeing a term throughout the expectation step, after each new observation of a term. However, EM does not traditionally do this since it is a more general learning technique that does not require a Bayesian prior.

Maximization

In the "maximization" step of our learning, we maximize the likelihood of observing our corpus by altering the Dirichlet distributions. Note that, as we have mentioned before, for efficiency we assume there are many more irrelevant documents than relevant documents ($p^{\text{rel}} \ll (1 - p^{\text{rel}})$). This means we will only concern ourselves with updating the relevant distribution on each iteration. The irrelevant document distribution we assume to be so close to the corpus distribution that it is not worth updating; we already have so much prior evidence as to what it is that new information will not tell us anything new.

As we have seen with our discussion of priors, it is easy to update a Dirichlet distribution by adding each observation we have of term i to s_i. Therefore, for the relevant distribution, for each occurrence of each term i in each document j, we have to update s_i. If we were sure that the document the term were in were relevant, we would add the number of occurrences of the term in that document ($s_i = s_i + d_i^j$ for all term in all relevant documents). But we're not sure. We only have our belief about the probability that that document is relevant, the posterior probability of relevance that we computed in the "expectation" step. What we do in this case is add to s_i is the posterior probability of relevance for that document times the observed number of occurrences of term i in that documents:

$$s_i = s_i + \Pr(r_j = 1|\mathbf{d^j}) \cdot d_i^j \qquad \forall i, j$$

3.2.7 Setting Constants

We now understand our implementation of the multinomial model fully. However, even given this model, which defines so much of the problem for us, there is quite a bit of room to tweak different constants in ways that effect the results. In this section we will discuss what constants are available for tweaking, and how we set them. These constants include the prior probability that a document is relevant, the strength of the prior we use for the relevant term distribution, and the strength with which we include the query into the relevant term distribution.

Prior probability of relevance

The prior probability that a document is relevant, p^{rel}, is an important value. Note that for ordinal ranking without learning, this value doesn't matter. However, it does determines the weight we put on our estimate that a document is relevant in comparison to our estimation that a document is irrelevant. And yet in our analysis we have taken this value as a given. How should it be set? Because we do not want to worry about the effect of this value during our experiments, we set it to the correct value for each query. For example, if we were performing a query where 10 documents out of 10,000 were known to be relevant, we would set the prior probability of relevance to be 0.001.

A more principled way that we could set this value would be to represent this value as a Beta distribution. Initially we might expect to see some small number of relevant documents, just as we initially expected our coin to be fair, but as we learn, and find either a lot of documents that appear to be relevant, or a lot of documents that appear to be irrelevant, we could adjust this belief. That way we could vary the actual number we did see somewhat, so that we will not find documents to be unreasonably likely to be relevant or irrelevant. The importance of this variable, and how to best set it is something we're interested in investigating in the future.

Strength of prior

Another constant that we have control over is the strength of our belief in the prior term distribution of relevant documents. We saw in the coin tossing example that weird behavior from a coin given to us by our mother is accounted for much differently than weird behavior in a coin given to us by a shady character. We have already mentioned that the only term distribution we know at the outset is the corpus distribution, and that that distribution is used to establish our prior belief of the term distribution in both relevant and irrelevant documents. We need to incorporate the query terms into that distribution in a way that expresses their importance. Should the observation of the query terms and the subsequent feedback from documents that we judge to be relevant effect our belief about the relevant distribution a lot or a little? Essentially, does our initial belief that the distribution of relevant terms resembles the corpus distribution come from our mother, or a shady character? The strength of our belief is reflected in the magnitude of the α_is.

Initially we assert that since we know that documents have a certain prior probability of being relevant, p^{rel}, we also have a prior belief about how many documents are relevant, $p^{\mathrm{rel}}n$. We set our prior for the relevant term distribution to be as strong as if we had seen $p^{\mathrm{rel}}n$ documents with the corpus distribution. The rationale behind doing this is that if we were to observe the true set of relevant documents, they would have an impact on the distribution, but not overwhelmingly so. However, we have tried scaling this value as well, and will discuss the effect of the strength of the prior on the results in Section 3.3.2.

Strength of query

As we discussed earlier in Section 3.2.4, we treat the query like a labeled document that we know is relevant. However, the query is not actually just a labeled relevant document. It consists of words that have been hand chosen by the user to be representative of the documents he is interested in.

That we treat the query as just a document results in some problems during learning. Since the query is a very short document, it can easily be overpowered by other documents that are also determined to fall into the relevant distribution. For each query term q_i, the related number of observations of the term, s_q, is updated. This reflects the fact that the query term is relevant. However, every other document that is judged at least somewhat relevant has many occurrences of many other terms. Each term i in each slightly document updates its respective s_i as well. Even though the probability of these other documents of being relevant may low, there are so many other documents that the cumulative effect can large. If these longer documents cause some terms to outweigh the query terms, then we could get documents that aren't necessarily related to the query. This is called "topic drift".

For this reason, we have experimented with making the importance of the query stronger as well, and treating it as 5, 10, or even 100 documents. The results from these experiments are discussed in Section 3.3.2.

3.3 Results

Let us look at how the multinomial model performs. As we discussed in Section 2.3, this is the most common point of interaction with the data for IR researchers. In this section first we will talk about the corpus on which we performed our tests. Then we will talk about what sort of results we found, as well as discuss our investigation into why we got the results we did.

3.3.1 Corpus

Because the potential computational complexity when using machine learning and because we wanted to be able to investigate manually the results of the proposed system, we decided to do our initial testing on a small subset of the TREC corpus. We used the Wall Street Journal articles from TREC Volume 1 and 2 that are relevant to the TREC topics 51-150. This includes articles from the years 1987-1992. The resulting corpus consists of 9,465 documents. This corpus is sufficiently small that we can perform the tests that we want to without having to worry too much about efficiency, while being large enough to give us at least some idea of how successful different trials are. From the corpus we removed all stop words and case. For the queries we use in testing, we selected the topic descriptions from the TREC topics 51-150. These are short, 2 to 3 word descriptions of the topic.

Note that our results are somewhat better than what is commonly seen in other information retrieval research. This is likely due to the odd construction of our corpus. We use only documents that are relevant to queries, so the probability of relevance is higher than over a standard TREC corpus. If we were to use the entire TREC corpus, chances are some of the irrelevant documents not included in our corpus would get ranked highly, cluttering the ranking. Larger scale testing is necessary to really be able to compare our results with other systems.

3.3.2 What we found

In this section we will discuss the results we found for retrieval using the multinomial model in two ways. First we will discuss the results we get when ranking using the initial estimate of the relevant term distribution we discuss in Section 3.2.4. We will then talk about what we found after doing learning with our model. What we find is that the learning generally hurts the performance of the model, implying that there is a flaw in the model.

Recall from Section 2.3.1 that the metric we will be using to evaluate the results is a precision-recall graph. Precision is the fraction of retrieved documents that are relevant, while recall is the fraction of all relevant documents that are retrieved. The better the performance, the further up and to the right on the precision-recall graph the results curve will be. Because we are testing over 100 queries, we will average the curve for each query to get a curve that describes the overall behavior of the system.

Posterior probability of relevance

Ranking by the posterior probability of relevance, using the initial term distribution discussed in Section 3.2.4, where we include the query as a relevant document in the corpus term distribution, performed reasonably well. In Figure 3-1 you can see the results compared with tf.idf, the standard ad-hoc retrieval method that we explained earlier in Section 2.2.2. This is not surprising, since in Section 3.2.5 we found it to produce results that are very similar to what we use for tf.idf.

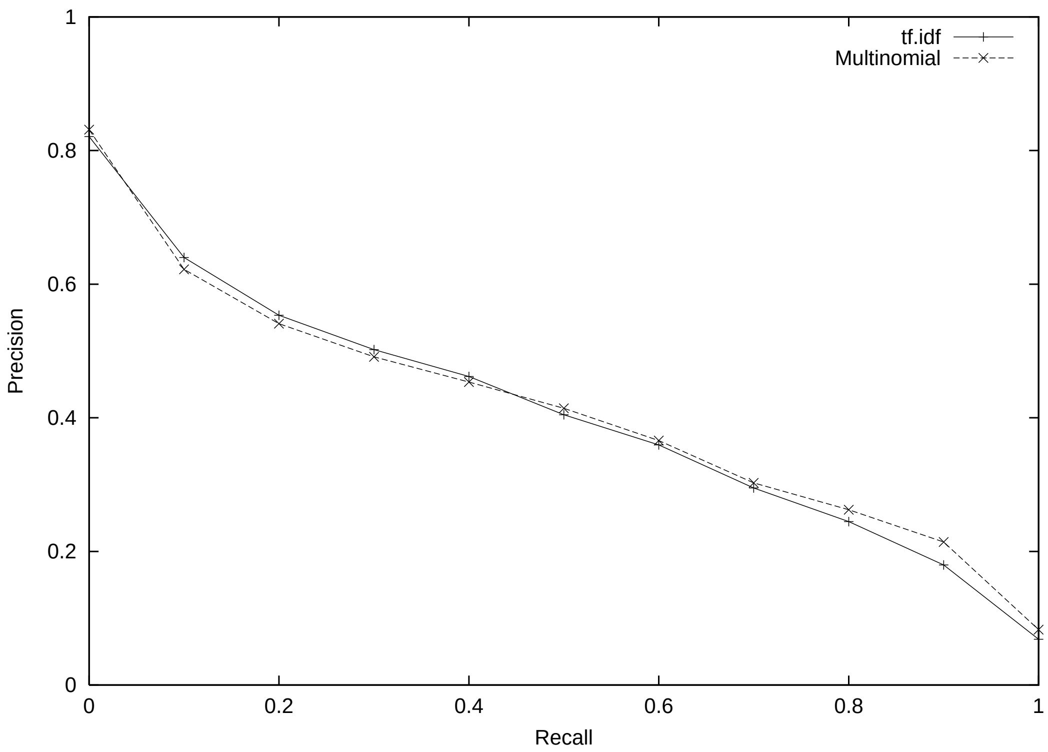


Figure 3-1: Results found using the multinomial model. The relevant distribution used is the initial estimated distribution. Results are compared with tf.idf.

Tuning initial retrieval We also investigated the effect that varying the two constants we discuss earlier, the prior weight and the query weight, has on the results found using the initial distribution estimates. When we refer to the prior weight, we are referring specifically to the number of times stronger we make the prior than what we discuss in Section 3.2.7. When we refer to the query weight we mean the number of relevant documents that the query is counted as. The default for this is one, but we test query weights of up to 1000.

Without any learning, the effect of changing these two variables is the same. This can be seen in Equation 3.3. You can see that doubling the prior for term i, α_i, has the same effect on the ranking as halving the query weight, q_i, would. Thus raising the query weight is the same as lowering the prior probability, and visa versa. We will look, therefore, only at the effect of query weight on the results curve.

In Figure 3-2 we show the results for a variety of query weights. In general, a higher query weight improves precision in the low recall areas, while lowering it in the high recall areas. As you can see, while the query weight does effect results somewhat, it needs to undergo a large change to significantly effect the results. It is interesting that if the query weight is set too high, the results actually get worse. This is because the query as a document totally overshadows all other term distribution information, including the prior probability of seeing the query term. Instead, each query term gets treated as equally informative about the query. "Granny Smith" is no longer being treated as more descriptive than "apples".

With learning

When we tried to learn the relevant distribution, the performance of the model got worse. In Figure 3-3 you can see the performance after various different number of iterations of EM. Each iteration causes worse results. Again, this could imply that something is wrong with our model, or it could imply that something is wrong with our learning algorithm.

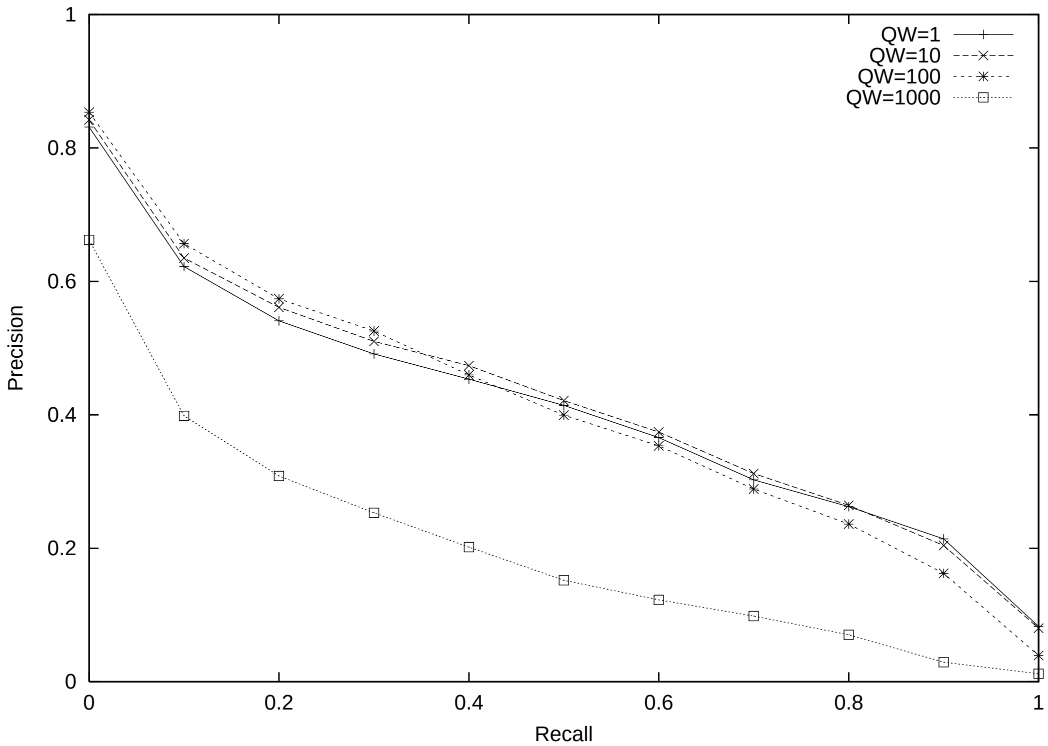


Figure 3-2: Results found using the multinomial model. The relevant distribution is the initial estimated distribution with varying query weights.

Tuning retrieval learning Again, we investigate the effect the different constants we discuss has on results. Setting the prior to be stronger improved performance during the learning processes. This can be seen in Figure 3-4. Here we graph the results after one iteration with different prior knowledge weights. The curves with high prior knowledge weight are very similar to the curves with no learning. This is because our prior is so strong that new information barely effects our expected probability of seeing terms occur in the relevant term distribution.

We also graph the results after one iteration with different query weights. Increasing query weights generally hurt performance, probably for similar reasons that high query weights hurt performance when no learning was performed.

Figure 3-5 shows one of the most successful combination of constant settings that we found. We wanted some generalization to occur, so we did not set the prior weight too high, in which case the curve would look very similar to the curve with no learning. The prior and the query are scaled up the same amount so that with no learning the

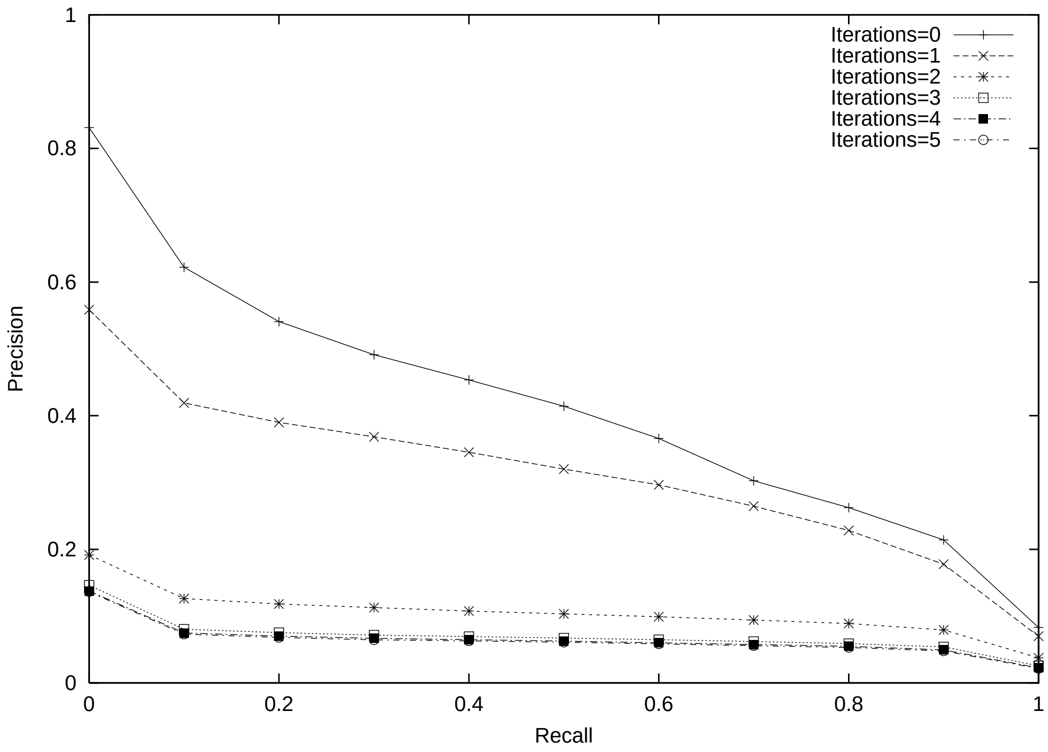
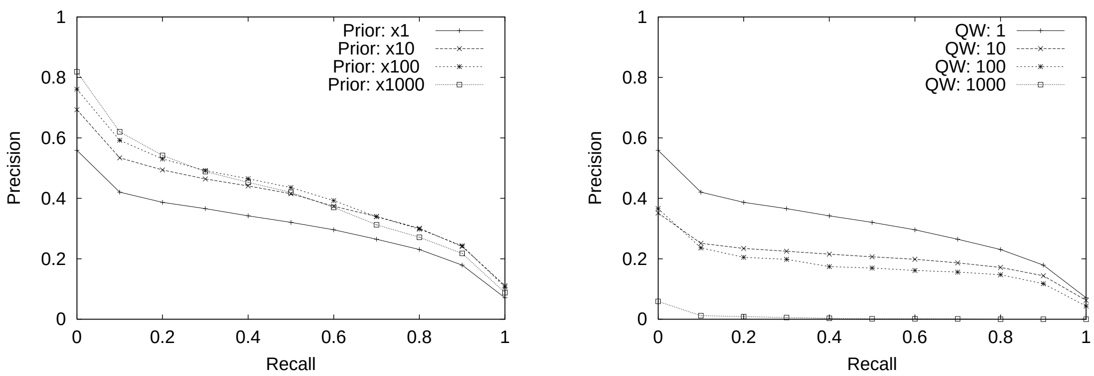


Figure 3-3: Results found using the multinomial model. The relevant distribution is found using learning. Results are compared for different number of iterations during the learning process.



(a) Changing prior knowledge weight

(b) Changing query weight

Figure 3-4: Results found using the multinomial model. The relevant distribution is found using learning, with different values for the prior weight and query weight set.

performance is the same as the performance in Figure 3-1. But you can see that with learning the results are not significantly worse they way there were with our original constant settings.

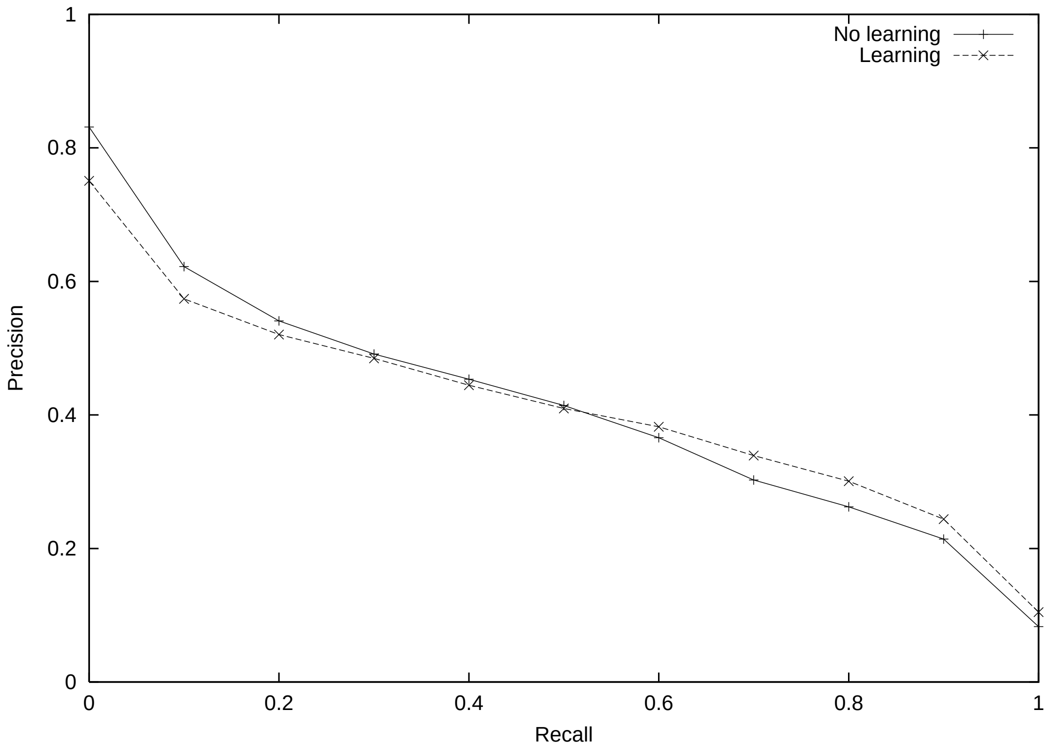


Figure 3-5: Results found using the multinomial model. The relevant distribution is found using learning, with good values for the prior weight and query weight set.

Topic drift Something we can see when looking at the text of the results produced by our retrieval system, and that is also apparent from graphs such as Figure 3-5, is that there is topic expansion/drift. The purpose of machine learning is to generalize, and we can see the generalization happening with the improved precision in the high recall area of the curve. The problem is that it is drifting in a way that hurts precision in the low recall area of the curve. In terms of our example, while without iterations we find documents only on "Granny Smith apples", with learning we generalize to find all documents on "apples". By doing so we find some relevant documents that we didn't know about before, meaning that our recall improves, but the greater number of irrelevant "apple" documents push some of the good, relevant "Granny Smith apples" documents down lower in the ranking, decreasing precision.

When documents are found to be relevant without learning, these documents are generally relevant (high precision in the low recall area). When we iterate, however, other less relevant documents are included because they match on terms that are present in the initial relevant documents that are not descriptive of the query. At times the topic expands and drifts so much that relevant documents are placed with a very low ranking. This is common in relevance feedback systems, but, ideally, with a good model, should not be a problem.

3.3.3 Why we found what we found

That learning a more likely probability distribution for relevant documents hurt the performance of the model implies that there must be something wrong with our system. The problem could stem from one of two sources: the method for estimating the underlying distribution could be flawed, or the model itself could be flawed, and would not be successful even with the correct distribution. We investigate the performance by experimenting with using the "correct" probability distribution to retrieve documents in hopes of understanding where the flaw lies.

Performance with correct distributions

We want to understand if our estimation of the term distributions was at fault for our poor performance, or if the fault actually lies in the multinomial model. For example, our EM algorithm may be reaching a local maximum for the term distribution that is not anywhere near the global one. To understand if this was the case we did several things.

First we created artificial documents according to our model, and ran the learning on them. The learning performed well when given input that was generated as expected. So we know that the model could work if its assumptions were correct.

We then tried setting the relevant distribution to be the actual term distribution in relevant documents. To do this we set the probability of relevance for each document that is relevant to 1. We continued to use a prior, however, so that the results would not be unfairly biased by having some terms have a 0 probability of occurrence.

As you can see in Figure 3-6, the retrieval performance of the multinomial model is actually worse with the correct relevant distribution set. While recall does go up, precision goes way down. The terms in the relevant documents not related to the query are over powering those that are. EM finds a more likely distribution than the correct relevant term distribution from this starting point, so the correct distribution clearly does not maximize the probability of seeing our corpus. The results after learning are even worse.

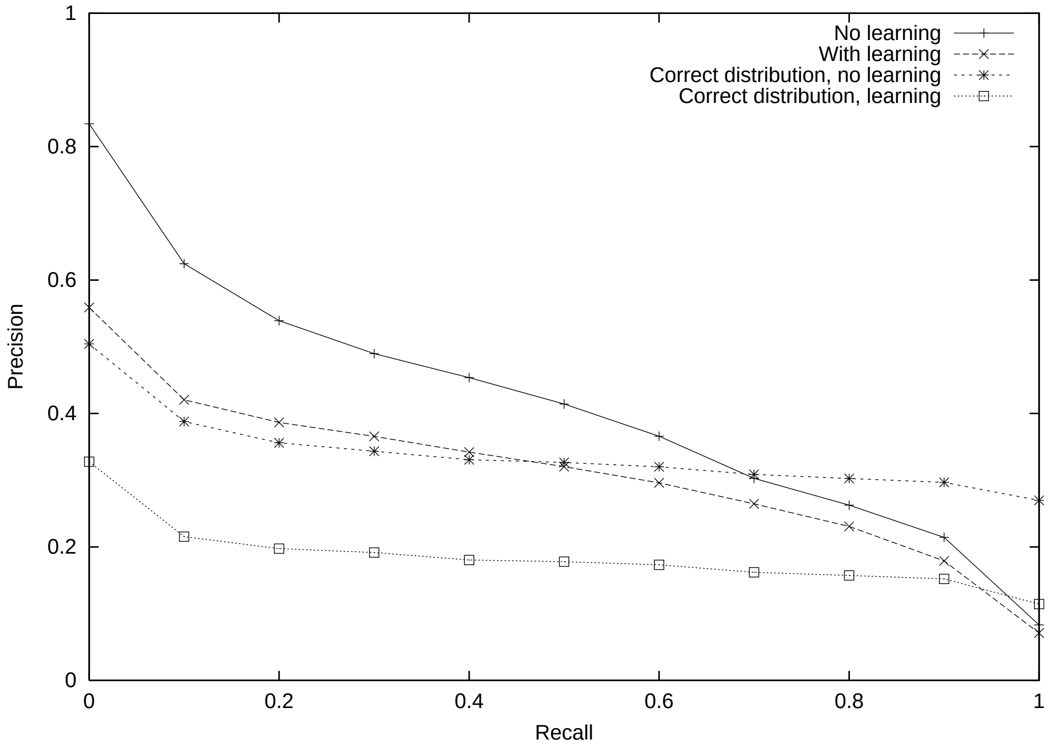


Figure 3-6: Results found using the multinomial model. The relevant distribution is the correct distribution.

Chapter 4

Textual Analysis

We want to use data analysis to understand which assumptions in our multinomial model are the most broken, and work to fix them. As we have argued before, if we make the model match reality better, we should be able to improve retrieval. In order to do this, we have looked at the information that is available in the statistical properties of text in a number of different ways. We will describe what we have found from our analysis in this chapter.

We will focus on looking for a term distribution that explains the entire corpus better than the multinomial distribution. This means that we will not be looking specifically at the relevant and irrelevant distributions. However, it is our hope that by better modeling the corpus distribution, we will also be able to better model the relevant and irrelevant distributions, and use these models to improve retrieval.

Looking only at the corpus term distribution has the advantage of allowing us to work with a large amount of data. If we wanted to look specifically at just the relevant distribution, since we only have a few relevant documents for each query, it would be difficult to draw conclusions based on the information available to us. Using the corpus distribution to understand the two term distributions that we care about is valid if we assume that the relevant and irrelevant distributions can be described by the same model, and that there are many more irrelevant documents in the corpus, so many that they dwarf the effect of the relevant distribution. The first assumption is worth investigating, as we will discuss in the section on future work, Section 6.1.1.

The second is reasonable given the TREC corpus.

In this chapter we will first briefly describe the corpus that we will be using for our textual analysis. We will then talk about our goals in performing textual analysis, describing in further detail specifically which assumption we will attempt to understand better, and what constraints we will continue to impose on our understanding of the data. After explaining the terminology that we use throughout our data analysis, we will talk about some of the ways we find the multinomial model does not match the textual data we analyze. Finally, we will explore what we can learn from the data so that we can build a model that more accurately reflects reality. In the next chapter we will discuss how we incorporate some of what we learn into a probabilistic model and the effect that that has on retrieval.

We will continue to use the "Granny Smith apples" corpus to illustrate our discussion. Recall that it looks like this:

1. all apple cultivate fruit most tree wide
2. apple apple away day doctor eat keep lot
3. apple apple granny green red smith washington
4. apple apple granny granny green smith smith wide
5. apple apple apple apple apple apple cider juice make pie sauce strüdel use

4.1 Corpus

While we used a fairly small corpus to test our multinomial model, for our data analysis we want a larger corpus to work with. We use all of the documents from TREC volume 1. This set includes documents from the Wall Street Journal, the Associated Press, Department of Energy Abstracts, the Federal Register, and Ziff. The document set contains 510,636 documents in total. As with the smaller corpus we use for testing, we remove stop words from the documents, but do not do any stemming. Stemming would reduce the number of terms and increase the number of occurrences for each term, thus increasing the density of the data. However, it would also hide information about the terms [33].

One of the reasons we are interested in working with a larger corpus is that we often want to look at terms that are the same with respect to many of the statistics we will discuss in Section 4.3, such as the length of the document in which the term appears, or the corpus frequency of the term. We will motivate this binning further in the next section. In order to look at such a specific set of terms, and yet still have enough data to generalize about the term occurrence distribution for terms within the bin, we need a large initial corpus.

4.2 Goal of Textual Analysis

While we have said many times in this thesis that we are going to explore the assumptions that the model we build is based on in order to build a well grounded model, we do not actually address every assumption. There are so many assumptions involved in our model that we will only focus on a small subset of them: those that are encompassed within the probabilistic framework, relating simply to term occurrence. We will not question whether stemming or stop word removal is useful. We will not investigate if using something other than words as our terms would be better (such as treating "Granny Smith" as a single term, rather than two). Nor will we question if probabilistic models for information retrieval are valid.

That the data doesn't appear multinomial is not surprising. The multinomial model is a very simple one. It is a model that tries to express something as complex as a large collection of natural language documents by using only one parameter per term. Given this, when we do our data analysis, the question we should be asking is not, "Is it possible to more accurately model the data?" as it surely is given a more complex model, but rather, "Is there a similarly simple model that more accurately models the data?" If we can find this, we should be able to improve the results the model produces while not hurting retrieval efficiency.

Two of the reasons we want to continue to work with a small number of parameters are practical. Using a similar number of parameters ensures that the efficiency of the algorithm will remain comparable. Also, fewer parameters make for simpler, more tractable math. This makes it is easier to understand what is happening within our model. It is best to start with the most straightforward model possible, and then build from there.

Using only one parameter per term can also be motivated from a theoretical standpoint. The more parameters we need to estimate from the data, the more likely we are to over-fit our model. Without an extremely large amount of data it is difficult to estimate an extremely large number of parameters.

In order to impose the requirement that the conclusions we draw from the data allow us to build a simple model, in addition to assuming the corpus has been generated by a probabilistic model, we will continue to assume that terms are independent of each other. For example, we will assume that the occurrence of the terms "granny" and "smith" in a document do not imply that the word "apple" is likely to appear in the document. People have tried to work around the independence assumption with dependence trees [43] and Bayes nets [38]. We will not do this.

Instead, we will continue to assume that there is an underlying distribution for each term that describes the term's occurrence entirely. This distribution can be seen empirically in the term occurrence probability distribution (t.o.p.d.). Given the probabilistic framework we are working in, and the assumption that terms are assumed to be independent, the only information we have about the documents is the number of times each term occurs in a document. Therefore, to understand a term, it makes sense to look at its t.o.p.d. By t.o.p.d. we mean specifically the number of times that the term occurs exactly one time, the number of times it occurs exactly two times, etc.

For example, the term "granny" occurs exactly zero times in three of the documents in our example corpus, exactly one time once, and exactly two times once. For all other occurrences, the term occurs zero times. Figure 4-1 shows the t.o.p.d. for "granny". Throughout our analysis, we will try to discover a better underlying term probability distribution function (t.p.d.f.) that describes the t.o.p.d. that we observe.

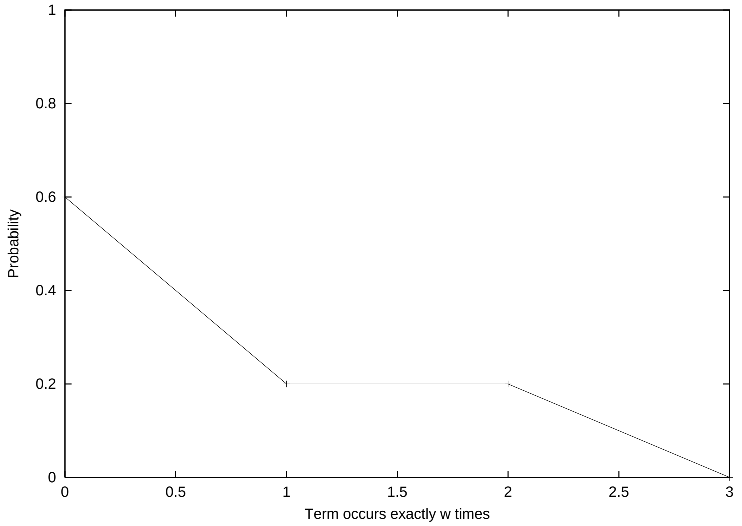


Figure 4-1: An example t.o.p.d. for the term "granny" in our example corpus. By looking at a term's empirical t.o.p.d. we hope to be able to find the underlying hidden parameter we assume describes the term's occurrence.

Specifically, we will look for a t.p.d.f. that can be described with only one parameter. For the multinomial model, we used the hidden parameter of the probability that a term occurs to describe what the multinomial model assumed was a term's t.p.d.f. We will see that the t.p.d.f. postulated by the multinomial model does not match the data.

Let us briefly discuss our strategy when looking for a term's hidden t.p.d.f. Because each particular term is rare, a term's t.o.p.d. may be very noisy, varying greatly from its underlying hidden t.p.d.f. We would like to be able to draw generalizations and average out the noise in the data. Therefore, throughout the course of our analysis, our goal will be to collect many terms with what we hope are the same underlying t.p.d.f.s to understand better what parameter describes it. The average of many empirical t.o.p.d.s that have been created by the same t.p.d.f. should approach the actual function.

Of course, we do not know the underlying t.p.d.f., so we cannot actually bucket terms by it. Instead, we will attempt to approximate this hidden t.p.d.f. through combinations of different statistics available to us from the corpus. Therefore, we will bin the terms by combinations of the different statistics we will describe in Section 4.3. We do not, however, want our binning to hide the information contained by the hidden t.p.d.f. for each term because we have binned together many terms with different underlying probability density functions. This will be the challenge throughout our investigation.

4.2.1 Finding the hidden t.p.d.f.

Let us discuss how we bin by a certain statistic in the corpus in our attempt to find the hidden t.p.d.f. with an example. We assume that in addition to the hidden t.p.d.f., the observable t.o.p.d. for each term is affected by the length of the documents in which the term appears. For this reason we begin our investigation by removing length as a factor. Here we will describe the procedure we use to gather all of the terms that appear in documents of similar lengths by binning documents of the same length. Although we describe the procedure with respect to this specific example, we will follow the same procedure each time we want to look at terms that are similar with respect to a certain statistic.

Binning by document length

Although it is irrelevant in the multinomial model, document length is often difficult to account for in probabilistic models. Because of this, many models assume constant length documents. We choose to remove document length as a statistic throughout the majority of our statistical analysis, allowing us, for the purpose of our analysis, to work with a constant length assumption. We will visit the effect of document length on the term distribution when it becomes necessary, in Section 5.1.3.

Because we want to be able to have enough data to draw valid conclusions, we would like the largest group of documents of the same length as possible. In order to do this, we look at the distribution of document length throughout the corpus. In Figure 4-2, you can see the number of documents at different lengths for each document sets that comprise our corpus (such as Wall Street Journal articles, or

Associated Press articles), as well as for the corpus as a whole.

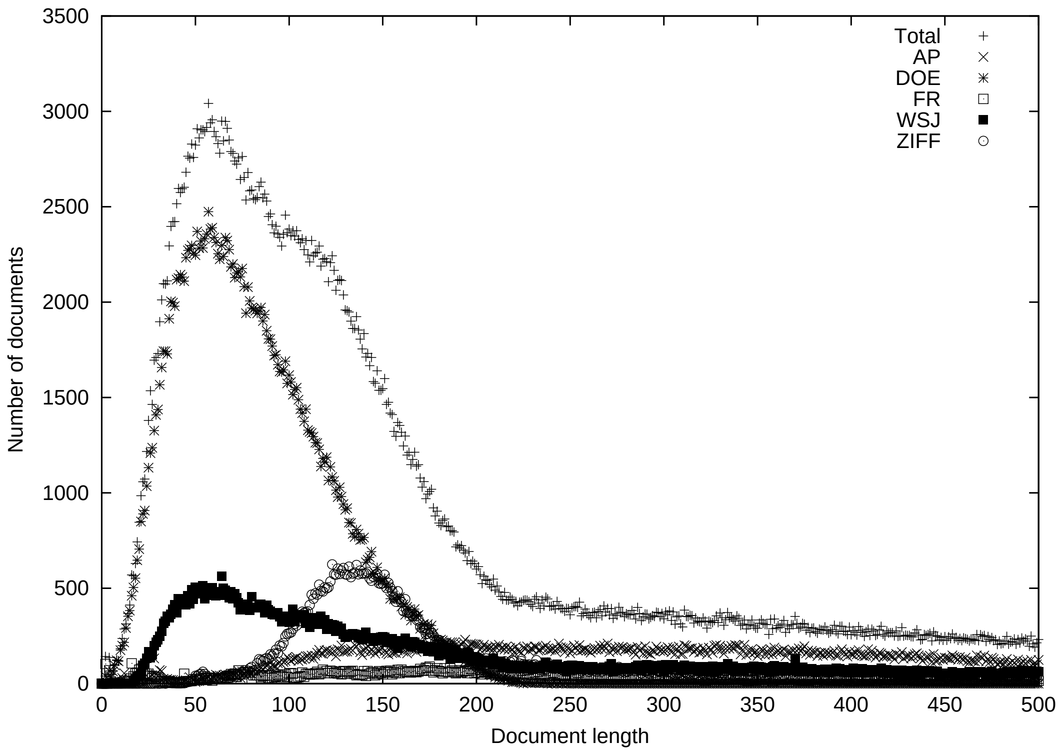


Figure 4-2: The number of documents of length n. Document length displayed by source, and for the corpus as a whole.

We are interested in looking at documents of any given length. However, there just aren't that many documents that are the same length. As you can see, even if we were to take the most popular document length, we would still only have around 3,000 documents to work with. This is not very many if you consider we begin with over 500,000.

To gather more documents of a certain length, rather than choosing documents of just one length we use documents that are "similar" in length. By this we mean that rather than requiring our documents to be of length exactly 100, we will allow the documents to perhaps have a length of 99, or perhaps have a length of 101. We want to create bins of documents of similar lengths, and then choose the largest bin. When binning, we use exponentially increasing the boundaries of the bins so that the longest document in a bin is a most 110% the length of the shortest document in the bin. In Figure 4-3 you can how many documents fall into each of the bins of similar document length. Note that because the bins are growing exponentially, the x-axis, which represents the average length of the documents in each bin, is on a log scale. When we use documents that are similar in length rather than exactly equal, we get many more documents in our bins.

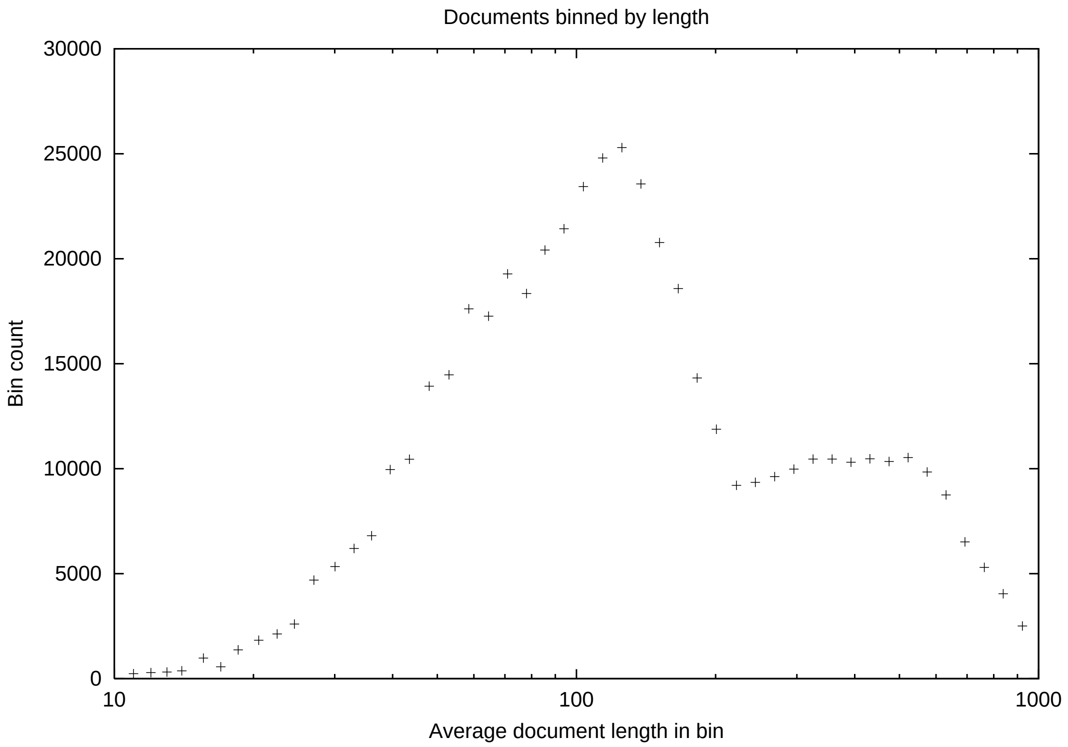


Figure 4-3: The number of documents within a bin. Each bin represents a document length range. The length range of a bin grows exponentially to keep the relative variation within a bin similar.

There are two humps in bin size that can be seen in Figure 4-3. We take the bins that represent the top of both humps. Table 4.1 shows some details of the two document length bins we selected. Note that while Data Set 1 has many more documents than Data Set 2, the number of term occurrences in Data Set 2 is greater. From now on, when we refer to Data Set 1 and 2, we will be referring to these two subsets of the corpus.

	Data Set 1	Data Set 2
Number of documents (n):	25,295	10,530
Average document length (ℓ):	125.4	431.5
Number of unique terms:	78,462	88,692

Table 4.1: Description of the two sub-corpora we will analyze. Each sub-corpus contains documents of the same length.

4.3 Terminology

It will be useful to define some terminology for our data analysis. We will be discussing the effect of different statistics on the term occurrence distribution, and in order to avoid having to define them repeatedly, we will define them now. We will discuss how we arrived at some of these statistics later on, in Section 4.5.1. Because sometimes it is difficult to understand exactly what certain statistics mean, we will give examples with respect to our "Granny Smith apples" corpus.

Corpus frequency The number of times that a term occurs throughout the entire corpus. This includes repeated occurrences of a term within the same document. In our example corpus, the corpus frequency of the word "apple" is 12, while the corpus frequency of the word "red" is 1. In equations, and occasionally for brevity, we will refer to the corpus frequency of term i as cf_i. $cf_i = \sum_{j=1}^{n} d_i^j$

Empirical term probability This is the empirical probability that, if you randomly selected a term from the entire corpus, that you would select a particular term. You will recall that the true value of the term probability is important for the multinomial model we discussed in the previous chapter. The empirical term probability is what we used in the multinomial model to try to estimate θ. It is equal to the corpus frequency divided by the total number of terms in the corpus ($\frac{cf_i}{\sum_{i=1}^{m} cf_i}$). The probability of the term "apple" appearing in our example corpus is $\frac{12}{43} = 0.28$, and the probability of the term "red" appearing in our example corpus is $\frac{1}{43} = 0.02$. We will occasionally refer to a term's empirical probability as p_i. Note that this value is not the same as the θ_i we discuss in Chapter 3 because p_i is the observable value, where as θ_i represents the actual value used to generate the documents.

Document frequency The number of documents that a term occurs in. In this value we do not include repeated occurrences of a term within a document. In our example corpus, the document frequency of the word "apple" is 5, while the document frequency of the word "red" is 1. We will sometimes refer to document frequency as df_i. If we assume that $[statement]$ evaluates to 1 if the statement is not equal to zero and 0 if the statement is equal to zero, then $df_i = \sum_{j=1}^{n}[d_i^j > 0]$

Conditional average When we use the words "conditional average" when referring to a term, we mean the average number of times that a term appears in a document, given that the term appears in the document. The average number of times a term occurs if we do not precondition on its occurrence is proportional to the corpus frequency $(\frac{cf_i}{n})$. When we do condition on appearance, this value is equal to a term's corpus frequency divided by its document frequency $(\frac{cf_i}{df_i})$. "apple" appears in every document in our example corpus, for a total of 12 occurrences. This means that while the average number of times it appears in the corpus is $\frac{12}{43} = 0.28$, the average number of times "apple" occurs given that it has occurred is $\frac{12}{5} = 2.4$. On the other hand, "granny" appears three times in only two documents, so by our definition of average, the average occurrence of "granny" is $\frac{3}{2} = 1.5$. If we refer to the average value of a term conditioned on occurrence in an equation, we will refer to it as avg_i.

We will also be interested in the effect that document length has on these statistics. Sometimes we will work with only documents of the same length in order to understand the other factors without the influence of length. Other times we will explicitly include length to understand its effect on term occurrence. There are several different things we could be referring to when talking about document length, and we will define them here:

Length Here we refer to what is typically thought of to when discussing document length. This is the number of terms that occur in the document, including repeated occurrences of the same term. The length of document 4 in our example corpus is 8. $\ell_j = \sum_{i=1}^{m} d_i^j$.

Unique words This refers to the number of unique words in a document, and will be occasionally expressed as u_j. Repeated occurrences of terms within a document are ignored. The number of unique words in document 4 is 5. If we again assume that [*statement*] evaluates to 1 if the statement is true and 0 if the statement is false, then $u_j = \sum_{i=1}^{m} [d_i^j > 0]$.

Another statistic that we will sometimes mention is the number of documents in the corpus. We define this as n in Section 3.1.2 and will continue to refer to it as such. You should note that here this value is dynamic because we occasionally work with subsets of the larger corpus we are using for data analysis. For example, we might want to only work with documents of more or less the same length in our example corpus. Say we decide we are only interested in documents of length 7 or 8 (documents 1, 2, 3 and 4). This means we would be working with $n = 4$ rather than $n = 5$.

4.4 Data Not Multinomial

It does not take a very much exploration into the data before we see that the distribution of terms throughout documents does not seem to have come from a fixed multinomial. In this section we will discuss a couple of the ways that the data shows us this. One is that the number of unique terms that occur in a document of a certain length is much smaller than would be expected given the multinomial model. We also find that the t.o.p.d. has a much heavier tail than the multinomial model predicts.

The evidence we see in the text documents suggests that people reuse words that they have already used in a document. On the other hand, the multinomial model we describe says that a term has the same probability of appearing in a document after we have used the term a number of times as it does in a document in which it hasn't yet appeared. For example, once we use the words "Granny Smith apples" at the beginning of this document, it is likely that we use them again. And indeed, in this document, we often do. If we hadn't first mentioned Granny Smith apples before, though, you would be less likely to expect to see the words here. On the other hand, the multinomial model we describe would say that the "Granny Smith apples" is only as likely to appear again as it is to appear in the next document you read. While this term behavior could perhaps be explained by a mixture of multinomial distributions, such an explanation would require us to break away from the extremely simple model with which we have been working and with which we want to continue to work.

4.4.1 Unique words

Let's look at the number of unique words in a document (u_j) compared with the length of a document (ℓ_j). Given the multinomial model, the expected number of unique words is the sum of the probability of seeing each term at least once in a document of that length. Mathematically, this can be expressed as:

$$E[u_j|\ell_j] = \sum_{i=1}^{m} 1 - (1 - \theta_i)^{\ell}$$

We can approximate this by using the p_is for each term in our corpus instead of the θ_is. In Figure 4-4 you can see what we would expect from the multinomial model for the number of unique terms compared with document length. This is compared with what we actually see in our corpus. Note that since there is a lot of variation in the actual data, with a wide variety of unique terms to be found in documents of the same length, we have binned the data to make the overall pattern more obvious. The standard deviation of each bin is plotted to give you an idea of the variation within the bin.

4.4.2 Heavy tailed

The reason we postulate there are many fewer unique terms in a document of a given length is, as we mentioned earlier, once a term is used in a document, it is likely

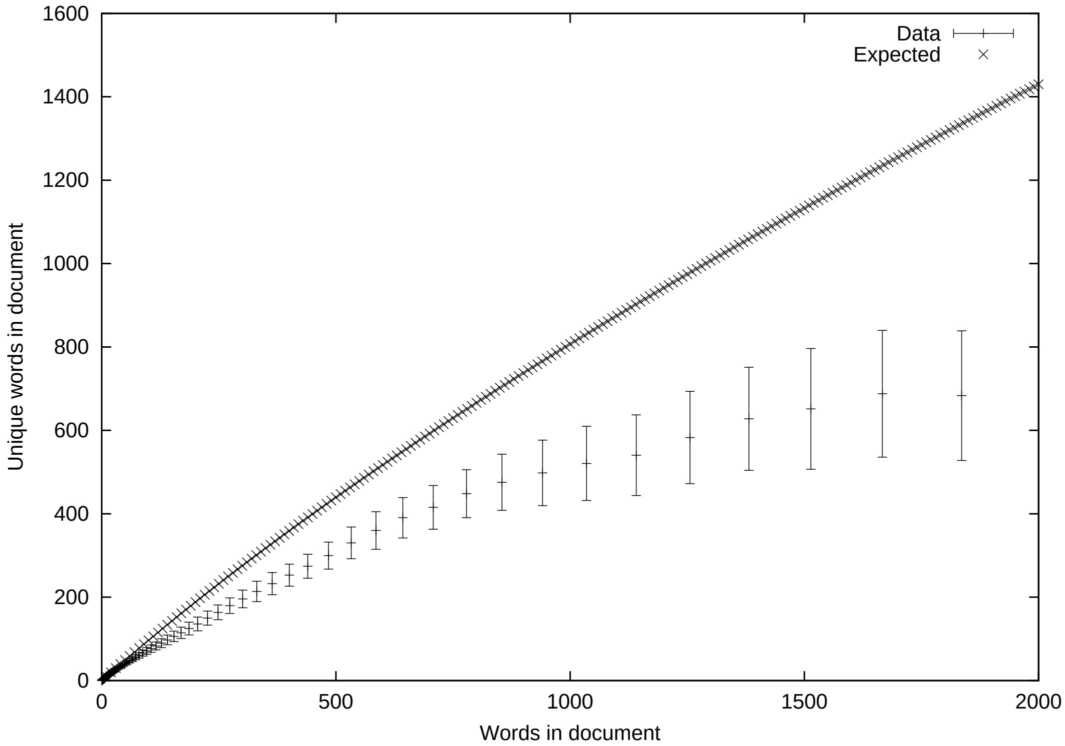


Figure 4-4: Number of unique terms in a document compared with the number of terms in a document. Graph displays empirical values and what is predicted given the multinomial model. For clarity, the empirical data has been binned. Error bars on the empirical data represents the standard deviation within each bin.

to be used again. One way to confirm this is to look at the distribution of term occurrences throughout the corpus. If, once a term has occurred in a document, it is much more likely to occur again, we should see, for example, the probability of that term occurring two times in a document is much higher than the multinomial model expects.

We find that the term occurrence probability distribution actually does have a much heavier tail than is predicted by the multinomial model. First, let us look at what we expect from the model. Because it is multinomial, the probability that a term occurs exactly ω number of times in a document is a function of the probability of that term occurring at all in the corpus (p_i). It can be expressed as follows:

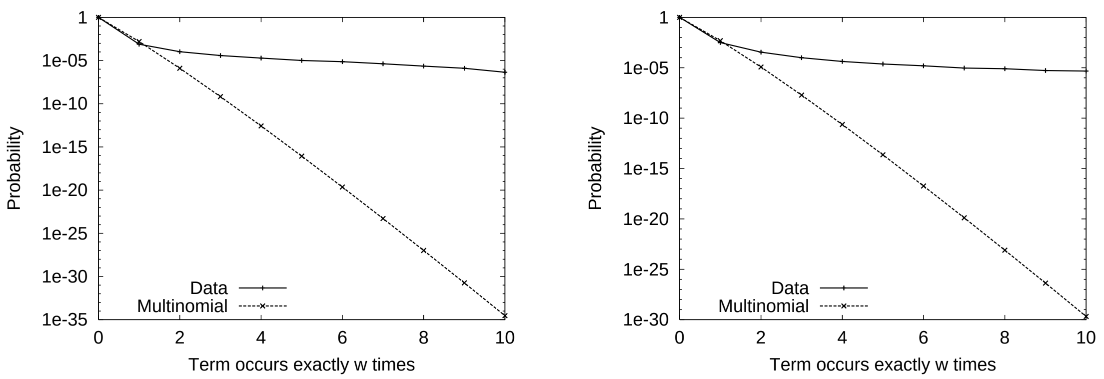
$$\begin{aligned}\Pr(\omega) &= \binom{\ell}{\omega} p_i^{\omega}(1-p_i)^{(\ell-\omega)} \\ &\approx \frac{(\ell p_i)^{\omega}}{\omega!}\end{aligned}$$

To understand what sort of distribution we can expect from this equation, let us consider a very similar distribution, a binomial distribution. For our binomial model we will be flipping a coin. If you flip a coin once, you'd expect to get head with a probability of $\frac{1}{2}$. If you flip a coin twice, you'd expect to get heads both times with a probability of $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. When you flip the coin a third time, you'd expect to get heads with a probability $\frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$. You can see that this probability of your observing all heads goes down exponentially with the number of times you flip the coin. Similarly, given the multinomial model, we will expect each term's t.o.p.d. to be roughly exponential.

We would like to compare what we expect the term occurrence probability distribution to be with what we observe. Given the multinomial model, if we know the probability of a term occurring, we also know the term's t.o.p.d. To see if the data matches what is expected from the model, we can compare, for a given p_i, the t.p.d.f. assumed by our multinomial model with the empirical t.o.p.d. we find in the data for terms with that p_i.

Again, rather than looking at the empirical t.o.p.d. for just one term, we will bin terms with a similar empirical probability of occurring in hopes of averaging out the noise present in the observation of each term. In order to work with a large set of terms, let us look at all terms that have more or less the same empirical probability of occurring in the corpus as the average term in the corpus. For Data Set 1 the average term has a probability of $\frac{1}{78462} = 0.0000127$ of occurring, and in Data Set 2 a term has, on average, a probability of $\frac{1}{88692} = 0.0000113$ of occurring.

Taking all terms that have similar empirical probabilities of occurring to the values discussed above, we are able to find the average t.o.p.d. for these terms. You can see in Figure 4-5 the actual term distribution for these terms, compared what the multinomial model would predict the term distribution would be. The average term has a probability of occurring zero times that is almost one, since most documents only contain a very small number of the possible terms they could contain. The probability of occurring exactly one or more times is almost zero for most terms. For this reason, the y-axis is shown in a log scale so that we can see greater detail. Since the expected distribution is nearly exponential, it appears as a straight line. You can see how much heavier the tail is than expected with the multinomial model, and that the probability of a term occurring alone is lower than would be expected.



(a) Data Set 1 (b) Data Set 2

Figure 4-5: The empirical t.o.p.d. for terms compared with what is expected from the multinomial model. Terms occurrence appears to be more clustered than expected.

4.5 Exploring the Data

We have seen that what we would expect given the multinomial model doesn't match the statistical properties data. In this section, now, we explore the data in order to understand the statistical properties of the text better so that we can build a model that better matches them. We will discuss what we have found in our data exploration with the constraints we impose in Section 4.2, focusing specifically on the correlation between different statistics and the term occurrence distribution for terms that are similar with respect to one or more statistic. In both of these explorations, we look at the statics for documents of a uniform length, working only with Data Set 1 and 2.

4.5.1 Possible statistics that describe the hidden t.p.d.f.

We will investigate combinations of the statistics we discuss in Section 4.3, in hopes of understanding what one parameter best describes the term occurrence distribution, and what sort of behavior they exhibit. In this way we hope to find the hidden t.p.d.f. that describes a term distribution. In this section we will discuss how we selected the statistics we choose to look at.

Several of the statistics we describe we select because of their presence in information retrieval literature. Specifically, we investigate document frequency because it is a very common statistic used in vector space models. We choose to investigate a term's corpus frequency because, as we have mentioned, it is related to a term's probability of occurrence, and therefore to the term's behavior in the multinomial model.

We also try several experimental approaches to arrive at the hidden parameter. In this section we will first talk about how we try to use unsupervised learning to find the hidden parameter for us. Then we will talk about the correlations between different statistics with the term occurrence values to understand our statistics.

Clustering

Initially we would like to investigate what commonalities there are among terms by grouping them without forcing onto the groupings any of our preconceived notions about what we expect to see. To do this, we use a clustering algorithm in hopes of creating clusters of terms that are similar with respect to the hidden parameter we are trying to find. We know that the hidden t.p.d.f. must imply the term's distribution. For this reason we try to cluster terms with similar empirical probability distributions. Here we will first discuss how we perform the clustering, and then we will discuss what we learn from it.

How we cluster We use a k-means clustering algorithm to cluster the terms in our corpus. k-means is a very simple greedy clustering algorithm. The way it works is to chose k points to act as cluster centers. Each data point is then clustered around the closest center. Using the points in the cluster, a new center is computer, and the process is repeated until no points change clusters any more.

k-means is similar to the machine learning technique "expectation maximization" that we use for learning the probability of a term occurring in a relevant document in the multinomial model. However, what it really is "maximization maximization", a lower bounds on EM. For each term, k-means finds the center that is most similar to that term. If this were an "expectation" step, we would find the probability of it belonging to each cluster. Instead, we maximize the term's location by placing it in the most likely cluster. Then, given a cluster and all of the terms it contains, k-means finds the center that maximizes the likelihood of that cluster.

Because we believe most of the information is contained in the first few occurrences, we truncate the t.o.p.d. and use only the number of times the term occurs zero times in a document, the number of times the term occurs one time, two times, three times, and four or more times. For the purpose of clustering, then, we represent each term as a vector of hs, where h_0 is the number of documents in which the term occurs exactly 0 times, h_1 is the number of documents in which the term occurs exactly once, and so on, up to h_4, which represents the number of times a term occurs four or more times in a document.

Initially, k-means is started with random cluster centers. These centers are created by randomly selecting a term from the data, and then modifying its value slightly by adding or subtracting small random values. During each iteration, we recompute the cluster center. We choose the center as the distribution that maximizes the likelihood of the points in the cluster by averaging the t.o.p.d. for each term in the cluster.

Note that it is often the case that after iterating a cluster center is left with no terms in the cluster. When this happens we replace the center with a randomly selected term. By choosing an actual term we are guaranteed that at least one term, namely the term that is the center, will fall into that cluster on the next iteration.

Distributing terms into clusters involves finding the "closest" center to each term. To do this we must find the distance of a term from each cluster center. The distance function we use in clustering isn't a standard Euclidean distance, but rather the probability that the point was generated by each center. If we have a cluster center $\mathbf{g}$, where g_i represents the number of times a term at the center occurs exactly i times, and g is the sum of all g_i, then the expected probability of a term generated by the center occurring exactly i times is $\frac{g_i}{g}$. If we look at a particular term's distribution, which, as we mentioned earlier is expressed as $\mathbf{h}$, we can find the likelihood that the distribution was generated by the center in the following way:

$$\begin{aligned} \text{distance} &= \binom{h}{h_0, h_1, h_2, h_3, h_4} \prod_{i=0}^{4} \left(\frac{g_i}{g}\right)^{h_i} \\ &\propto \prod_{i=0}^{4} \left(\frac{g_i}{g}\right)^{h_i} \end{aligned}$$

This value is monotonically increasing with its log, so we will use $\sum_{i=0}^{4} h_i \log \frac{g_i+1}{g}$ as our distance function. Note that we add to 1 to g_i so that even if there is no term in the cluster that occurs exactly i times in a document, a new term that does could possibly be added to the cluster.

What we learn We are unfortunately unable to learn much from clustering. Regardless of how many centers we start with, most of the terms end up falling into one cluster. We can try to get rid of outliers by taking that one cluster, and then trying to sub-clustering it. But still we find the same behavior. This implies that most terms are very similar to each other. Not surprisingly, the large cluster that we find has a term distribution that looks very much like term distribution for the average term that we see in Figure 4-5.

Because the k-means algorithm starts with random cluster centers, the results of the clusterings do vary somewhat. Occasionally the clustering algorithm finds a second largest cluster that contains a number of terms, enough terms not to be considered an outlier group, although it does contain significantly fewer terms than the large cluster. This small cluster has a term distribution where the term is more likely to occur exactly two times than the large cluster, but slightly less likely to occur any other number of times. We will see a similar dip and then rise in the term probability distribution when we investigate the effect of average on the term occurrence distribution (Section 4.5.2).

Correlation

Another way that we look for the hidden parameter is by looking at the ways in which the term occurrence distributions are correlated with the statistics we are interested in, as well as their correlations with themselves. By understanding a little bit about which values available to us are related to each other, we can understand what information about a term's occurrence is not being represented in the statistics we choose to investigate.

To show relationships between two statistics we use the correlation coefficient for those two statistics. The correlation coefficient is equal to the covariance of the two statistics normalized so that the value is 1 if the two statistics are perfectly correlated and -1 if the two statistics are perfectly not correlated. This value gives us an idea as to whether the statistics are linearly related.

Table 4.2 shows the correlation coefficients the relationships for some of the statistics we have discussed with the term occurrence distribution. Notice that the corpus frequency (cf) and document frequency (df) are, not surprisingly, both highly correlated with the number of times a term occurs exactly ω times. However, it is surprising how uncorrelated they are with correlated with the probability of the term occurring any number of times given that it has occurred. That is to say, based on the correlation table, both document frequency and corpus frequency seem to give us a good idea of how likely the term is to appear at all, but no idea of how many times the term will occur once it has.

It was for this reason that we introduced the conditional average into our investigations. The average number of times that a term occurs is, while hardly correlated at all with the number of occurrences of a term, highly correlated with the probability that the term will occur given that the term occurs at all, informing us about the term in a way that cf and df do not.

While it is not surprising that corpus frequency and document frequency are related to the number of occurrences and not the probability of occurrence given that the term has occurred, and vice versa for average, it is slightly surprising how uncorrelated each value is with what it does not explain. That is to say corpus frequency and document frequency seem to say surprisingly little about the term's probability distribution given that the term has occurred, and average seems to says surprisingly little about the overall number of occurrences of the term. This may suggest that two parameters, and not one, are actually necessary.

Inter-term occurrence correlation Also included in the correlation table is the correlation of the number of times a term occurs exactly once with all other values, as well as the correlation of the probability that a term occurs exactly once, given it has occurred, with the other statistics. Given how highly correlated values such as the corpus frequency, document frequency, and average are with various aspects of the term occurrence distribution, it is surprising to not see as strong a correlation with these two values.

For example, while corpus frequency and document frequency are highly correlated

	cf	df	avg	Ex 1	Pr(ex 1|occ)
cf	1.00	0.92	0.02	0.77	-0.02
df	0.92	1.00	-0.01	0.96	0.00
avg	0.02	-0.01	1.00	0.02	-0.77
Exactly 1	0.77	0.96	-0.02	1.00	0.02
Pr(exactly 1|occurs)	-0.02	0.00	-0.78	0.02	1.00
Exactly 2	0.84	0.90	0.01	0.81	-0.3
At least 2	0.97	0.83	0.03	0.63	-0.04
Pr(exactly 2|occurs)	0.01	0.01	0.18	0.00	-0.67
Pr(at least 2|occurs)	0.02	0.00	0.78	-0.02	-1.00
Exactly 3	0.87	0.78	0.03	0.61	-0.04
At least 3	0.88	0.62	0.04	0.37	-0.03
Pr(exactly 3|occurs)	0.01	0.00	0.31	-0.01	-0.43
Pr(at least 3|occurs)	0.02	-0.01	0.86	-0.02	-0.67

(a) Data set 1

	cf	df	avg	Ex 1	Pr(ex 1|occ)
cf	1.00	0.87	0.04	0.68	-0.05
df	0.87	1.00	0.01	0.94	-0.04
avg	0.04	0.01	1.00	-0.01	-0.62
Exactly 1	0.68	0.94	-0.01	1.00	-0.02
Pr(exactly 1|occurs)	-0.05	-0.04	-0.62	-0.02	1.00
Exactly 2	0.80	0.93	0.02	0.81	-0.06
At least 2	0.94	0.90	0.03	0.69	-0.06
Pr(exactly 2|occurs)	0.03	0.03	0.10	0.02	-0.69
Pr(at least 2|occurs)	0.05	0.04	0.62	0.02	-1.00
Exactly 3	0.85	0.87	0.03	0.66	-0.05
At least 3	0.94	0.78	0.04	0.52	-0.06
Pr(exactly 3|occurs)	0.02	0.02	0.17	0.01	-0.40
Pr(at least 3|occurs)	0.05	0.02	0.74	0.00	-0.68

(b) Data set 2

Table 4.2: Correlation coefficients for various different statistics available during our textual analysis. When the correlation coefficient is close to one the statistics are highly correlated. A correlation coefficient that is close to negative one means that the statistics are negatively correlated, and when it is close to 0 the statistics are not correlated at all.

with the number of times a term occurs at least two times (ranging from a correlation of 0.83 and upwards), the number of times a term occurs exactly one time has a much lower correlation coefficient (around 0.65 in both data sets). Also worth noting is that the probability that a term occurs exactly one time (given that the term occurs) is negatively correlated with the probability that a term occurs two or three times, given it has occurred. This is unsurprising because the the probability of occurring zero times is so high as to be basically constant across terms, so when a term has a higher probability of occurring exactly once, it is much less likely to occur some other number of times.

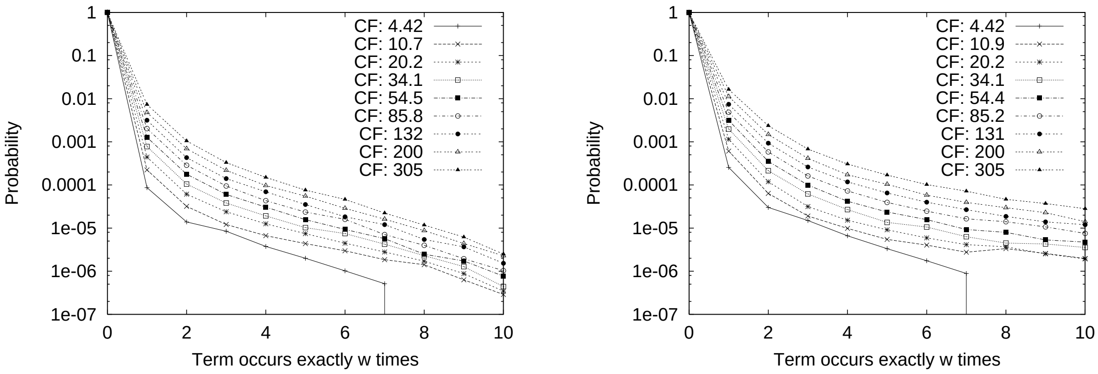
Relationship between corpus frequency and document frequency As can be seen in the correlation table, corpus frequency and document frequency are almost linearly correlated. Document frequency is the more common metric showing up in information retrieval, because it is used in vector space retrieval. However, since our basic multinomial probabilistic model is based on a term's probability of occurring, θ_i, which is directly related to the term's corpus frequency, we begin our investigations with this value instead instead. After we understand somewhat the effect of corpus frequency, we move on to use the more common value, document frequency, instead. The linear relationship between the two will ideally means that the patterns we find by investigating will also appear when investigating the other.

4.5.2 Finding the hidden t.p.d.f.

In this section we will discuss what we found in our search to understand the hidden t.p.d.f. using the possibilities described in Section 4.3. We will specifically talk about how we tried to find terms that are similar with respect to their hidden parameter by binning terms with similar corpus frequency, document frequency, and conditional average.

Corpus frequency

While document frequency is the more common statistic found in information retrieval literature, we first look at corpus frequency because that is the statistic that relates most directly to our multinomial model (since $\theta_i = \frac{cf_i}{\sum_{j=1}^{n} \ell_j}$). For this reason, we begin our analysis of the t.o.p.d. by looking at the effect that corpus frequency has on it. To do this, within Data Set 1 and 2 we bin the terms by corpus frequency. Figure 4-6 shows the probability occurrence distribution for each corpus frequency bin. Because a majority of the time the term occurs exactly 0 times in a document, we show the graphs with the y-axis on a log scale.

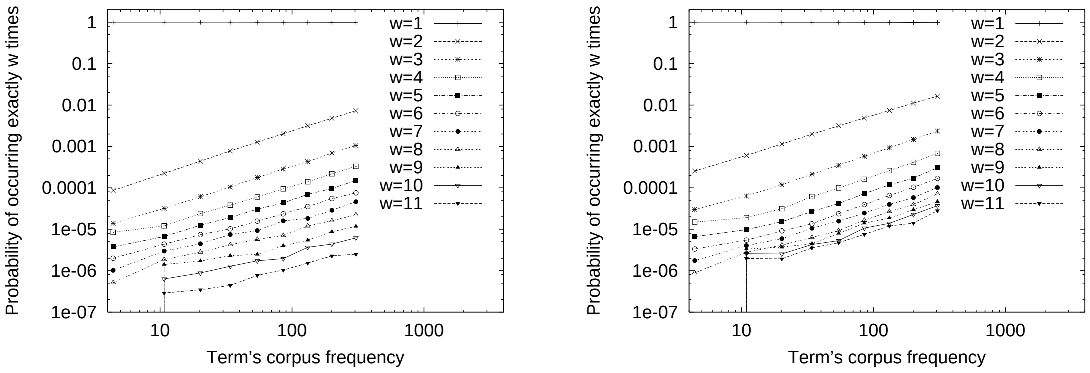


(a) Data set 1 (b) Data set 2

Figure 4-6: The t.o.p.d. for terms with different corpus frequencies.

An interesting thing to note about these graphs is that the lines for terms of with different corpus frequencies appear to be evenly spaced. This leads to questions what information is contained in knowing multiple occurrences of a term. It is interesting that we see that the number of times a term occurs in a document might not contain additional information, since using a term's frequency of occurrence within a document has become a common practice in IR often leads to retrieval performance [26]. We investigate this further in Figure 4-7. Each line in these graphs follows the probability that a term will occur a certain number of times as a function of the term's corpus frequency. We are essentially looking at a vertical slice of Figure 4-6.

For example, the straight line at the top of the graph shows us what the average



(a) Data set 1

(b) Data set 2

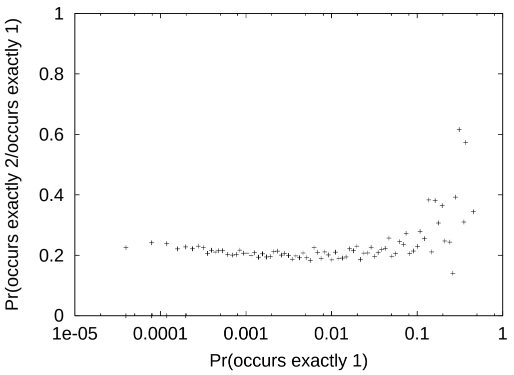
Figure 4-7: The probability that a term will occur exactly ω times as a function of the term's corpus frequency.

probability of seeing a term zero times for different corpus frequencies. At low corpus frequencies this probability is almost one, and at high corpus frequencies, the probability is also one. Of course, it is not actually straight, but the minor variations are hidden by the log scale. That it appears so straight implies that corpus frequency does not have a relatively significant affect on a term's probability of occurring zero times.

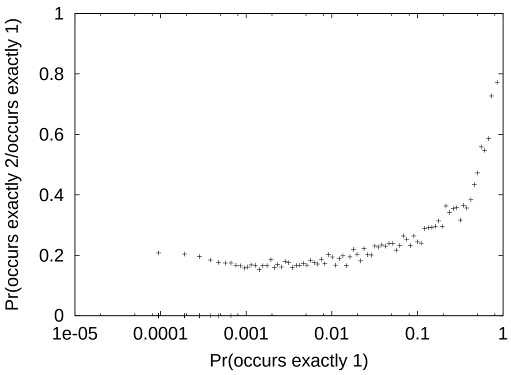
On the other hand, the slanted line labeled "$\omega = 1$" shows us what the expected probability of seeing a term one time is. You can see that as a term's corpus frequency increases, so does the probability of seeing a term exactly once in a document. This appears to be true for all values where $\omega \neq 0$.

That these lines are almost parallel implies that for terms binned by corpus frequency, there is no new information in the number of times a term occurs once you know that it has occurred. To investigate this further we graphed the probability that a term will occur at least once compared to the probability that a term will occur at least twice divided by the probability that the term will occur once. If there is no new information in the probability that a term occurs at least twice, the graph should be straight.

In Figure 4-8 you can see what we found. For Data Set 1 it is mostly straight in the region where a majority of terms occur. The values flare up for high probabilities of occurring exactly once, but this flare up represents a very small number of terms. On the other hand, the flat region for Data Set 2 is slightly shorter, and, in fact, it is arguable as to whether it really should be considered flat at all.



(a) Data Set 1



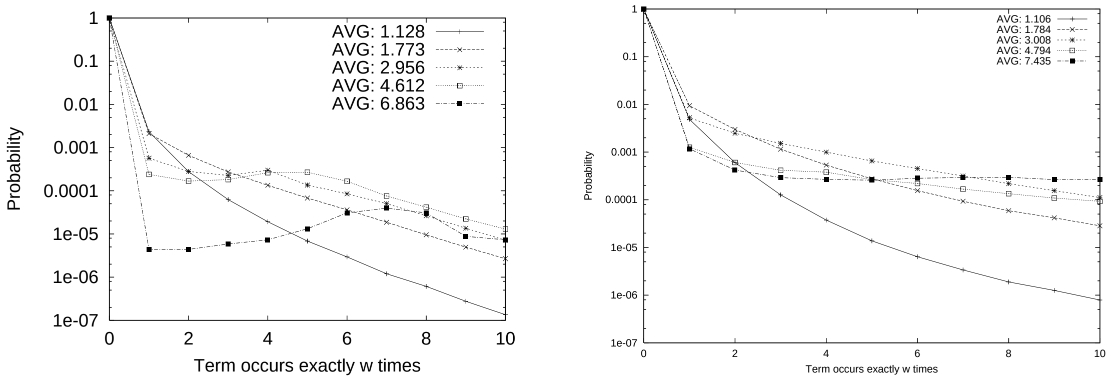
(b) Data Set 2

Figure 4-8: Probability of a term occurring once compared with the probability of a term occurring two times.

Looking at the terms binned by the conditional average does not seem to give a clear pattern for term behavior. This can be seen in Figure 4-9. Terms with higher average have a heavier tail, as is expected given a that a term that has a higher conditional average occurs more often on average in documents it occurs in. But the lack of a consistent pattern implies that the *avg* is not a good indicator of the overall t.o.p.d.

Probability given occurrence

If it is true that term occurrence once a term has appeared in a document behaves in a manner independent of the term's overall corpus frequency, as seems to be implied by Figure 4-8, then one thing we can look at is the term's probability distribution given the term has occurred, and try to differentiate a term given that we know it has occurred. Looking at the conditional t.o.p.d. is motivated by the possible existence of two hidden parameters rather than one. One of the hidden parameters tells us if the term occurs or not, and the other tells us something about the probability



(a) Data Set 1

(b) Data Set 2

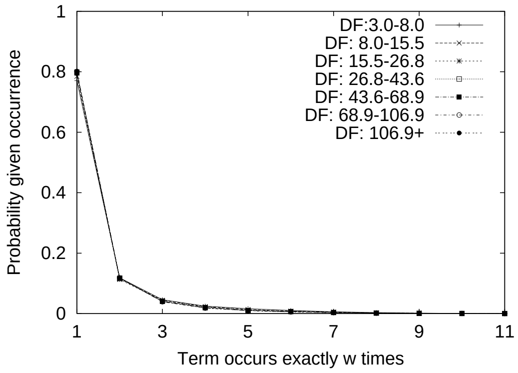
Figure 4-9: The t.o.p.d. for terms with different conditional averages. There does not seem to be a very consistent pattern for term occurrence across different conditional averages.

distribution once it has occurred.

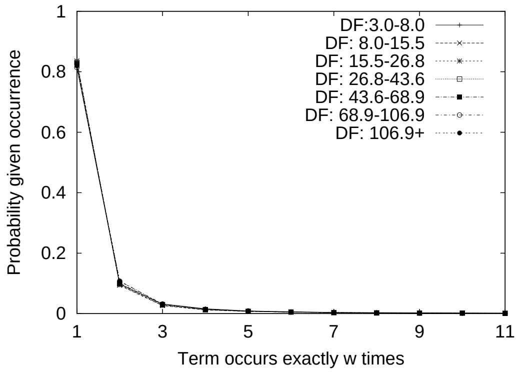
If a statistic does not have an effect on the conditional t.p.d.f., as we suspect could be the case for corpus frequency and document frequency, then when we bin by that statistic and plot the probability distribution, the lines for each bin should be identical. On the other hand, if the value is indicative the probability of a term occurring a certain number of times given it has occurred, then we should see different behavior depending on the value of the statistic.

Here we show that, if we condition on the term occurring, the document frequency does indeed create overlapping probability distributions given occurrence. This can be seen clearly by looking Figure 4-10. Each bin contains terms with the same empirical probability of occurring, and yet there is hardly any noticeable difference between the probability that a term will occur exactly ω times. The variation only appears when we plot the distribution on a log scale, and even there it is tiny.

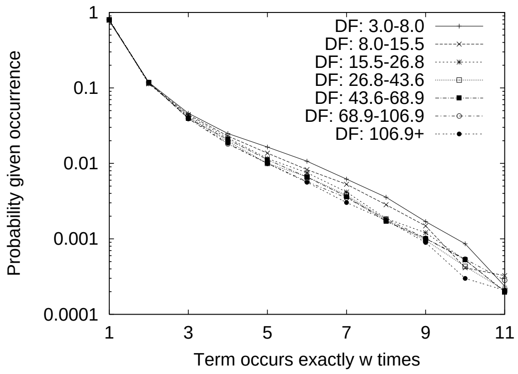
On the other hand, investigation into the effect of the conditional average number of times a term occurs given that it has occurred on the distribution shows that it is a much more informative value. Of course, this is not surprising, since it is the empirical statistic corresponding to what is left over after removing the probability that the term occurs. We saw in Table 4.2 that conditional average is much more highly



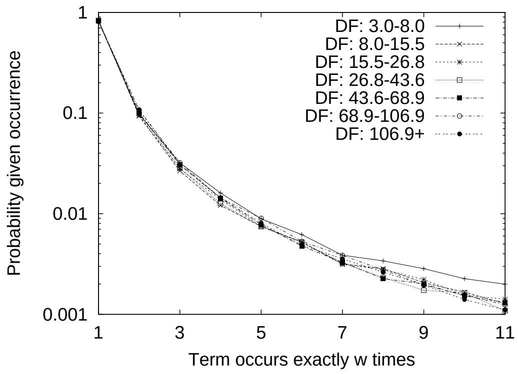
(a) Data set 1



(b) Data set 2



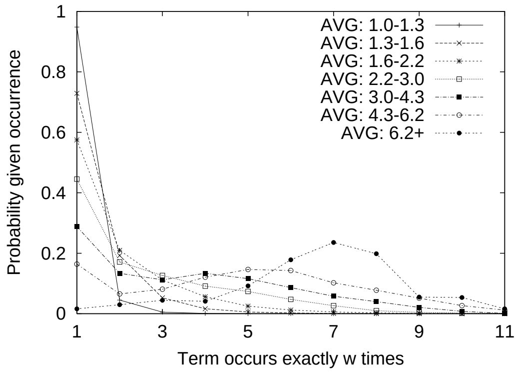
(c) Data Set 1 – Log scale



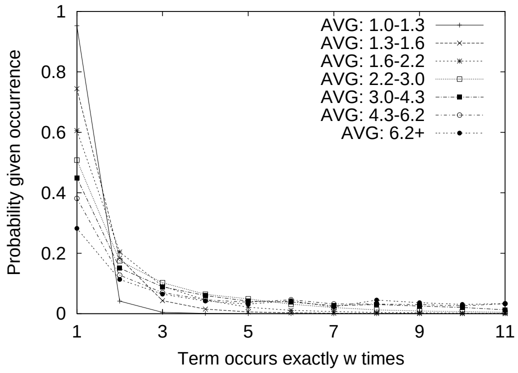
(d) Data Set 2 – Log scale

Figure 4-10: The t.o.p.d. given that the term has occurred for terms with different document frequencies. The variation is small across different document frequencies, implying that document frequency is not a good measure of the probability a term will occur given that the term has occurred.

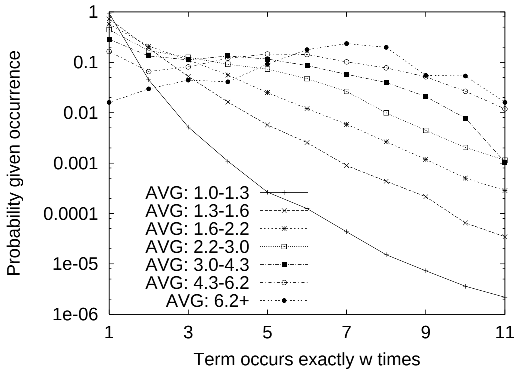
correlated with the probability of a term occurring a certain number of times given it occurs, and Figure 4-11 demonstrates this further. Binning the term occurrence probability distribution given that the term has occurred by the term's conditional average occurrence gives us groups of terms with very different distributions.



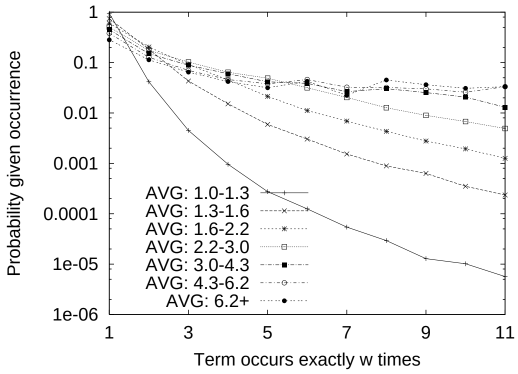
(a) Data Set 1



(b) Data Set 2



(c) Data Set 1 – Log scale



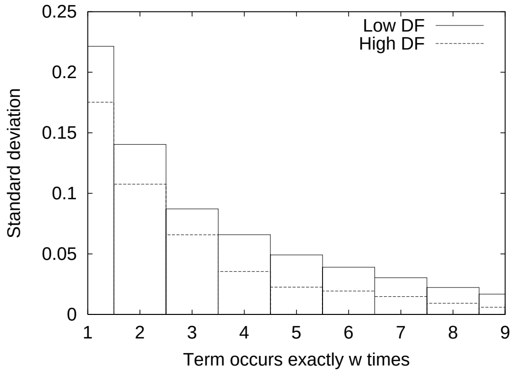
(d) Data Set 2 – Log scale

Figure 4-11: The t.o.p.d. given the term has occurred for terms with different conditional averages. Binning by conditional average produces much more variation than binning by document frequency.

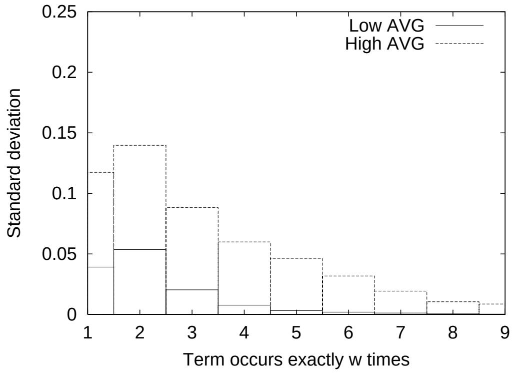
Looking at the standard deviation of the t.o.p.d. between terms in a bin gives us an idea of how much variation there actually is in each bin. If there is a lot of variation within a bin, then the bin is not doing a good job of summarizing the terms it contains.

We find that there is less deviation in the bins that are created by binning terms by avg_i than the bins created by binning by df_i. It is interesting to note there is a change in the standard deviation within a bin as the statistic that the bin represents increases in magnitude. When terms are binned by document frequency, as the document frequency increases, the standard deviation in the bins decreases. On the other hand, when the terms are binned by conditional average, as the conditional average increases, the standard deviation in the bins increases. Since most terms typically have relatively low document frequency and conditional average values, the small valued bins are the largest. This means that the value df_i, which has a particularly large standard deviation for low document frequency bins, is a lousy indicator of the term's conditional t.o.p.d. On the other hand, the value avg_i, which does well for low conditional average bins, is a better one.

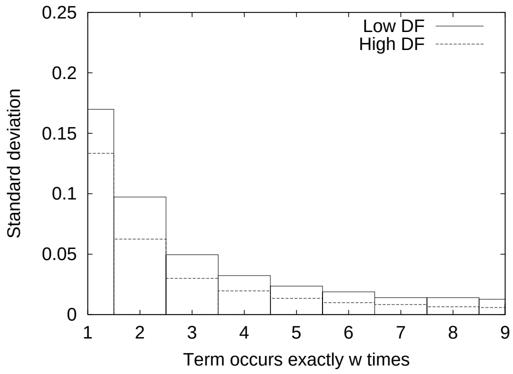
We have chosen several representative document frequency and average bins to illustrate this point, and they can be seen in Figure 4-12. For each bin, and for each integer ω, we measured the standard deviation over terms in the bin of the number of documents where that term appeared ω times. The dark line in the graphs represents a large bin where the average document frequency or average in that bin is low. You can see that the dark line is significantly lower for the average bin than the document frequency bin. The light line represents the standard deviation for a smaller bin where the average document frequency or average value is high. You can see that the standard deviation for average bins is now similar to document frequency bins.



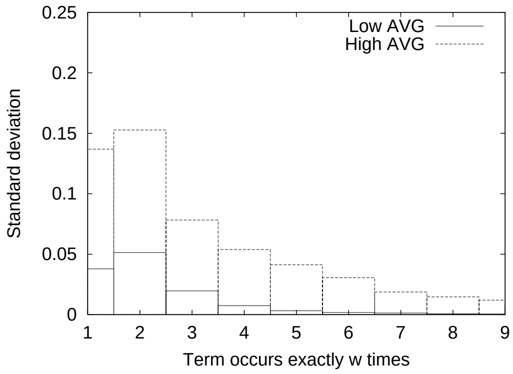
(a) *df* – Data Set 1



(b) *avg* – Data Set 1



(c) *df* – Data Set 2



(d) *avg* – Data Set 2

Figure 4-12: Comparison of standard deviation of the t.o.p.d. for terms binned by document frequency and conditional average.

Chapter 5

Building a Better Model

In Chapter 4 we saw how the multinomial model isn't reflected in the TREC data. We also gained a better understanding of what sort of information, if not that which is predicted by the multinomial model, is actually available in the text. Now we will look at how we can improve retrieval by improving our model to match what we learned more closely. Recall from Section 4.2 that when building a new model we are interested in focusing on only having one parameter per distribution per term.

Given the textual analysis that we have done, there are several conclusions that we could draw with the constraint of one parameter per term. We could say that, as we saw in Section 4.4.2, once a term occurs it is much more likely to occur again. A closer look at the t.o.p.d. of the average term reveals that it appears virtually straight on a log-log scale. This implies a power law distribution. We could therefore attempt to do retrieval assuming the t.p.d.f. for each term is a power law distribution. We will discuss this approach in Section 5.1.

Or, we could declare, as we saw in Section 4.5.2, that once a term has occurred, the number of additional times that a term occurs always has the same distribution. This would imply that information about the number of term occurrences in a document beyond the first occurrence is irrelevant. A model built using this assumption would be a binary model. The initial probabilistic models were binary, and these were later discarded in favor of models that accounted for the number of term occurrences. It is surprising that our data analysis would lead us back to this model. We will discuss using a binary model in Section 5.2.

We will conclude this chapter by looking at the results we get from these two new models that we try, compared with the results we found for the multinomial model. We find that while the binary model does poorly when we try to estimate the relevant term distribution, it does well when given the correct term distribution. On the other hand, the power law model does well both when trying to estimate the relevant term distribution and when given the correct one.

5.1 Power Law Distribution

In this section we will discuss using a power law distribution to model the t.p.d.f. First we will motivate using a power law distribution. Then we will talk about two ways that we try to model the assumption that the relevant and irrelevant term occurrence distributions follow a power law distribution.

In Section 5.1.2 we will discuss the first method we use to incorporate a power law distribution into our model. In this method we will pre-process each term's t.o.p.d. so that they appear multinomial. Then we can retrieve over these pre-processed distributions using our multinomial infrastructure. This allows us to quickly test our assumption that the t.p.d.f. for each term is a power law distributions.

The second approach we will take involves directly including the power law assumption into our model, and we will discuss this in Section 5.1.3. However, it turns out that the assumptions we make to do this are too strong. We will discuss some of the problems we have with this model, as well as potential ways to fix them. Since we have not yet avoided the problems, though, in Section 5.3, when we compare the results we get from this model with the multinomial model and the other model we develop, we will use the first approach to produce the power law model results.

5.1.1 Motivation

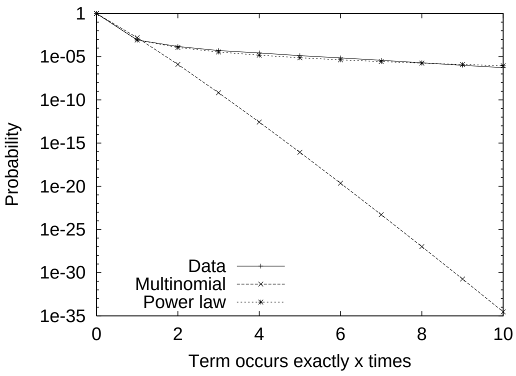
Recall from Section 4.4.2 that the multinomial model implies that the t.p.d.f. for a term is more or less straight on a graph with a log-scale y-axis. However, as we saw in Figure 4-5 during our discussion of our data analysis, the empirical t.o.p.d. isn't straight on a log scale. We propose instead that the t.p.d.f. is a power law distribution.

Figure 5-1 consists of four graphs of the t.o.p.d. that we found through our textual analysis. In the first two graphs, graphs a and b, the multinomial distribution is also displayed. A power law distribution is also graphed on these two graphs, but since it matches the data so much more closely than the multinomial distribution, it is hard to tell the two lines apart. When we ignore the expected distribution for the multinomial model and focus on only the data and the power law distribution, we find that the two distributions actually are very close.

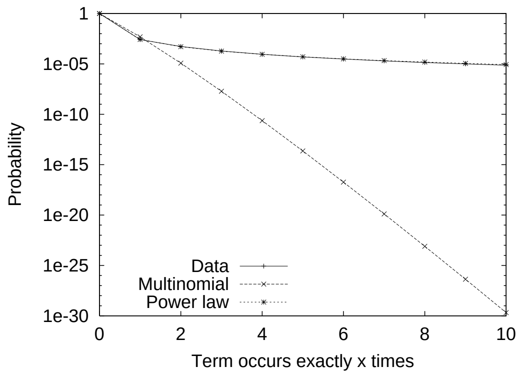
A power law distribution defines the probability of observing ω occurrences of a term as being proportional to $(a+\omega)^b$, where a and b are constants for that distribution. Let us discuss what sort of behavior we expect from our terms if they are distributed according to a power law distribution. If the probability of observing a term ω number of times ($\Pr(\omega)$) is proportional to $(a+\omega)^b$ then $\log(\Pr(\omega)) \propto b\log(a+\omega)$. This means that a plot of the term occurrence probability distribution should appear straight on a log-log scale. As you can see in Figure 5-2, this is close to true for what we observe in the data.

How we set the values for a and b is, of course, important if we are to use a power law distribution in our model. In Figure 5-1, where we match a power law distribution to the term distribution we observe in the data, for Data Set 1 we set a to 0.114 and b to 2.1. For Data set 2, we use $a = 0.114$ and $b = 1.6$. In Figure 5-2 we see the difference it makes to set $a = 1$ as compared to $a = 0.114$. It is clear that when a is 0.114 the data is closer to being a straight line on a log-log scale. This means that it approximates a power law distribution much more closely than when a is 1.

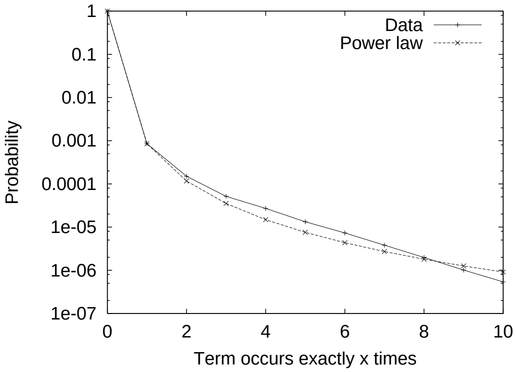
Because we want our model for the term distribution to be represented by only one parameter, while a power law distribution actually requires two, we will take one of these values, namely a, to be constant across all terms. We will consider each term to have its own specific b_i.



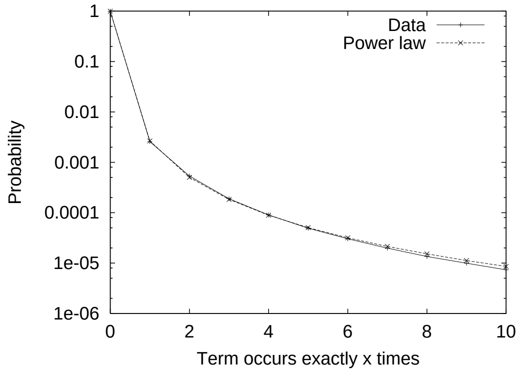
(a) Data Set 1 – with multinomial



(b) Data Set 2 – with multinomial

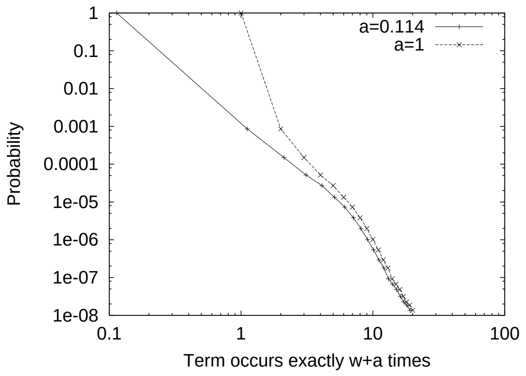


(c) Data Set 1

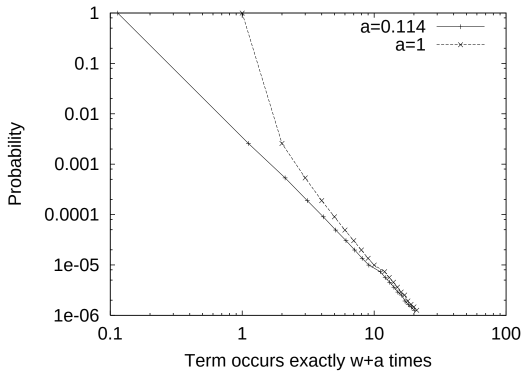


(d) Data Set 2

Figure 5-1: The empirical t.o.p.d. compared with what is expected from multinomial and power law distributions. Even when viewed closely (graphs c and d) the empirical t.o.p.d. overlaps significantly with the t.p.d.f. for a power law distribution.



(a) Data Set 1



(b) Data Set 2

Figure 5-2: The empirical t.o.p.d. for terms in Data Set 1 and 2 shown on a log-log scale, with different values for α. A straight line implies a power law distribution.

5.1.2 Pre-process data to match multinomial model

The multinomial model assumes that the term occurrence distribution is straight on a log scale. The power law model that we propose assumes that the term occurrence distribution is straight on a log-log scale. This suggests that it might be possible to pre-process the data into the straight line distribution that the multinomial model expects, and then to do retrieval using our multinomial infrastructure using this pre-processed data instead.

The pre-processing is simple. Note that the power law distribution, $x^{\log(p)}$, is equal to $p^{\log(x)}$. Compare this with the p^x that is roughly expected by the multinomial model. It becomes apparent that using the log of the number of times a term occurs in a document gives us a term occurrence distribution that is roughly similar to what we would expect from a multinomial model. In other words, the log of the number of occurrences should follow a multinomial distribution.

Thus one model we could use that assumes the t.p.d.f. is a power law distribution is a modified multinomial model. Instead of using the value for d_i^j we could use the value for $\log(1 + d_i^j)$. There is a potential issue here in that by using the log of the term occurrence we will be using real numbers in our multinomial framework rather than integers as we were using before. However, this fortunately does not pose a problem. The results using this method of retrieval are discussed in Section 5.3.

Done in practice Note that our taking the log of the term frequency is similar in practice to the log that is taken in the standard tf.idf vector space model. This model was developed empirically because it performed the best. What we are probably seeing here is a textual motivation for what was developed because it performed the best.

5.1.3 Alter model

The other thing that we can do, given that the probability of a term occurring ω number of times seems to actually obey a power law rather than being multinomial, is to directly alter our model to incorporate a power law distribution. Here we will describe the details of how we can do this. First we will talk about the generative model associated with using a power law distribution, and go into more detail about a power law distribution. Then we will talk about how we learn the distribution using this model. Finally, since this model does not naturally incorporate variable length documents the way that the multinomial model did, we will discuss how we avoid assuming all documents are the same length.

Generative model

The generative model for the power law distribution is quite simple. When creating a document, each term is sampled. Since each term obeys a power law distribution, while most of the time we will get zero occurrences of the term in our document, occasionally we will get one or more occurrences of the term in the document, distributed according to the power law.

Given this is our generative model, it is clear that we do not have any control over the length of a document. The expected document length is the document length found by summing the number of times that we expect each term to occur in a document. Because of the law of large numbers, documents that do not have nearly this expected length will be highly unlikely. This means that we must somehow incorporate length into the model separately.

Understanding a power law distribution

Let us look a little more closely at how we express the probability of the number of term occurrences we will get in our document. In our new model, we want the probability of a term occurring ω number of times be distributed according to a power law ($\Pr(\omega) \propto (a+\omega)^{-b_i}$). Recall that we are taking a to be constant. We are left with only b_i to estimate. If we estimate b_i we know, within our model, everything there is to know about term i.

$$\Pr(d_i^j|b_i) = \frac{(a+d_i^j)^{-b_i}}{\sum_0^\infty (a+d_i^j)^{-b_i}} \tag{5.1}$$

Let us look a little more closely at the normalizing constant. There is actually no closed form solution to this summation. For this reason, we will use the integral of the value instead. We discuss the ramifications of doing this later. But for now, let us solve the integral to find the normalizing factor.

$$\begin{aligned}\int_0^\infty (a+d_i^j)^{-b_i} dd_i^j &= \int_a^\infty y^{-b_i} dy \\ &= \frac{-1}{(b_i-1)y^{b_i-1}}|_a^\infty \\ &= 0 - \frac{-1}{(b_i-1)a^{b_i-1}}\end{aligned}$$

To further simplify this equation, despite the fact that we saw better estimates for a in figures 5-1 and 5-2, we will assume that a is 1.

$$= \frac{1}{b_i - 1}$$

Substituting this into Equation 5.1, we find $Pr(d_i^j|b_i) = (b_i-1)(1+d_i^j)^{-b_i}$.

Continuous v. discrete It is important to note that are using a continuous distribution to model one that is actually discrete. While the same computation we are doing could be done with a discrete distribution, the math in intractable. In some sense using a continuous distribution is reasonable, because the discrete outcomes that we see in the data are possible under a continuous probability distribution. Using a probability density function to model a discrete probability distribution is reasonable if the distribution is more or less flat. However, we are modeling a distribution that drops very quickly from a very high probability of a term not occurring at all to a very small probability of occurring exactly once. It turns out that this sharp drop results in a large enough problem to make the model not work.

That is not to say that there aren't ways around the problem. For example, instead of using the probability density function to model the entire distribution, we could us the discrete values for the probability that a term occurs exactly 0 and 1 times, and then use a density function for the tail, where the distribution is actually much more flat. We have not looked into such hybrid distributions yet, but it is surely worth doing.

Learning the distribution

Let us revisit the learning that we discussed in Section 3.2.6 for the multinomial model, and understand now how we will estimate the relevant term distribution given that we assume the t.p.d.f. a power law distribution. We need to build an understanding of how we can find the true b_i for term i. Since we really don't know it we can place a distribution over the possible values b_i could take on. In order to perform retrieval, we first must find $E[b_i]$. We will use ϑ, as we did in our discussion of the multinomial model, to represent the expected value of this parameter. Finding ϑ involves first understanding the initial estimate we make about the distribution, and then understanding how we use expectation maximization to learn the distribution.

Setting the prior Setting the prior for the distribution, as well as learning it, would be much easier if the distribution we were using matched a distribution for which we already knew a nice update rule, the way we did for the multinomial model. If this were the case, we would not have to derive everything ourselves. One well known distribution that this seems like it might be related to the power law distribution is an exponential distribution. With a little bit of math, we can change our power law distribution into an exponential distribution:

$$\begin{aligned}
\Pr(d_i^j|b_i) &= (b_i - 1)(1 + d_i^j)^{-b_i} \\
\Pr(d_i^j > Z|b_i) &= \int_Z^{\infty} (b_i - 1)(1 + d_i^j)^{-b_i} dd_i^j \\
&= -(1 + d_i^j)^{-(b_i+1)}|_Z^{\infty} \\
&= (1 + Z)^{-(b_i+1)} \qquad (5.2)
\end{aligned}$$

If we now let $y_i^j = \log(1 + d_i^j)$:

$$\begin{aligned}
\Pr(y_i^j > c|b_i) &= \Pr(e^y > e^c) \\
&= \Pr(1 + d_i^j > e^c) \\
&= \Pr(d_i^j > e^c - 1) \\
&= (1 + e^c - 1)^{-(b_i+1)} \qquad \text{Using Eqn. 5.2} \\
&= e^{-c(b_i+1)}
\end{aligned}$$

Of course, what we are interested in is the probability density function of y_i^j. This is easy to find from the above, by simply taking the derivative of $\Pr(y_i^j > c|b_i)$, that $\Pr(y_i^j|b_i) = (b_i + 1)e^{-(b_i+1)y_i^j}$. We arrive at an exponential distribution.

Since this distribution now matches the well understood exponential distribution, we know how to find the expected value for $b_i + 1$, and thus find the probability of observing y_i^j in our data. The expected value of $b_i + 1$ is $\frac{n}{\sum_{j=1}^n \log(1+d_i^j)}$ [2], and using this we can set our initial estimates. The exponential distribution, like the Dirichlet distribution we use for the multinomial model, also has a nice update rule that we can use when learning.

Expectation We will now briefly discuss how we perform learning to find the most likely relevant distribution. Recall Equation 2.1 that says the posterior probability of a document is:

$$\Pr(r_j|\mathbf{d^j}) = \frac{\Pr(\mathbf{d^j}|r_j)\Pr(r_j)}{\Pr(\mathbf{d^j})}$$

In our power law model, term occurrence remains independent, so we can continue to define $\Pr(\mathbf{d^j}|r_j) = \prod_{i=1}^{m} \Pr(d_i^j|r_j)$. Given our new model, this value becomes $\prod_{i=1}^{m}(b_i + 1)e^{-(b_i+1)y_i^j}$. Note that while for the multinomial model we could ignore terms not in a document, we can no longer do that. We can, however, factor a constant out of the distribution for comparable efficiency.

Recall that in the expectation step we are given the expected values for the parameters of our distributions, ϑ_i and $\hat{\vartheta}_i$ This means that the posterior probability of a document is:

$$\Pr(r_j = 1|\mathbf{d^j}) = \frac{p^{\text{rel}} \prod_{i=1}^{m} \vartheta_i e^{-\vartheta_i \log(1+d_i^j)}}{p^{\text{rel}} \prod_{i=1}^{m} \vartheta_i e^{-\vartheta_i \log(1+d_i^j)} + (1-p^{\text{rel}}) \prod_{i=1}^{m} \hat{\vartheta}_i e^{-\hat{\vartheta}_i \log(1+d_i^j)}}$$

Maximization In the "maximization" step we set the parameter for the t.p.d.f. for each term to maximize the likelihood of observing our data. Our belief about the distribution over b_i has a simple update rule similar to that for a multinomial that we can use to do this.

Recall that for the multinomial model, the Dirichlet prior allowed us to update our belief about the corpus by simply adding in our observations. Here, also, we can update our belief about the corpus by adding in our observations, in this case our observations of $\log(1 + d_i^j)$. This is another motivation for setting a to one. Since $\log(1) = 0$, we do not need to update our distribution for terms which do not occur in a document. $\vartheta_i = \frac{n}{\sum_{j=1}^{n} \log(1+d_i^j)}$. This simplicity comes directly from the fact that we are using an exponential distribution, and Bernando and Smith's book on Bayesian theory is a good resource to learn more [2].

Incorporating length

Length is also an important factor for us to consider when using a power law model. While not the case with the multinomial model, often models have the implicit assumption that all documents are of the same length, and require some sort of normalization before hand to make the actual documents match this assumption. The multinomial model is free of this assumption because it assumes that each word occurrence is chosen independently. Length does not affect the probability of term selection. However, as we have mentioned, the power law model we describe here does need to take special consideration when understanding length.

Here we will look at the effect that length has on our documents in an attempt to better understand how we should incorporate length into our model. While we can find the expected length of a document, it is messy. We need to find the expected probability of seeing each term and sum over them. However, while finding the expected value for d_i^j is difficult, finding the expected value for $\log(1 + d_i^j)$ is straightforward.

$$\begin{aligned} E[\log(1+d_i^j)] = E[y_i] &= \int_0^\infty \Pr(y_i^j > c) db_i \\ &= \int_0^\infty e^{-(1+b_i)c} db_i \\ &= \frac{1}{1+b_i} \end{aligned}$$

Let us define the document statistic *log-length*. The log-length of a document refers to the sum of the log of number of times each term occurs in the document. We will refer to the log-length as ll_j. $ll_j = \sum_{i=1}^m \log(1 + d_i^j)$. You can see that in this model there is an expected log-length for a document, and anything that varies from it is very unlikely.

Given that b_i is not a function of the log-length of a document, the log-length will be constant constant, namely $\sum_{i=1}^m \frac{1}{1+b_i}$. What we would like is for the expected value of $\log(1 + x)$ to scale linearly with the log-length, ll_j. If, instead of $b_i + 1$, we use $\frac{b_i+1}{ll_j}$, we will get this desired behavior.

$$\begin{aligned} \Pr(y_i^j > c) &= \int_0^\infty e^{-(1+b_i)\frac{c}{ll_j}}\, db_i \\ &= e^{-c\frac{(b_i+1)}{ll_j}} \end{aligned}$$

This means that we should observe $\frac{y_i^j}{ll_j}$ so that we have an exponential distribution again (since $\Pr(\frac{y_i^j}{ll_j} > c) = e^{-c(1+b_i)}$).

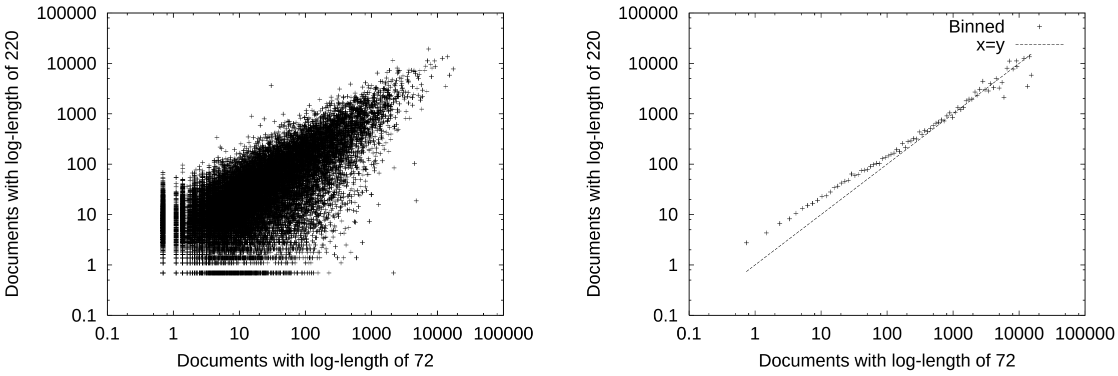
Whether or not this is a reasonable thing to do involves going to the data. Let us look at the effect of log-length in relationship to the log of the number of times a term occurs ($\log(1+d_i^j)$). We look at the average value for $\log(1+d_i^j)$ for each term in a corpus of documents of two different log-lengths. This allows us to understand if length effects all terms in the same way.

Let's look closely at how a document's log-length effects the average value for $\log(1+d_i^j)$ on a term by term basis. To do this, as we earlier binned documents into two bins of documents with more or less the same lengths, we now bin documents into two bins of documents with more or less the same log-lengths. The document in the first bin have a log-length of near 72, and the documents in the second bin have a log-length of near 220. For every term occurring in either set, we look at its average $\log(1+d_i^j)$ in both document sets.

These values can be seen in Figure 5-3. Because there are so many terms, we provide in addition to the graph of each individual term the binned values showing us the overall trend. The relationship, on a log-log scale, is reasonably close to the $x = y$ line, but not perfectly, or, in other words, close to linear, but not actually linear. If it were linear, that would imply that the value $\log(1+d_i^j)$ is linear with respect to log-length. Although this is not exactly the case, the model seems plausible.

5.2 Binary Model

The other sort of model we investigate is a binary model. A binary model postulates that only a term's occurrence in a document, and not the number of times that term



(a) All terms

(b) Binned

Figure 5-3: The average value of $\log(1+\omega)$ for a term when it appears in documents of two different log lengths. Although the relationship is reasonably close to linear, it is not actually linear.

occurs in the document, is important for retrieval. Initial probabilistic models were built taking into account only term presence, but were later updated to account for term frequency within a document. For example, McCallum and Nigam discuss how they find improved performance for naive Bayes text classification when not using binary features [26].

One potential problem for the binary model is similar to the problem that we saw in the power law model, and that is the problem of length. The generative binary model is that each term either appears or does not appear in a document. There is no way to control the length of the document, and lengths that are distant from the expected length are not likely to occur. Early binary models did not have this problem, since early corpora consisted of documents of primarily the same length. However, modern corpora, such as TREC, have much greater variation for length that must be accounted for [18]. Here, however, we discuss our first investigations into using a binary model, where we do not account for length.

As we do for the power law model, we approximate a binary model within our multinomial model framework. This allows us to quickly implement a binary model without too much additional coding. When we observe a term in a document, instead of using the actual count of the number of times a term occurs in that document, we use the value one. This allows us to quickly do initial tests of a binary model without a significant amount of coding.

The simple modification is a reasonable approximation for the following reason. Suppose we are generating a document with ℓ terms. In the multinomial generative model, we do this by sampling from the appropriate term distribution in the multinomial model. However, in the binary model, where we are only interested in the presence or absence of a term, we want to guarantee that each of these terms is unique.

Actually, the odds that we get more than one occurrence of a term with the multinomial model is fairly low. In fact, this is exactly why we had trouble with the standard multinomial model; the model was not predicting that terms occur in documents as often as they do. If the odds of getting duplicates are small enough, we can ignore this possibility and assume that all ℓ terms we generate are actually distinct.

The difference between the multinomial model and a binary model lies in the document generation. When generating a document under the multinomial model, we sample with replacement from a term distribution. When generating a document under the binary model, we sample without replacement. But removing a few terms doesn't change the probabilities very much at all, so the probability of a term appearing in a document is very similar either way.

5.3 Results

In this section we look at the results we find when we test the two new models described here, the power law model and the binary model, as compared with the multinomial model we begin with. We use the same corpus we use for testing the multinomial model, described in Section 3.3.1 to test the new models.

Recall from Section 3.3, where we discuss the results we found for the multinomial model, that there are three distinct approaches that we take when looking at the retrieval performance of a model, based on how we find the relevant term distribution.

The three term distributions that we retrieve with are, one, a crude initial estimate that merely says the query terms are more likely in the relevant term distribution, two, the maximum likelihood estimate that we find using machine learning, and three, the correct distribution, found by using the documents that are actually relevant.

5.3.1 With an initial estimate

Here we discuss what we found by doing retrieval using only our initial guess as to the relevant distribution. The results that we find using this distribution represent the quality of search we could expect to be able to retrieve given that we wanted to maintain performance time similar to that of the vector space retrieval systems. It does not necessarily reflect how well the model matches the text but rather also encompasses the quality of our initial estimate. The results can be seen in Figure 5-4.

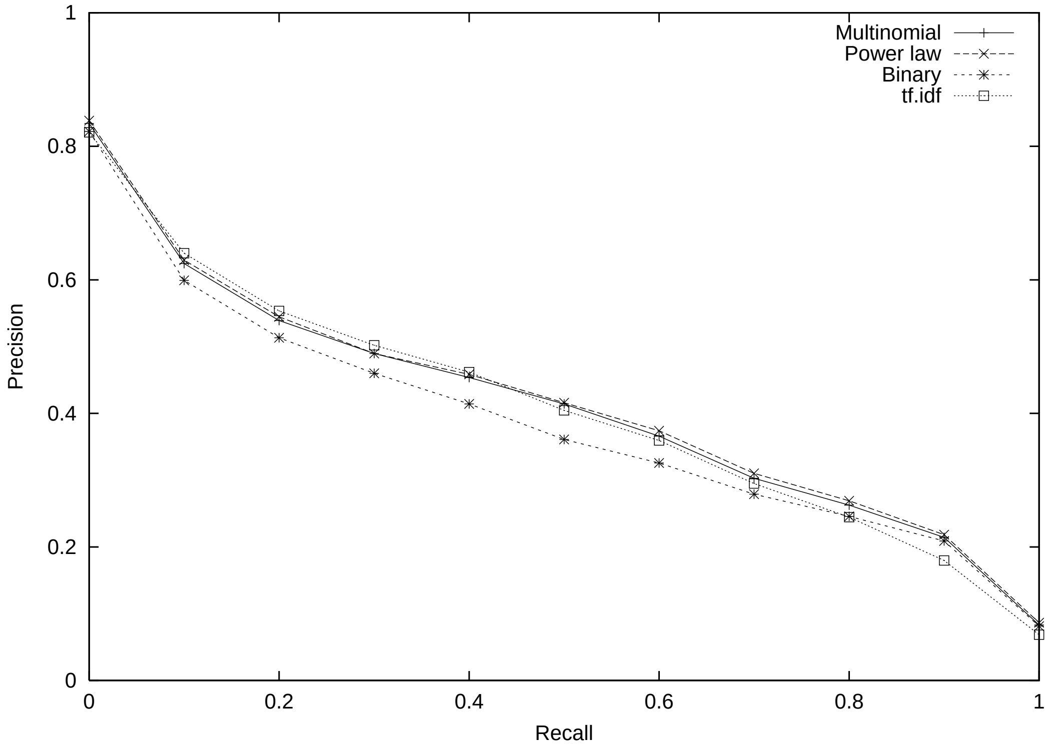


Figure 5-4: Results found using the new models. The relevant distribution used is the initial estimated distribution.

It appears that the power law model and the multinomial model produce results that are remarkably similar. However, the binary model sees a marked decrease in performance. It appears matching the model more closely to the data does not help in this case.

5.3.2 With learning

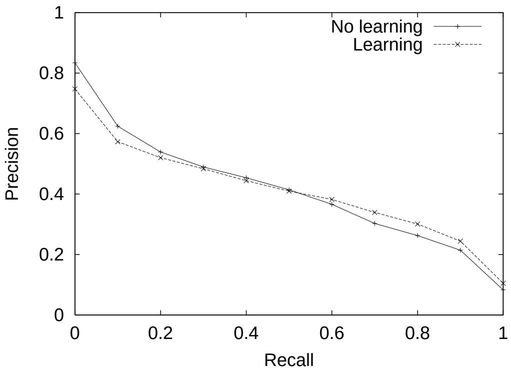
Now let us discuss what we found by doing retrieval using the relevant term distribution we find after performing expectation maximization in an attempt to learn a distribution that increases the likely of the corpus. This gives us some idea as to how good our model combined with our initial estimate of the relevant distribution and our learning technique is. Only if all of these factors work successfully will we see improvement in our results.

Recall from Section 3.2.7 how we discuss there are several constants over which we have control when building a model. We show the results for a good setting of these constants in Figure 5-5. With learning you can see that the multinomial results performs the best. Only in the very low recall portion of the curve do we see the precision drop. For most of the curve the precision is improved. Because the binary model changes in a similar manner to the multinomial model with learning, it continues to perform the least well of the three models.

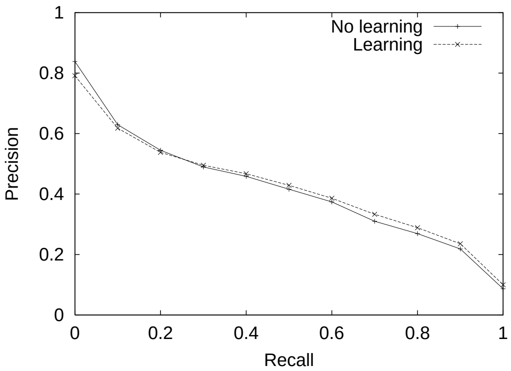
5.3.3 With the correct distribution

We look also at the results we get with our models using the correct relevant term distribution. Using the correct distributions gives us an idea of how good the model is unclouded by the quality of our estimation. It does not, however, give us an idea of the quality of results we could expect to obtain for our retrieval system.

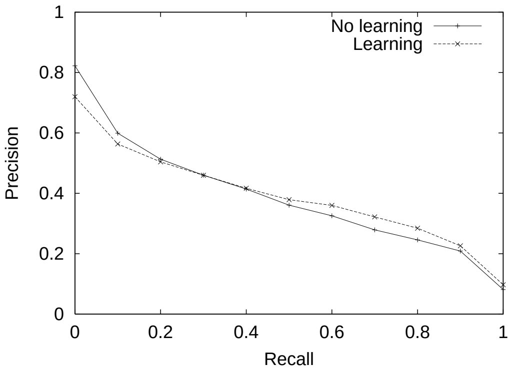
We discuss specifically how we find the correct distributions with respect to the multinomial model in Section 3.3.3. Recall that estimating the correct distribution is similar to how we make our initial crude estimate of the relevant term distribution, only instead of viewing the query as a relevant document, we view all of the correct results as being labelled relevant. Because we continue to start with the corpus distribution as our prior on the term distribution, there is some nonzero probability



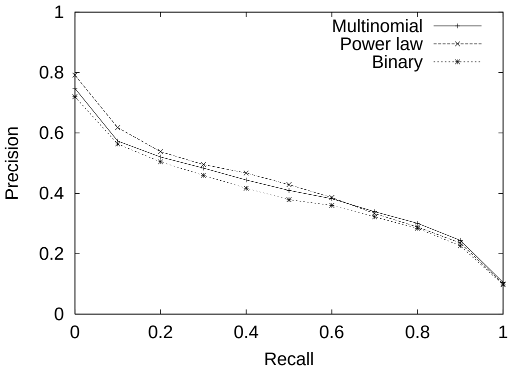
(a) Multinomial model – no learning v. learning



(b) Power law model – no learning v. learning



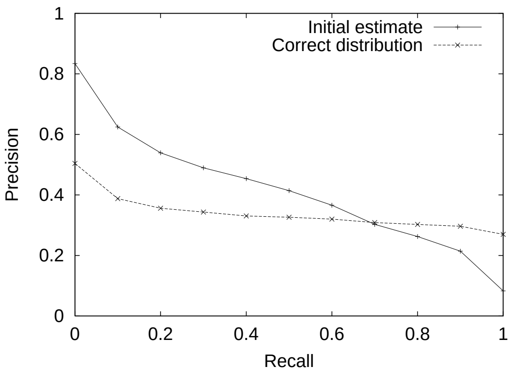
(c) Binary model – no learning v. learning



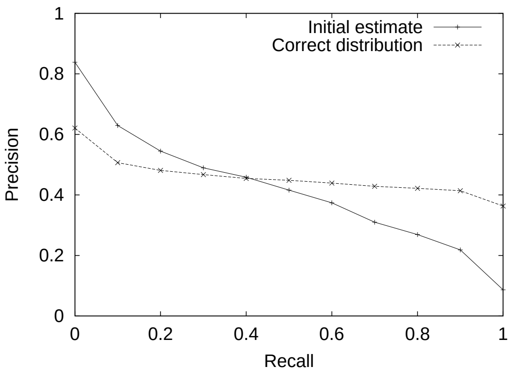
(d) Comparison of learning across models

Figure 5-5: Results found using the new models. The relevant distribution is found with learning.

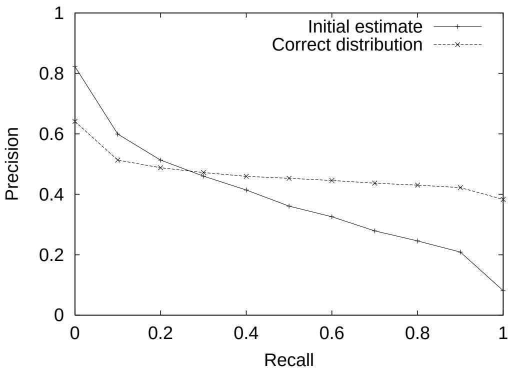
of observing every term. Recall from section 3.2.7 that the prior that we place on the terms is as strong as if we had observed as many documents as are relevant with the corpus distribution.



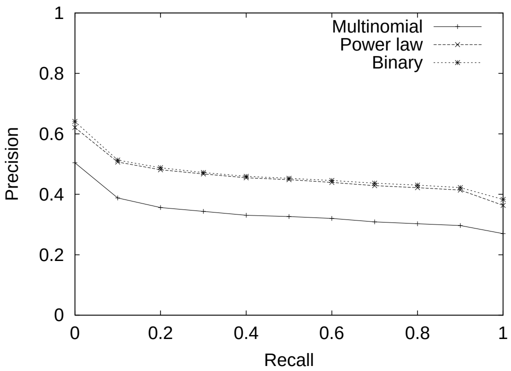
(a) Multinomial model – initial estimate v. correct distribution



(b) Power law model – initial estimate v. correct distribution



(c) Binary model – initial estimate v. correct distribution



(d) Comparison of correct distribution across models

Figure 5-6: Results found using the new models. The relevant distribution is found by learning from the correct distribution.

As you can see in Figure 5-6, both the power law model and the binary model perform significantly better than the multinomial model. This implies that, while there may be difficulties with learning the distribution, we do have a better understanding of the distribution after our analysis. We can see that is is not because the binary model matches the data poorly that people began using term frequency. The binary model does do well with the correct distribution. It is rather because the binary model performs poorly when we try to estimate the relevant term distribution that they do so.

Learning from the correct distribution

When we try using expectation maximization from the correct distribution, if our model is correct, we should not be able to find a distribution that makes our corpus more likely. The results from learning from the correct distribution can be seen in Figure 5-7. As you can see, all models learn a new distribution, although the binary model seems to do so the least. This means that, while we may have found better models to use for retrieval, they are still not perfect.

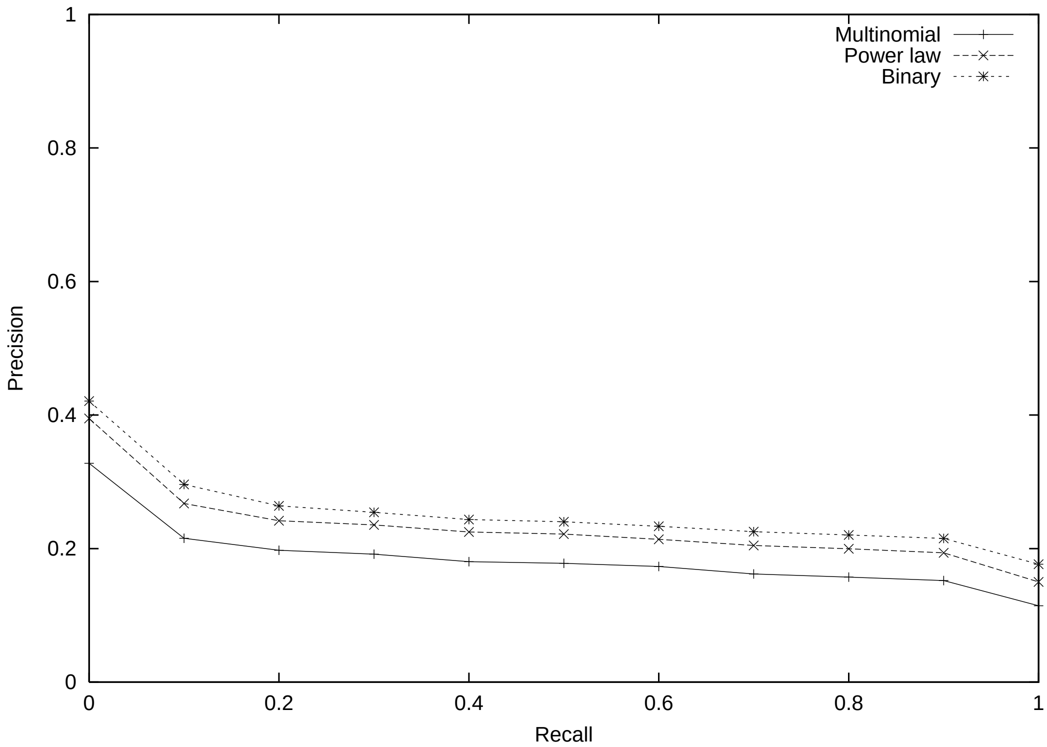


Figure 5-7: Learning from the correct distribution.

Chapter 6

Conclusion

In this thesis we have discussed the importance of building a model for information retrieval. One of the benefits we talked about in using a model is that it allows us to make our assumptions explicit. However, the primary advantage of having explicit assumptions is that then we analyze them. By understanding how valid our assumptions are we can understand how valid our model is. A model is useless if we do not ground the assumptions that it is build on in the real world. For this reason, we have used textual analysis to understand which assumptions in the multinomial model are broken. We have endeavored to fix those assumptions using what we learned about the text. This processes, we have shown, results in improved retrieval.

Specifically, we found that a term appears much more clustered within a document than is predicted by a multinomial model. When we updated the model to reflect the power law distribution that the terms more closely resembled, we found that the quality of the search results improved. We were able to do this while not changing the retrieval complexity.

6.1 Future Work

What we have found so far has been very exciting, but we are really only just beginning to understand the statistical properties of the text. There are a number of interesting avenues to pursue from here. They fall under the same categories as what we have pursued in this thesis. There is much more to understand about the statistical properties of term occurrence within the text. There are also new models to try even given just the textual analysis that we have already done. Additionally, the improved models that we propose in this thesis can be understood further.

6.1.1 Analysis

There is much more we can learn from the text. We will continue to explore ways to find the hidden parameter that describes term occurrence from the observations of term occurrence that we have. One specific type of analysis that we are interested in pursuing is query specific analysis.

Query specific analysis

So far we have been looking at the corpus as a whole. We are interested in looking at what differences are actually seen in the term distributions between relevant documents and non-relevant documents. In addition to learning more about the distributions, by looking at where the query terms fall in these distributions, we can incorporate the query terms in a more principled way.

6.1.2 Model building

We had some trouble implementing a power-law model because, to simplify the math, we assumed that the t.o.p.d. was continuous rather than discrete. However, since we saw evidence that using a power-law distribution does improve retrieval, it is worth pursuing this model further. The power-law model can surely be improved incorporating a discrete distribution instead. If a discrete distribution proves to difficult to work with, we could explore a hybrid of discrete and continuous distributions.

Also, a two parameter model suggests itself given the analysis we have done into document frequency, the conditional average, and their relationship. One possible generative model is that we use the document frequency to determine if a term appears in a document or not, and then, if the term appears, we use the conditional average to determine how many times the term should appear.

6.1.3 Testing

In order to allow us to understand the results we get, as well as to do the necessary machine learning without worrying too much about efficiency, we have done our testing with a relatively small corpus. By running the same tests on a larger corpus, we can get a better idea of how successful the modifications we have made to the model really are.

We also plan to use cross-validation to determine the optimal parameter settings for our models when using the empirically correct term distributions. In doing so, the comparison between performance of different models on the correct distribution will be more fair.

Bibliography

[1] Adam Berger and John D. Lafferty. Information retrieval as statistical translation. In *Research and Development in Information Retrieval*, pages 222–229, Berkeley, CA, August 1999.

[2] Jose M. Bernardo and Adrian F.M. Smith. *Bayesian Theory*. John Wiley and Son Ltd, London, first edition, 1994.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[4] Chris Buckley, Amit Singhal, Mandar Mitra, and Gerard Salton. New retrieval approaches using SMART: TREC 4. In *The Fourth Text REtrieval Conference (TREC-4)*, 1996.

[5] Kenneth W. Church. One term or two? In *18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[6] W. Bruce Croft and David J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35:285–295, 1979.

[7] William Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.

[8] Nir Friedman. The bayesian structural EM algorithm. In *Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, San Francisco, California, 1998.

[9] Norbert Fuhr. Models in information retrieval.

[10] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[11] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *Research and Development in Information Retrieval*, pages 121–128, 1999.

[12] Warren R. Greiff. A theory of term weighting based on exploratory data analysis. In *Proceedings of SIGIR-98*, Melbourn, Australia, August 1998.

[13] Alan Griffith, H. Claire Luckhurst, and Peter Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37:3 – 11, 1986.

[14] Donna Harman. The trec conferences. In *Hypertext, Information REtrieval, Multimedia*, pages 9–28, 1995.

[15] S. Harter. A probabilistic approach to automatic keyword indexing, part I. on the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206, July-August 1975.

[16] S. Harter. A probabilistic approach to automatic keyword indexing, part II. an algorithm for probabilistic indexing. *Journal of the American Society for Information Science*, 26(4):280–289, July-August 1975.

[17] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR_TR-95-06, Microsoft Research, Redmond, Washington, 1995. Revised June 1996.

[18] K. Sparck Jones, S. Walker, and S.E. Robertson. A probabilistic model of information retrieval: development and status. Technical Report TR-446, Cambridge University Computer Laboratory, September 1998.

[19] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[20] T. Kalt. A new probabilistic model of text classification and retrieval. Technical Report IR-78, University of Massachusetts Center for Intelligent Information Retrieval, 1996.

[21] Boris Katz. From sentence processing to information access on the world wide web. In *AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, Stanford, CA, 1997.

[22] Michelle Keim, David D. Lewis, and David Madigan. Bayesian information retrieval: Preliminary evaluation. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 303–310, Fort Lauderdale, Florida, January 1997.

[23] K. L. Kwok and M. Chan. Improving two-stage ad-hoc retrieval for short queries. In *Research and Development in Information Retrieval*, pages 250–256, 1998.

[24] David D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In *EMCL*, pages 4–15, 1998.

[25] Christopher D. Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. The MIT Press, first edition, 1999.

[26] Andrew McCallum and Kamal Nigam. A comparison of event models for naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[27] Kathleen McKeown, Judith Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. Towards multidocument summarization by reformulation: Progress and prospects. In *AAAI/IAAI*, 1999.

[28] Kenny Ng. A maximum likelihood ratio information retrieval model. In *Eighty Text REtrieval Conference (TREC-8)*, 1999.

[29] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Using EM to classify text from labeled and unlabeled documents. Technical Report CMU-CS-98-120, School of Computer Science, CMU, Pittsburgh, PA 15213, 1998.

[30] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US, 1998. AAAI Press, Menlo Park, US.

[31] Yoram Singer Nir Friedman. Efficient bayesian parameter estimation. In *Advances in Neural Information Processing Systems*. The MIT Press, 1999.

[32] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Research and development in Information Retrieval*, pages 275–281, 1998.

[33] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[34] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[35] S. E. Roberston. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, December 1977.

[36] S. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, Dublin, Ireland, 1994.

[37] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *IEEE*, 2000.

[38] M. Sahami. Learning limited dependence Bayesian classifiers. In *KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 335–338, 1996.

[39] G. Salton. *The SMART retrieval system: experiments in automatic document processing*. Prentice-Hall, Englewood Cliffs, N.J., 1971.

[40] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[41] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large altavista query log. Technical Report 1998-014, Digital Systems Research Center, October 1998.

[42] Don Swanson. Information retrieval as a trial-and-error process. *Library Quarterly*, 47(2):128–148, 1977.

[43] C. J. vanRijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.