Recent Advances in Applied Probability

Recent Advances in Applied Probability

Edited by

RICARDO BAEZA-YATES
Universidad de Chile, Chile

JOSEPH GLAZ
University of Connecticut, USA

HENRYK GZYL
Universidad Simón Bolívar, Venezuela

JÜRGEN HÜSLER
University of Bern, Switzerland

JOSÉ LUIS PALACIOS
Universidad Simón Bolívar, Venezuela

Springer

eBook ISBN: 0-387-23394-6
Print ISBN: 0-387-23378-4

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

Contents

Preface

The possibility of the present collection of review papers came up the last day of IWAP 2002. The idea was to gather in a single volume a sample of the many applications of probability.

As a glance at the table of contents shows, the range of covered topics is wide, but it sure is far away of being close to exhaustive.

Picking up a name for this collection not easier than deciding on a criterion for ordering the different contributions. As the word ‘advances” suggests, each paper represents a further step toward understanding a class of problems. No last word on any problem is said, no subject is closed.

Even though there are some overlaps in subject matter, it does not seem sensible to order this eclectic collection except by chance, and such an order is already implicit in a lexicographic ordering by first author’s last name: Nobody (usually, that is) chooses a last name, does she/he? So that is how we settled the matter of ordering the papers.

We thank the authors for their contribution to this volume.

We also thank John Martindale, Editor, Kluwer Academic Publishers, for inviting us to edit this volume and for providing continual support and encouragement.

Acknowledgments

The editors thank the Cyted Foundation, Institute of Mathematical Statistics, Latin American Regional Committee of the Bernoulli Society, National Security Agency and the University of Simon Bolivar for co-sponsoring IWAP 2002 and for providing financial support for its participants.

The editors warmly thank Alfredo Marcano of Universidad Central de Venezuela for having taken upon his shoulders the painstaking job of rendering the different idiosyncratic contributions into a unified format.

MODELING TEXT DATABASES

Ricardo Baeza-Yates
Depto. de Ciencias de la Computación
Universidad de Chile
Casilla 2777, Santiago, Chile
rbaeza@dcc.uchile.cl

Gonzalo Navarro
Depto. de Ciencias de la Computación
Universidad de Chile
Casilla 2777, Santiago, Chile
gnavarro@dcc.uchile.cl

Abstract We present a unified view to models for text databases, proving new relations between empirical and theoretical models. A particular case that we cover is the Web. We also introduce a simple model for random queries and the size of their answers, giving experimental results that support them. As an example of the importance of text modeling, we analyze time and space overhead of inverted files for the Web.

1.1 Introduction

Text databases are becoming larger and larger, the best example being the World Wide Web (or just Web). For this reason, the importance of the information retrieval (IR) and related topics such as text mining, is increasing every day [Baeza-Yates & Ribeiro-Neto, 1999]. However, doing experiments in large text collections is not easy, unless the Web is used. In fact, although reference collections such as TREC [Harman, 1995] are very useful, their size are several orders of magnitude smaller than large databases. Therefore, scaling is an important issue. One partial solution to this problem is to have good models of text databases to be able to analyze new indices and searching algorithms before making the effort of trying them in a large scale. In particular if our application is searching the Web. The goals of this article are two fold: (1) to present in an integrated manner many different results on how to model nat-

ural language text and document collections, and (2) to show their relations, consequences, advantages, and drawbacks.

We can distinguish three types of models: (1) models for static databases, (2) models for dynamic databases, and (3) models for queries and their answers. Models for static databases are the classical ones for natural language text. They are based in empirical evidence and include the number of different words or vocabulary (Heaps' law), word distribution (Zipf's law), word length, distribution of document sizes, and distribution of words in documents. We formally relate the Heaps' and Zipf's empirical laws and show that they can be explained from a simple finite state model.

Dynamic databases can be handled by extensions of static models, but there are several issues that have to be considered. The models for queries and their answers have not been formally developed until now. Which are the correct assumptions? What is a random query? How many occurrences of a query are found? We propose specific models to answer these questions.

As an example of the use of the models that we review and propose, we give a detailed analysis of inverted files for the Web (the index used in most Web search engines currently available), including their space overhead and retrieval time for exact and approximate word queries. In particular, we compare the trade-off between document addressing (that is, the index references Web pages) and block addressing (that is, the index references fixed size logical blocks), showing that having documents of different sizes reduces space requirements in the index but increases search times if the blocks/documents have to be traversed. As it is very difficult to do experiments on the Web as a whole, any insight from analytical models has an important value on its own.

For the experiments done to backup our hypotheses, we use the collections contained in TREC-2 [Harman, 1995], especially the Wall Street Journal (WSJ) collection, which contains 278 files of almost 1 Mb each, with a total of 250 Mb of text. To mimic common IR scenarios, all the texts were transformed to lower-case, all separators to single spaces (except line breaks); and stopwords were eliminated (words that are not usually part of query, like prepositions, adverbs, etc.). We are left with almost 200 Mb of filtered text. Throughout the article we talk in terms of the size of the filtered text, which takes 80% of the original text. To measure the behavior of the index as n grows, we index the first 20 Mb of the collection, then the first 40 Mb, and so on, up to 200 Mb. For the Web results mentioned, we used about 730 thousand pages from the Chilean Web comprising 2.3Gb of text with a vocabulary of 1.9 million words.

This article is organized as follows. In Section 2 we survey the main empirical models for natural language texts, including experimental results and a discussion of their validity. In Section 3 we relate and derive the two main empirical laws using a simple finite state model to generate words. In Sections 4 and 5 we survey models for document collections and introduce new models

for random user queries and their answers, respectively. In Section 6 we use all these models to analyze the space overhead and retrieval time of different variants of inverted files applied to the Web. The last section contains some conclusions and future work directions.

1.2 Modeling a Document

In this section we present distributions for different objects in a document. They include characters, words (unique and total) and their length.

1.2.1 Distribution of Characters

Text is composed of symbols from a finite alphabet. We can divide the symbols in two disjoint subsets: symbols that separate words and symbols that belong to words. It is well known that symbols are not uniformly distributed. If we consider just letters (a to z), we observe that vowels are usually more frequent than most consonants (e.g., in English, the letter 'e' has the highest frequency.) A simple model to generate text is the Binomial model. In it, each symbol is generated with certain fixed probability. However, natural language has a dependency on previous symbols. For example, in English, a letter 'f' cannot appear after a letter 'c' and vowels, or certain consonants, have a higher probability of occurring after 'c'. Therefore, the probability of a symbol depends on previous symbols. We can use a finite-context or Markovian model to reflect this dependency. The model can consider one, two or more letters to generate the next symbol. If we use k letters, we say that it is a k-order model (so the Binomial model is considered a 0-order model). We can use these models taking words as symbols. For example, text generated by a 5-order model using the distribution of words in the Bible might make sense (that is, it can be grammatically correct), but will be different from the original [Bell, Cleary & Witten, 1990, chapter 4]. More complex models include finite-state models (which define regular languages), and grammar models (which define context free and other languages). However, finding the correct complete grammar for natural languages is still an open problem.

For most cases, it is better to use a Binomial distribution because it is simpler (Markovian models are very difficult to analyze) and is close enough to reality. For example, the distribution of characters in English has the same average value of a uniform distribution with 15 symbols (that is, the probability of two letters being equal is about 1/15 for filtered lowercase text, as shown in Table 1).

1.2.2 Vocabulary Size

What is the number of distinct words in a document? This set of words is referred to as the document *vocabulary*. To predict the growth of the vocabulary

size in natural language text, we use the so called *Heaps' Law* [Heaps, 1978], which is based on empirical results. This is a very precise law which states that the vocabulary of a text of n words is of size $V = Kn^{\beta} = \Theta(n^{\beta})$, where K and β depend on the particular text. The value of K is normally between 10 and 100, and β is a positive value less than one. Some experiments [Araújo et al, 1997; Baeza-Yates & Navarro,1999] on the TREC-2 collection show that the most common values for β are between 0.4 and 0.6 (see Table 1). Hence, the vocabulary of a text grows sub-linearly with the text size, in a proportion close to its square root. We can also express this law in terms of the number of words, which would change K.

Notice that the set of different words of a language is fixed by a constant (for example, the number of different English words is finite). However, the limit is so high that it is much more accurate to assume that the size of the vocabulary is $O(n^{\beta})$ instead of $O(1)$ although the number should stabilize for huge enough texts. On the other hand, many authors argue that the number keeps growing anyway because of the typing or spelling errors.

How valid is the Heaps' law for small documents? Figure 1 shows the evolution of the β value as the text collection grows. We show its value for up to 1 Mb (counting words). As it can be seen, β starts at a higher value and converges to the definitive value as the text grows. For 1 Mb it has almost reached its definitive value. Hence, the Heaps' law holds for smaller documents but the β value is higher than its asymptotic limit.

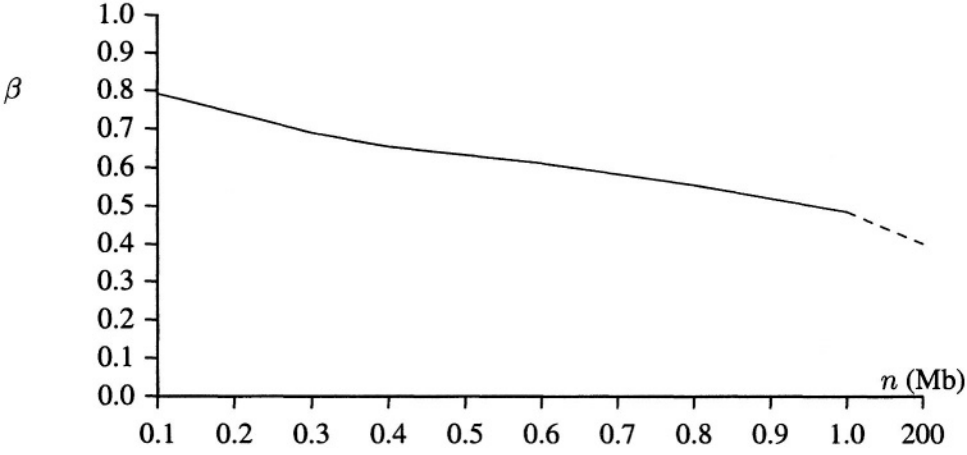

Figure 1. Value of β as the text grows. We added at the end the value for the 200 Mb collection.

For our Web data, the value of β is around 0.63. This is larger than for English text for several reasons. Some of them are spelling mistakes, multiple languages, etc.

1.2.3 Distribution of Words

How are the different words distributed inside each document?. An approximate model is the *Zipf's Law* [Zipf, 1949; Gonnet & Baeza-Yates, 1991], which attempts to capture the distribution of the frequencies (that is, number of occurrences) of the words in the text. The rule states that the frequency of the i-th most frequent word is $1/i^{\theta}$ times that of the most frequent word. This implies that in a text of n words with a vocabulary of V words, the i-th most frequent word appears $n/(i^{\theta} H_V(\theta))$ times, where $H_V(\theta)$ is the harmonic number of order θ of V, defined as

$$H_V(\theta) = \sum_{j=1}^{V} \frac{1}{j^{\theta}}$$

so that the sum of all frequencies is n. The value of θ depends on the text. In the most simple formulation, $\theta = 1$, and therefore $H_V(\theta) = O(\log n)$. However, this simplified version is very inexact, and the case $\theta > 1$ (more precisely, between 1.7 and 2.0, see Table 1) fits better the real data [Araújo et al, 1997]. This case is very different, since the distribution is much more skewed, and $H_V(\theta) = O(1)$. Experimental data suggests that a better model is $k/(c+i)^{\theta}$ where c is an additional parameter and k is such that all frequencies add to n. This is called a Mandelbrot distribution [Miller, Newman & Friedman, 1957; Miller, Newman & Friedman, 1958]. This distribution is not used because its asymptotical effect is negligible and it is much harder to deal with mathematically.

It is interesting to observe that if, instead of taking text words, we take n-grams, no Zipf-like distribution is observed. Moreover, no good model is known for this case [Bell, Cleary & Witten, 1990, chapter 4]. On the other hand, Li [Li, 1992] shows that a text composed of random characters (separators included) also exhibits a Zipf-like distribution with smaller θ, and argues that the Zipf distribution appears because the rank is chosen as an independent variable. Our results relating the Zipf's and Heaps' law (see next section), agree with that argument, which in fact had been mentioned well before [Miller, Newman & Friedman, 1957].

Since the distribution of words is very skewed (that is, there are a few hundred words which take up 50% of the text), words that are too frequent, such as *stopwords,* can be disregarded. A stopword is a word which does not carry meaning in natural language and therefore can be ignored (that is, made not searchable), such as `"a"`, `"the"`, `"by"`, etc. Fortunately the most frequent words are stopwords, and therefore half of the words appearing in a text do not need to be considered. This allows, for instance, to significantly reduce the space overhead of indices for natural language texts. Nevertheless, there are very frequent words that cannot be considered as stopwords.

For our Web data, $\theta = 1.59$, which is smaller than for English text. This what we expect if the vocabulary is larger. Also, to capture well the central part of the distribution, we did not take in account very frequent and unfrequent words when fitting the model. A related problem is the distribution of k-grams (strings of exactly k characters), which follow a similar distribution [Egghe, 2000].

1.2.4 Average Length of Words

A last issue is the average length of words. This relates the text size in words with the text size in bytes (without accounting for punctuation and other extra symbols). For example, in the different sub-collections of TREC-2 collection, the average word length is very close to 5 letters, and the range of variation of this average in each sub-collection is small (from 4.8 to 5.3). If we remove the stopwords, the average length of a word increases to little more than 6 letters (see Table 1). If we take the average length in the vocabulary, the value is higher (between 7 and 8 as shown in Table 1). This defines the total space needed for the vocabulary. Figure 2 shows how the average length of the vocabulary words and the text words evolve as the filtered text grows for the WSJ collection.

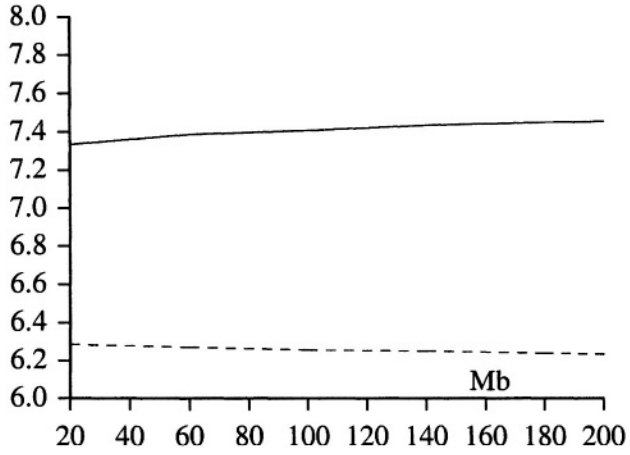

Figure 2. Average length of the words in the vocabulary (solid line) and in the text (dashed line).

Heaps' law implies that the length of the words of the vocabulary increase logarithmically as the text size increases, and longer and longer words should appear as the text grows. This is because if for large n there are n^{β} different words, then their average length must be $\log_{\sigma}(n^{\beta}) = \beta \log_{\sigma} n$ at least (counting once each different word). However, the average length of the words in the overall text should be constant because shorter words are common enough (e.g.

stopwords). Our experiment of Figure 2 shows that the length is almost constant, although decreases slowly. This balance between short and long words, such that the average word length remains constant, has been noticed many times in different contexts. It can be explained by a simple finite-state model where the separators have a fixed probability of occurrence, since this implies that the average word length is one over that probability. Such a model is considered in [Miller, Newman & Friedman, 1957; Miller, Newman & Friedman, 1958], where: (a) the space character has probability close to 0.2, (b) the space character cannot appear twice subsequently, and (c) there are 26 letters.

1.3 Relating the Heaps' and Zipf's Law

In this section we relate and explain the two main empirical laws: Heaps' and Zipf's. In particular, if both are valid, then a simple relation between their parameters holds. This result is from [Baeza-Yates & Navarro,1999].

Assume that the least frequent word appears $O(1)$ times in the text (this is more than reasonable in practice, since a large number of words appear only once). Since there are $\Theta(n^\beta)$ different words, then the least frequent word has rank $i = \Theta(n^\beta)$. The number of occurrences of this word is, by Zipf's law,

$$\frac{n}{i^\theta H_V(\theta)} = \Theta\left(\frac{n}{n^{\beta\theta} H_V(\theta)}\right)$$

and this must be $O(1)$. This implies that, as n grows, $\beta = 1/\theta$. This equality may not hold exactly for real collections. This is because the relation is asymptotical and hence is valid for sufficiently large n, and because Heaps' and Zipf's rules are approximations. Considering each collection of TREC-2 separately, $\beta\theta$ is between 0.80 and 1.00. Table 1 shows specific values for K and β (Heaps' law) and θ (Zipf's law), without filtering the text. Notice that $1/\beta$ is always larger than θ. On the other hand, for our Web data, the match is almost perfect, as $\beta\theta \approx 1$.

Text	K	β	$1/\beta$	θ	*Len. (text)*	*Len. (vocab.)*	*Eq. σ*
AP	26.8	0.46	2.17	1.87	6.328	8.012	15.44
DOE	10.8	0.52	1.92	1.70	6.429	8.423	15.41
FR	13.2	0.48	2.08	1.94	6.096	6.827	15.64
WSJ	43.5	0.43	2.33	1.87	6.233	7.453	15.37
ZIFF	11.3	0.51	1.96	1.79	6.441	7.181	15.79

Table 1. Experimental results for the parameters of Heaps' and Zipf's laws, as well as the average length of words and equivalent alphabet size.

The relation of the Heapst' and Zipt's Laws is mentioned in a line of a paper by Mandelbrot [Mandelbrot, 1954], but no proof is given. In the Appendix

we give a non trivial proof based in a simple finite-state model for generating words.

1.4 Modeling a Document Collection

The Heaps' and Zipf's laws are also valid for whole collections. In particular, the vocabulary should grow faster (larger β) and the word distribution could be more biased (larger θ). That would match better the relation $\beta\theta = 1$, which in TREC-2 is less than 1. However, there are no experiments on large collections to measure these parameters (for example, in the Web). In addition, as the total text size grows, the predictions of these models become more accurate.

1.4.1 Word Distribution Within Documents

The next issue is the distribution of words in the documents of a collection. The simplest assumption is that each word is uniformly distributed in the text. However, this rule is not always true in practice, since words tend to appear repeated in small areas of the text (locality of reference). A uniform distribution in the text is a pessimistic assumption since it implies that queries appear in more documents. However, a uniform distribution can have different interpretations. For example, we could say that each word appears the same number of times in every document. However, this is not fair if the document sizes are different. In that case, we should have occurrences proportional to the document size. A better model is to use a Binomial distribution. That is, if f is the frequency of a word in a set of D documents with n words overall, the probability of finding the word k times in a document having w words ($w \leq f$) is

$$Pr(k, n, w) = \binom{w}{k} p^k (1-p)^{w-k}, \quad p = \frac{f}{n}$$

For large w, we can use the Poisson approximation $Pr(k, n, w) = \frac{\lambda^k}{k!} \mathrm{e}^{-\lambda}$ with $\lambda = w\, f/n$. Some people apply these formulas using the average for all the documents, which is unfair if document sizes are very different.

A model that approximates better what is seen in real text collections is to consider a negative binomial distribution, which says that the fraction of documents containing a word k times is

$$F(k) = \binom{\alpha + k - 1}{k} p^k (1+p)^{-\alpha-k}$$

where p and α are parameters that depend on the word and the document collection. Notice that $F(k) = D\ Pr(k, n, w)$ if we use $w = n/D$, the average number of words per document, so this distribution also has the problem of being unfair if document sizes are different. For example, for the Brown Corpus

[Francis & Kucera, 1982] and the word "said", we have $p = 9.24$ and $\alpha = 0.42$ [Church & Gale, 1995]. The latter reference gives other models derived from a Poisson distribution. Another model related to Poisson which takes in account locality of reference is the Clustering Model [Thom & Zobel, 1992].

1.4.2 Distribution of Document Sizes

Static databases will have a fixed document size distribution. Moreover, depending on the database format, the distribution can be very simple. However, this is very different for databases that grow fast and in a chaotic manner, such as the Web. The results that we present next are based in the Web.

The document sizes are self-similar [Crovella & Bestavros, 1996], that is, the probability distribution remains unchanged if we change the size scale. The same behavior appears in Web traffic. This can be modeled by two different distributions. The main body of the distribution follows a Logarithmic Normal curve, such that the probability of finding a Web page of x bytes is given by

$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} \; e^{-(\ln x - \mu)^2/2\sigma^2}$$

where the average (μ) and standard deviation (σ) are 9.357 and 1.318, respectively [Barford & Crovella, 1998]. See figure of an example in 3 (from [Crovella & Bestavros, 1996]).

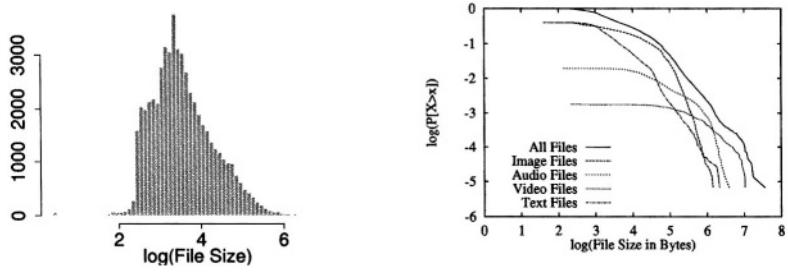

Figure 3. Left: Distribution for all file sizes. Right: Right tail distribution for different file types. All logarithms are in base 10. (Both figures are courtesy of Mark Crovella).

The right tail of the distribution is "heavy-tailed". That is, the majority of documents are small, but there is a non trivial number of large documents. This is intuitive for image or video files, but it is also true for textual pages. A good fit is obtained with the Pareto distribution, that says that the probability of finding a Web page of x bytes is

$$p(x) = \frac{\lambda k^{\lambda}}{x^{1+\lambda}}$$

for $x \geq k$, and zero otherwise. The cumulative distribution is

$$F(x) = 1 - \left(\frac{k}{x}\right)^{\lambda}$$

where k and λ are constants dependent on the particular collection [Barford & Crovella, 1998]. The parameter k is the minimum document size, and λ is about 1.36 for textual data, being smaller for images and other binary formats [Crovella & Bestavros, 1996; Willinger & Paxson, 1998] (see the right side of Figure 3). Taking all Web documents into account, using $k = 9.3\text{Kb}$, we get $\lambda = 1.1$, and 93% of all the files have a size below this value. The parameters of these distributions were obtained from a sample of more than 50 thousand Web pages requested by several users in a period of two months. Recent results show that these distributions are still valid [Barford et al, 1999], but the exact parameters for the distribution of all textual documents is not known, although average page size is estimated in 6Kb including markup (which is traditionally not indexed).

1.5 Models for Queries and Answers

1.5.1 Motivation

When analyzing or simulating text retrieval algorithms, a recurrent problem is how to model the queries. The best solution is to use real users or to extract information from query logs. There are a few surveys and analyses of query logs with respect to the usage of Web search engines [Pollock & Hockley, 1997; Jensen et al, 1998; Silverstein et al, 1998]. The later reference is the study of 285 million AltaVista user sessions containing 575 million queries. Table 2 gives some results from that study, done in September of 1998. Another recent study on Excite, shows similar statistics, and also the queries topics [Spink et al, 2002]. Nevertheless, these studies give little information about the exact distribution of the queries. In the following we give simple models to select a random query and the corresponding average number of answers that will be retrieved. We consider exact queries and approximate queries. An approximate query finds a word allowing up to k errors, where we count the minimal number of insertions, deletions, and substitutions.

1.5.2 Random Queries

As half of the text words are stopwords, and they are not typical user queries, stopwords are not considered. The simplest assumption is that user queries are distributed uniformly in the vocabulary, i.e. every word in the vocabulary can be searched with the same probability. This is not true in practice, since unfrequent words are searched with higher probability. On the other hand,

Measure	*Average value*	*Range*
Number of words	2.35	0 to 393
Number of operators	0.41	0 to 958
Repetitions of each query	3.97	1 to 1.5 million

Table 2. Queries on the Web: average number of words, Boolean operations, and query repetitions.

approximate searching makes this distribution more uniform, since unfrequent words may match with k errors with other words, with little relation to the frequencies of the matched words. In general, however, the assumption of uniform distribution in the vocabulary is pessimistic, at least because a match is always found.

Looking at the results in the AltaVista log analysis [Silverstein et al, 1998], there are some queries much more popular than others and the range is quite large. Hence, a better model would be to consider that the queries also follow a Zipf's like distribution, perhaps with θ larger than 2 (the log data is not available to fit the best value). However, the actual frequency order of the words in the queries is completely different from the words in the text (for example, "sex" and "xxx" appear between the top most frequent word queries), which makes a formal analysis very difficult. An open problem, which is related to the models of term distribution in documents, is whether the distribution for query terms appearing in a collection of documents is similar to that of document terms. This is very important as these two distributions are the base for relevance ranking in the vector model [Baeza-Yates & Ribeiro-Neto, 1999]. Recent results show that although queries also follow a Zipf distribution (with parameter θ from 1.24 to 1.42 [Baeza-Yates & Castillo, 2001; Baeza-Yates & Saint-Jean, 2002]), the correlation to the word distribution of the text is low (0.2) [Baeza-Yates & Saint-Jean, 2002]. This implies that choosing queries at random from the vocabulary is reasonable and even pessimistic.

Previous work by DeFazio [DeFazio, 1993] divided the query vocabulary in three segments: high (words representing the most used 90% of the queries), moderate (next 5% of the queries), and low use (words representing the least used 5% of the queries). Words are then generated by first randomly choosing the segment, the randomly picking a token within that segment. Queries are formed by choosing randomly one to 50 words. According to currently available data, real queries are much shorter, and the generation algorithm does not produce the original query distribution. Another problem is that the query vocabulary must be known to use this model. However, in our model, we can generate queries from the text collection.

1.5.3 Number of Answers

Now we analyze the expected number of answers that will be obtained using the simple model of the previous section. For a simple word search, we will find just one entry in the vocabulary matching it. Using Heaps' law, the average number of occurrences of each word in the text is $n/V = \Theta(n^{1-\beta})$. Hence, the average number of occurrences of the query in the text is $O(n^{1-\beta})$. This fact is surprising, since one can think in the process of traversing the text word by word, where each word of the vocabulary has a fixed probability of being the next text word. Under this model the number of matching words is a fixed proportion of the text size (this is equivalent to say that a word of length ℓ should appear about $O(n/\sigma^{\ell})$ times). The fact that this is not the case (demonstrated experimentally later) shows that this model does not really hold on natural language text.

The root of this fact is not in that a given word does not appear with a fixed probability. Indeed, the Heaps' law is compatible with a model where each word appears at fixed text intervals. For instance, imagine that Zipf's law stated that the i-th word appeared $n/2^i$ times. Then, the first word could appear in all the odd positions, the second word in all the positions multiple of 4 plus 2, the third word in all the multiples of 8 plus 4, and so on. The real reason for the sublinearity is that, as the text grows, there are more words, and one selects randomly among them. Asymptotically, this means that the length of the vocabulary words must be $\ell = \Omega(\log n)$, and therefore, as the text grows, we search on average longer and longer words. This allows that even in the model where there are n/σ^{ℓ} matches, this number is indeed $o(n)$ [Navarro, 1998]. Note that this means that users search for longer words when they query larger text collections, which seems awkward but may be true, as the queries are related to the vocabulary of the collection.

How many words of the vocabulary will match an approximate query? In principle, there is a constant bound to the number of distinct words which match a given query with k errors, and therefore we can say that $O(1)$ words in the vocabulary match the query. However, not all those words will appear in the vocabulary. Instead, while the vocabulary size increases, the number of matching words that appear increases too, at a lower rate. This is the same phenomenon observed in the size of the vocabulary. In theory, the total number of words is finite and therefore $V = O(1)$, but in practice that limit is never reached and the model $V = O(n^{\beta})$ describes reality much better. We show experimentally that a good model for the number of matching words in the vocabulary is $O(n^{\nu})$ (with $\nu < \beta$). Hence, the average number of occurrences of the query in the text is $O(n^{1-\beta+\nu})$ [Baeza-Yates & Navarro, 1999].

1.5.4 Experiments

We present in this section empirical evidence supporting our previous statements. We first measure V, the number of words in the vocabulary in terms of n (the text size). Figure 4 (left side) shows the growth of the vocabulary. Using least squares we fit the curve $V = 78.81n^{0.40}$. The relative error is very small (0.84%). Therefore, $\beta = 0.4$ for the WSJ collection.

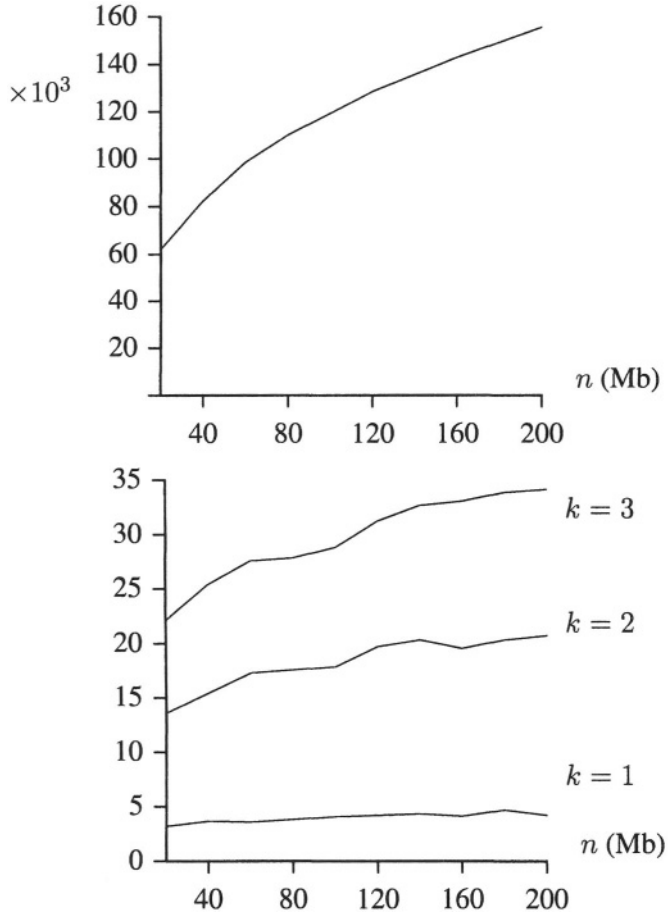

Figure 4. Vocabulary tests for the WSJ collection. On the left, the number of words in the vocabulary. On the right, number of matching words in the vocabulary.

We measure now the number of words that match a given pattern in the vocabulary. For each text size, we select words at random from the vocabulary allowing repetitions. In fact, not all user queries are found in the vocabulary in

practice, which reduces the number of matches. Hence, this test is pessimistic in that sense.

We test $k = 1, 2$ and 3 errors. To avoid taking into account queries with very low precision (e.g. searching a 3-letter word with 2 errors may match too many words), we impose limits on the length of words selected: only words of length 4 or more are searched with one error, length 6 or more with two errors, and 8 or more with three errors.

We perform a number of queries which is large enough to ensure a relative error smaller than 5% with a 95% confidence interval. Figure 4 (right side) shows the results. We use least squares to fit the curves $0.31n^{0.14}$ for $k = 1$, $0.61n^{0.18}$ for $k = 2$ and $0.88n^{0.19}$ for $k = 3$. In all cases the relative error of the approximation is under 4%. The exponents are the ν values mentioned later in this article. One possible model for ν is $\beta(1 - e^{-\alpha k})$, because for $k = 0$ we have $\nu = 0$ and when $k \to \infty$, $\nu \to \beta$, as expected.

We could reduce the variance in the experiments by selecting once the set of queries from the index of the first 20 Mb. However, our experiments have shown that this is not a good policy. The reason is that the first 20 Mb will contain almost all common words, whose occurrence lists grow faster than the average. Most uncommon words will not be included. Therefore, the result would be unfair, making the results to look linear when they are in fact sublinear.

1.6 Application: Inverted Files for the Web

1.6.1 Motivation

Web search engines currently available use inverted files that reference Web pages [Baeza-Yates & Ribeiro-Neto, 1999]. So, reference pointers should have as many bits as needed to reference all Web pages (currently, about 3 billion). The number and size of pointers is directly related with the space overhead of the inverted file. For the whole Web, this implies at least 600 GB. Some search engines also index word locations, so the space needed is increased. One way to reduce the size of the index is to use fixed logical blocks as reference units, trading the reduction of space obtained with an extra cost at search time. The block mechanism is a logical layer and the files do not need to be physically split or concatenated. In which follows we explain this technique in more detail.

Assume that the text is logically divided into "blocks". The index stores all the different words of the text (the vocabulary). For each word, the list of the blocks where the word appears is kept. We call b the size of the blocks and r the number of blocks, so that $n \approx rb$. The exact organization is shown in Figure 5. This idea was first used in *Glimpse* [Manber & Sun Wu, 1994].

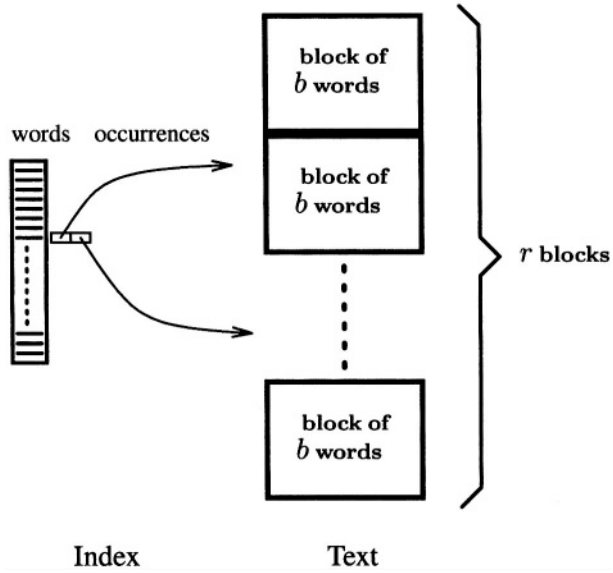

Figure 5. The block-addressing indexing scheme.

At this point the reader may wonder which is the advantage of pointing to artificial blocks instead of pointing to documents (or files), this way following the natural divisions of the text collection. If we consider the case of simple queries (say, one word), where we are required to return only the list of matching documents, then pointing to documents is a very adequate choice. Moreover, as we see later, it may reduce space requirements with respect to using blocks of the same size. Moreover, if we pack many short documents in a logical block, we will have to traverse the matching blocks (even for these simple queries) to determine which documents inside the block actually matched.

However, consider the case where we are required to deliver the exact positions which match a pattern. In this case we need to sequentially traverse the matching blocks or documents to find the exact positions. Moreover, in some types of queries such as phrases or proximity queries, the index can only tell that two words are in the same block, and we need to traverse it in order to determine if they form a phrase.

In this case, pointing to documents of different sizes is not a good idea because larger documents are searched with higher probability and searching them costs more. In fact, the expected cost of the search is directly related to the variance in the size of the pointed documents. This suggests that if the documents have different sizes it may be a good idea to (logically) partition

large documents into blocks and to put together small documents, such that blocks of the same size are used.

In [Baeza-Yates & Navarro,1999], we show analytically and experimentally that using fixed size blocks it is possible to have a sublinear-size index with sublinear search times, even for approximate word queries. A practical example shows that the index can be $O(n^{0.94})$ in space and in retrieval time for approximate queries with at most two errors. For exact queries the exponent lowers to 0.85. This is a very important analytical result which is experimentally validated and makes a very good case for the practical use of this kind of index. Moreover, these indices are amenable to compression. Block-addressing indices can be reduced to 10% of their original size [Bell et al, 1993], and the first works on searching the text blocks directly in their compressed form are just appearing [Moura et al, 1998a; Moura et al, 1998] with very good performance in time *and* space.

Resorting to sequential searching to solve a query may seem unrealistic for current Web search engine architectures, but makes perfect sense in a near future when a remote access could be as fast as a local access. Another practical scenario is a distributed architecture where each logical block is a part of a Web server or a small set of Web servers locally connected, sharing a local index.

As explained before, pointing to documents instead of blocks may or may not be convenient in terms of query times. We analyze now the space and later the time requirements when we point to Web pages or to logical blocks of fixed size. Recall that the distribution has a main body which is log-normal (that we approximate with a uniform distribution) and a Pareto tail.

We start by relating the free parameters of the distribution. We call C the cut point between both distributions and f the fraction of documents smaller than C. Since Then the integral over the tail (from C to infinity) must be $(1 - f)$, which implies that $k = (1 - f)^{1/\lambda}C$. We also need to know the value of the distribution in the uniform part, which we call t, and it holds $tC = f$. For the occurrences of a word inside a document we use the uniform distribution taking into account the size of the document.

1.6.2 Space Overhead

As the Heaps' law states that a document with x words has x^β different words, we have that each new document of size x added to the collection will insert x^β new references to the lists of occurrences (since each different word of each different document has an entry in the index). Hence, an index of r blocks of size b takes $O(rb^\beta)$ space. If, on the other hand, we consider the Web document size distribution, we have that the average number of new entries in

the occurrence list per document is

$$\int_0^\infty p(x)x^\beta dx = \int_0^C tx^\beta dx + \int_C^\infty \lambda k^\lambda x^{\beta-\lambda-1} = \frac{tC^{1+\beta}}{1+\beta} + \frac{\lambda k^\lambda}{(\lambda-\beta)C^{\lambda-\beta}} \tag{6.1}$$

where $p(x)$ was defined in Section 1.4.2.

To determine the total size of the collection, we consider that r documents exist, whose average length is b^* given by

$$b^* = \int_0^\infty p(x)x dx = \frac{tC^2}{2} + \frac{\lambda k^\lambda}{(\lambda-1)C^{\lambda-1}} \tag{6.2}$$

and therefore the total size of the collection is

$$n = rb^* = r\left(\frac{tC^2}{2} + \frac{\lambda k^\lambda}{(\lambda-1)C^{\lambda-1}}\right) \tag{6.3}$$

The final size of the occurrence lists is (using Eq. (6.1))

$$r\left(\frac{tC^{1+\beta}}{1+\beta} + \frac{\lambda k^\lambda}{(\lambda-\beta)C^{\lambda-\beta}}\right) \tag{6.4}$$

We consider now what happens if we take the average document length and use blocks of that fixed size (splitting long documents and putting short documents together as explained). In this case, the size of the vocabulary is $O(n^\beta)$ as before, and we assume that each block is of a fixed size $b = zb^*$. We have introduced a constant z to control the size of our blocks. In particular, if we use the same number of blocks as Web pages, then $z = 1$. Then the size of the lists of occurrences is

$$(r/z)b^\beta = \frac{r}{z^{1-\beta}}\left(\frac{tC^2}{2} + \frac{\lambda k^\lambda}{(\lambda-1)C^{\lambda-1}}\right)^\beta$$

(using Eq. (6.3)). Now, if we divide the space taken by the index of documents by the space taken by the index of blocks (using the previous equation and Eq. (6.4)), the ratio is

$$\frac{\text{document index}}{\text{block index}} = \frac{r\left(\frac{tC^{1+\beta}}{1+\beta} + \frac{\lambda k^\lambda}{(\lambda-\beta)C^{\lambda-\beta}}\right)}{\frac{r}{z^{1-\beta}}\left(\frac{tC^2}{2} + \frac{\lambda k^\lambda}{(\lambda-1)C^{\lambda-1}}\right)^\beta} = z^{1-\beta}\frac{\frac{tC}{1+\beta} + \frac{\lambda k^\lambda}{(\lambda-\beta)C^\lambda}}{\left(\frac{tC}{2} + \frac{\lambda k^\lambda}{(\lambda-1)C^\lambda}\right)^\beta}$$

$$= z^{1-\beta}\frac{\frac{f}{\beta+1} + \frac{\lambda(1-f)}{\lambda-\beta}}{\left(\frac{f}{2} + \frac{\lambda(1-f)}{\lambda-1}\right)^\beta} \tag{6.5}$$

which is independent of r, n, k and C; and is about 85% for $z = 1$, $f = 0.93$ and $\beta = 0.4..0.6$. We approximated $f = 0.93$, which corresponds to all the Web pages, because the value for textual pages is not known. This shows that indexing documents yields an index which takes 85% of the space of a block addressing index, if we have as many blocks as documents. Figure 6 shows the ratio as a function of λ and β. As it can be seen, the result varies slowly with β, while it depends more on λ (tending to 1 as the document size distribution is more uniform).

The fact that the ratio varies so slowly with β is good because we already know that the β value is quite different for small documents. As a curiosity, see that if the documents sizes were uniformly distributed in all the range (that is, letting $f \to 1$) the ratio would become $2^{\beta}/(1+\beta)$, which is close to 0.94 for intermediate β values. On the other hand, letting $f \to 0$ (as in the simplified model [Crovella & Bestavros, 1996]) we have a ratio near 0.83. As another curiosity, notice that there is a β value which gives the minimum ratio for document versus block index (that is, the worst behavior for the block index). This is $\beta = .57$ for $z = 1$, quite close to the real values (0.63 in our Web experiments).

If we want to have the same space overhead for the document and the block indices, we simply make the expression of Eq. (6.5) equal to 1 and obtain $z \approx 1.27..1.48$ for $\beta = 0.4..0.6$, that is, we need to make the blocks larger than the average of the Web pages. This translates into worse search times. By paying more at search time we can obtain smaller indices (letting z grow over 1.48).

1.6.3 Retrieval Time

We analyze the case of approximate queries, given that for exact queries the result is the same by using $\nu = 0$. The probability of a given word to be selected by a query is $O(n^{\nu-\beta})$. The probability that none of the words in a block is selected is therefore $(1 - O(n^{\nu-\beta}))^b$. The total amount of work of an index of fixed blocks is obtained by multiplying the number of blocks (r) times the work to do per selected block (b) times the probability that some word in the block is selected. This is

$$\Theta\left(rb\left(1-\left(1-n^{\nu-\beta}\right)^b\right)\right) = \Theta\left(n\left(1-e^{-\Theta(b/n^{\beta-\nu})}\right)\right) \qquad (6.6)$$

where for the last step we used that $(1-x)^y = e^{y\ln(1-x)} = e^{y(-x+O(x^2))} = \Theta(e^{-\Theta(yx)})$ provided $x = o(1)$.

We are interested in determining in which cases the above formula is sublinear in n. Expressions of the form "$1 - e^{-x}$" are $O(x)$ whenever $x = o(1)$ (since $e^{-x} = 1 - x + O(x^2)$). On the other hand, if $x = \Omega(1)$, then e^{-x} is far away from 1, and therefore "$1 - e^{-x}$" is $\Omega(1)$.

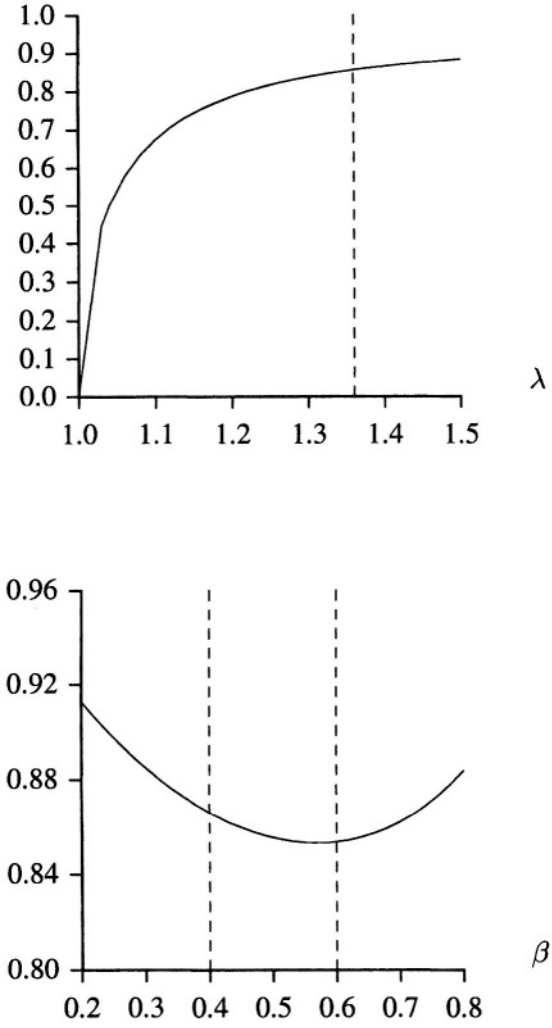

Figure 6. On the left, ratio between block and document index as a function of λ for fixed $\beta = 0.5$ (the dashed line shows the actual λ value for the Web). On the right, the same as a function of β for $\lambda = 1.36$ (the dashed lines enclose the typical β values). In both cases we use $f = 0.93$ and the standard $z = 1$.

For the search cost to be sublinear, it is thus necessary that $b = o(n^{\beta-\nu})$. When this condition holds, we derive from Eq. (6.6) that

$$Time = \Theta\left(n^{\beta} + bn^{1-\beta+\nu}\right) \tag{6.7}$$

We consider now the case of an index that references Web pages. As we have shown, if a block has size x then the probability that it has to be traversed is $(1-e^{-\Theta(x/n^{\beta-\nu})})$. We multiply this by the cost x to traverse it and integrate over all the possible sizes, so as to obtain its expected traversal cost (recall Eq. (6.6))

$$\int_k^\infty x(1-e^{-\Theta(x/n^{\beta-\nu})})p(x)dx$$

which we cannot solve. However, we can separate the integral in two parts, (*a*) $x = o(n^{\beta-\nu})$ and (*b*) $x = \Omega(n^{\beta-\nu})$. In the first case the traversal probability is $O(x/n^{\beta-\nu})$ and in the second case it is $\Omega(1)$. Splitting the integral in two parts and multiplying the result by $r = n/b^*$ we obtain the total amount of work:

$$\Theta\left(\frac{n}{\frac{1}{2}+\frac{\lambda}{\lambda-1}}\left(\left(\frac{C}{3}-\frac{\lambda f}{2-\lambda}\right)n^{\nu-\beta}+\frac{\lambda C^{\lambda-1}}{(2-\lambda)(\lambda-1)}n^{(\nu-\beta)(\lambda-1)}\right)\right)$$

where since this is an asymptotic analysis we have considered $C = o(n^{\beta-\nu})$, as C is constant.

On the other hand, if we used blocks of fixed size, the time complexity (using Eq. (6.7)) would be $O(bn^{1-\beta+\nu})$, where $b = zb^*$. The ratio between both search times is

$$\frac{\text{doc. index traversal}}{\text{block index traversal}} = \Theta\left(n^{(\beta-\nu)(2-\lambda)}\right)$$

which shows that the document index would be asymptotically slower than a block index as the text collection grows. In practice, the ratio is between $O(n^{0.2})$ and $O(n^{0.4})$. The value of z is not important here since it is a constant, but notice that k is usually quite large, which favors the block index.

1.7 Concluding Remarks

The models presented here are common to other processes related to human behavior [Zipf, 1949] and algorithms. For example, a Zipf like distribution also appears for the popularity of Web pages with $\theta < 1$ [Barford et al, 1999]. On the other hand, the phenomenon of sublinear vocabulary growing is not exclusive of natural language words. It appears as well in many other scenarios, such as the number of different words in the vocabulary that match a given query allowing errors as shown in Section 5, the number of states of the deterministic automaton that recognizes a string allowing errors [Navarro, 1998], and the number of suffix tree nodes traversed to solve an approximate query [Navarro & Baeza-Yates, 1999]. We believe that in fact the finite state model for generating words used in Section 3 could be changed for a more general