Similarity Search with Set Intersection as a Distance Measure

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

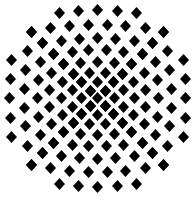
Benjamin Sascha Hoffmann

aus Böblingen

Hauptberichter: Prof. Dr. rer. nat. habil. Volker Diekert

Mitberichter: Prof. Dr. rer. nat. habil. Ulrich Hertrampf

Tag der mündlichen Prüfung: 25. März 2010



Universität Stuttgart
Institut für Formale Methoden der Informatik (FMI)
2010

Abstract

This thesis deals with a fundamental algorithmic problem. Given a database of sets and a query set, we want to determine a set from the database that has a maximal intersection with the query set. It is allowed to preprocess the database so that queries can be answered efficiently.

We solve the approximate version of this problem. We investigate two randomized input models which are derived from real inputs. We present a deterministic algorithm for each of them. Under the assumption that the database and the query set follow one of these models, the corresponding algorithm determines with high probability a set from the database that has no maximal intersection with the query set, but an intersection that achieves a large proportion of the maximal size. Depending on the model, the query time is either quasi-linear in the sum of the database size and the number of different elements from all sets, or it is polylogarithmic in the database size. Thus, both algorithms are significantly faster than a naive algorithm intersecting the query set with each single database set.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit einem elementaren Problem aus dem Gebiet der Algorithmentheorie. Sei $\mathcal{W} = \{w_1, \ldots, w_m\}$ eine endliche Menge. Das **Maximaler-Schnitt-Problem** ist wie folgt definiert: Gegeben sei eine Datenbank von n endlichen Mengen $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ und eine endliche Anfragemenge $q \subseteq \mathcal{W}$. Das Ziel ist, möglichst effizient eine Menge $d \in \mathcal{D}$ zu bestimmen, die mit der Anfrage q einen Schnitt maximaler Größe besitzt ($\forall d' \in \mathcal{D}$: $|q \cap d'| \leq |q \cap d|$). Dabei ist es erlaubt, die Datenbank vorzuverarbeiten.

Im Allgemeinen ist das Bestimmen einer solchen Menge zeitaufwändig. Daher werden wir zwei aus der Praxis hergeleitete *randomisierte Eingabemodelle* untersuchen und zeigen, dass beide Modelle eine sogenannte Schwellenwert-Eigenschaft besitzen. Wir werden sehen, dass diese Eigenschaft eine effiziente Lösung des Problems ermöglicht. Wir zeigen, dass wenn die Eingabe sich gemäß einem dieser Modelle verhält, dann existiert ein deterministischer Algorithmus, welcher mit hoher Wahrscheinlichkeit eine k-approximierte Antwort findet (eine k-approximierte Antwort ist eine Menge $d \in \mathcal{D}$, so dass $|q \cap d'| \leq k \cdot |q \cap d|$ für alle $d' \in \mathcal{D}$ gilt). Man beachte, dass hierbei die Wahrscheinlichkeit über die gemäß dem Modell randomisierte Eingabe gebildet wird, und nicht über Zufallsbits des Algorithmus.

Beide Eingabemodelle basieren darauf, dass die Elemente der Anfragemenge bezüglich einer Ordnung, die durch die zufällige[1] Datenbank gegeben ist, sortiert sind. Wir unterscheiden zwei verschiedene Arten von Übereinstimmungen. Eine r-Übereinstimmung ist eine Menge der Datenbank, die mindestens r Elemente der Anfragemenge enthält. Eine r-Präfixübereinstimmung ist eine Menge der Datenbank, die mindestens die ersten r Elemente der Anfragemenge enthält (man beachte, dass eine r-Präfixübereinstimmung immer auch eine r-Übereinstimmung ist, wohingegen die umgekehrte Richtung

[1] gemäß des Eingabemodells

nicht stimmt). Wie wir zeigen werden, verhalten sich beide Übereinstimmungen wie folgt: Die Wahrscheinlichkeit, dass eine Übereinstimmung exisitiert, ist für kleines r nahe bei Eins, und fällt ab einem gewissen „Schwellenwert“ schnell gegen Null. Entscheidend dabei ist, dass die Schwellenwerte beider Übereinstimmungen nahe beieinander liegen. Dies ermöglicht es, mit hoher Wahrscheinlichkeit eine approximierte Antwort in quasilinearer Zeit in der Summe von n und m beziehungweise in polylogarithmischer Zeit in n zu bestimmen. Im Detail sind die Ergebnisse für beide Modelle wie folgt (Teile wurden in [HLN07] und [HLLN09] veröffentlicht):

Zipfsches Modell. Eingaben in diesem Modell verhalten sich gemäß dem *Zipfschen Gesetz*. Das Zipfsche Gesetz tritt hauptsächlich in natürlichsprachlichen Texten auf und besagt, dass die absolute Häufigkeit eines Wortes umgekehrt proportional zu seinem Rang ist. Indem wir zeigen, dass im Zipfschen Modell die Schwellenwerte beider Übereinstimmungen nahe beieinander liegen, können wir einen deterministischen Algorithmus angeben, der eine $\frac{1+E(\varepsilon,n)}{1-\Delta(\delta,n)}$-approximierte Antwort mit einer von ε, δ und n abhängigen Wahrscheinlichkeit liefert. Dabei sind $\varepsilon, \delta > 0$ und es gilt $E(\varepsilon, n) \to \varepsilon$, $\Delta(\delta, n) \to \delta$ für $n \to \infty$. Die Wahrscheinlichkeit tendiert gegen Eins für $n \to \infty$. Die Vorverarbeitungszeit dieses Algorithmus liegt in $\tilde{O}(nm)$, die Anfragezeit beträgt $\tilde{O}(\log m + n)$ im Mittel und $\tilde{O}(m + n)$ im schlechtesten Fall. Der Platzbedarf ist in $n^{1+o(1)}$. Das Zipfsche Modell verallgemeinern wir dahingehend, dass wir Eingaben betrachten, die dem Gesetz von Mandelbrot folgen (welches eine Verallgemeinerung des Zipfschen Gesetzes darstellt). Wir zeigen, dass sich obiger Algorithmus auf alle Eingaben, die dem verallgemeinertem Modell folgen, anwenden lässt und dabei vergleichbare Resultate liefert.

Hierarchische Schemata. Hierbei nehmen wir an, dass die Elemente der Menge $\mathcal{W}$ in einem baumartigem Schema hierarchisch angeordnet sind, und zwar so, dass jedes Element sich einem festen Level zuordnen lässt. Wir zeigen, dass jedes hierarchische Schema die Schwellenwert-Eigenschaft besitzt und diese einen deterministischen Algorithmus ermöglicht, der mit hoher Wahrscheinlichkeit eine $2 \cdot \frac{1+\varepsilon_n}{1-\varepsilon_n}$-approximierte Antwort liefert. Dabei ist $\varepsilon_n > 0$ und die Wahrscheinlichkeit tendiert für $n \to \infty$ gegen Eins. Der Algorithmus hat $\tilde{O}(n)$ Vorverarbeitungs- und $O(\log^2 n)$ Anfragezeit. Der benötigte Platz liegt in $\tilde{O}(n)$.

Neben diesen zwei Modellen und zugehörigen Algorithmen zeigen wir, dass ein randomisierter Algorithmus existiert, der ebenfalls eine approx-

imierte Antwort liefert. Dieser Algorithmus wählt eine zufällige Stichprobe aus der Datenbank aus und bestimmt in dieser Stichprobe eine optimale Antwort (d.h. eine Menge, die unter allen Mengen der Stichprobe maximale Schnittgröße mit der Anfrage hat). Die Qualität dieser Antwort lässt sich abschätzen indem wir zeigen, dass die erwartete Anzahl an verbleibenden Mengen der Datenbank, die einen größeren Schnitt mit der Anfrage besitzen, höchstens $\frac{n-r}{r+1}$ beträgt. Die Anfragezeit des Algorithmus liegt in $O(m \log m)$ und ist somit unabhängig von der Datenbankgröße n.

Wir komplettieren die Untersuchung des Problems durch die Herleitung von unteren Schranken im Cell-Probe-Modell. Angenommen, es gilt $m \in \omega(\log n) \cap n^{o(1)}$. Dann besitzt jeder Algorithmus (deterministisch oder randomisiert, mit beidseitigem Fehler), der polynomiell in n und m viel Platz benötigt, eine Anfragezeit von $\Omega(m/\log n)$.

Acknowledgments

First of all, I would like to express my gratitude to my advisor, Prof. Dr. Volker Diekert. He helped me to focus on the important problems and was always ready to listen to my questions and ideas. His continual interest, support, and motivation over all my years in his group have made this thesis possible. Furthermore, I would like to thank Prof. Dr. Ulrich Hertrampf for his time and valuable feedback as my second advisor.

I would also like to thank all members of our department for their help and support. In particular, I want to thank Jürn Laun, Manfred Kufleitner, and Dirk Nowotka for valuable discussions and Horst Prote for technical assistance.

Finally, I would like to thank my family and Karin for their great support and patience.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, technological advance has made it possible to accumulate huge amounts of data. According to the 2008 annual review of Thomson Reuters, every day 15 petabytes of new data are created [Reu08]. As consequence, algorithms which are capable of handling huge data sets are required. This thesis deals with a fundamental computational problem appearing in this context. Consider a database of sets and a single set, called the query set. How can one quickly find a member of the database that has a maximal intersection with the query set? Such maximal intersection queries arise in a wide range of applications, including text clustering, web search, recommendation systems, and the distribution of online advertisements. In general, maximal intersection queries are computationally expensive. We investigate two well-motivated input distributions which lead to efficient algorithms. Both algorithms are based on the observation that each distribution exhibits a threshold phenomenon on the probabilities of intersecting sets.

1.1 Structure of the Thesis

The structure of the thesis is as follows:

In **Chapter 1** we state the *maximal intersection* problem in detail, give an overview of the results and related work, and introduce the notation and basic facts we will use.

In **Chapter 2** we derive lower bounds for the maximal intersection problem from known lower bounds for the nearest neighbor search problem in the Hamming cube. The bounds are stated in the cell probe model.

In **Chapter 3** we present a randomized algorithm that returns an approximate answer to the maximal intersection problem. It is based on the sampling lemma and has a running time which does not depend on the database

size.

Our main results are presented in **Chapter 4** and **Chapter 5**. We investigate two input distributions and construct a deterministic algorithm for each distribution solving an approximate version of the maximal intersection problem. We show that under the assumption that the input behaves according to one of these distributions, the time complexity and accuracy of the corresponding algorithm are reasonably good with high probability. Here we use the probability over the input distribution, not over random choices of the algorithm. Parts of these results have been published in [HLN07] and [HLLN09]. The former one is joint work with Yury Lifshits and Dirk Nowotka, while the latter one is joint work with Mikhail Lifshits, Yury Lifshits, and Dirk Nowotka.

Chapter 6 shows experimental results of the algorithm described in Chapter 4.

We give a conclusion and point out open questions in **Chapter 7**.

1.2 Problem Statement

Let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a finite set, sometimes called the *vocabulary*. The *maximal intersection (MI)* problem is the following:

Input: A database of n finite sets $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$.

Query: Given a finite query set $q \subseteq \mathcal{W}$, we ask for a set $d \in \mathcal{D}$ having a maximal intersection with q, that is, $|q \cap d'| \leq |q \cap d|$ for all $d' \in \mathcal{D}$.

Such a database member d is called an *answer* (to the query). In order to determine an answer efficiently, we allow to preprocess the database. We demand $m \geq \log_2 n$. (Note that the above problem is mentioned in a more restrictive version with respect to time constraints in [HLN07].)

The approximate version of the maximal intersection problem, called the *k-approximate maximal intersection (k-AMI* , or just *AMI)* problem , is defined as follows:

Input: A database $\mathcal{D}$ of n finite sets $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$.

Query: Given a finite query set $q \subseteq \mathcal{W}$ and a value $k > 1$, we ask for a set $d \in \mathcal{D}$ such that the k-fold intersection size between d and q is at least as large as the intersection size between q and any other set from $\mathcal{D}$. That is, we ask for a set $d \in \mathcal{D}$ such that for all $d' \in \mathcal{D}$ we have $|q \cap d'| \leq k \cdot |q \cap d|$.

Such a database member d is called a *k-approximate answer*. The factor k is called the *approximation factor* or the *quality* of the answer.

1.3 Overview of the Results

The main results of this thesis are two new randomized input models for the maximal intersection problem, called the *Zipf model* and the *hierarchical schemes*. Both models are based on the following: Assume that the elements of the query set are sorted according to *some* order which is given by the random (according to our models) database. We distinguish between two kinds of matches. A *r-match* is a set from the database that contains at least r elements of the query set. A *r-prefix match* is a set from the database that contains at least the first r elements of the query set. Clearly, a r-prefix match is always a r-match, while the other way round this is not true. For both kinds of matches, it holds that the probability that such a match exists is close to one for small r, but at some "threshold value" it falls to nearly zero. Our main observation is that the thresholds for r-match and r-prefix match are close to each other. And this is extremely important for solving the maximal intersection problem efficiently. We show that closeness of thresholds with high probability allows to determine an approximate answer in quasi-linear time in the sum of n and m, or in polylogarithmic time in n. In the following we state the results in more detail.

Inputs in the *Zipf model* behave according to *Zipf's law*. Zipf's law is mainly observed in natural language texts and states that the absolute frequency of a word is inversely proportional to its rank. Proving that the Zipf model exhibits the above mentioned closeness of thresholds, we can state a deterministic algorithm that returns a $\frac{1+E(\varepsilon,n)}{1-\Delta(\delta,n)}$-approximate[1] answer with probability $p(\varepsilon,\delta,n)$ tending to one as $n \to \infty$, where $\varepsilon,\delta > 0$ and $E(\varepsilon,n) \to \varepsilon, \Delta(\delta,n) \to \delta$ as $n \to \infty$. The algorithm has a preprocessing time of $\tilde{O}(nm)$ and a query time of $\tilde{O}(\log m + n)$ in the average case and $\tilde{O}(m+n)$ in the worst case. The space required is $n^{1+o(1)}$. In order to cover a wide range of possible inputs we generalize the Zipf model such that it exhibits *Mandelbrot's formula*, which is a generalization of Zipf's law. We show that the algorithm mentioned above can be applied if the input follows the generalized model. While the time and space bounds remain the same in this case, the approximation factor becomes larger about a constant factor which depends on the particular input database.

For the *hierarchical schemes*, we assume that the elements of $\mathcal{W}$ are ordered hierarchically such that each element is assigned to a fixed level in a binary tree-like table (scheme). By showing that in each hierarchical scheme the above mentioned threshold property holds, we can state a deterministic

[1] The exact approximation factor is stated in the proof of Theorem 4.3.1.

Table 1.1: Space and time bounds for the k-AMI algorithms (P-Time denotes the preprocessing time, Q-Time denotes the query time.)

	Zipf Model	Hierarchical Schemes
Space	$n^{1+o(1)}$	$O(n \log n)$
P-Time	$O(nm \log m)$	$O\left(n \log^2 n\right)$
Q-Time (average)	$O\left(\log m \log\log m + n\sqrt{\log n}\right)$	-
Q-Time (worst)	$O\left(m \log m + n\sqrt{\log n}\right)$	$O\left(\log^2 n\right)$

algorithm that returns a $2 \cdot \frac{1+\varepsilon_n}{1-\varepsilon_n}$-approximate[2] answer with probability tending to one as $n \to \infty$, where $\varepsilon_n > 0$. The algorithm has a preprocessing time of $\tilde{O}(n)$ and a query time of $O(\log^2 n)$. The space required is $\tilde{O}(n)$. Table 1.1 shows a detailed overview of the complexity of both algorithms.

Besides these two models and the corresponding algorithms, we present a randomized algorithm which also returns an approximate answer. This algorithm works as follows: It takes a random sample of sets from the database and determines an optimal answer from this sample (i.e., a set that has an intersection of maximal size with the query set among the sample sets). We show that the expected number of remaining sets from the database having a larger intersection with the query set is at most $\frac{n-r}{r+1}$. The algorithm does not require preprocessing on the database and has $O(m \log m)$ query time. That is, the query time does not depend on the database size..

To complement our considerations about the maximal intersection problem, we give the following lower bound in the cell probe model: Assuming $m \in \omega(\log n) \cap n^{o(1)}$, it holds that any (deterministic or randomized, two sided error) algorithm in the cell probe model that uses at most polynomial space in n and m must have a query time of $\Omega(m/\log n)$.

1.4 Related Work

The maximal intersection problem is closely related to the *nearest neighbor search* (NNS) problem which is defined as follows. Given a set $\mathcal{P}$ of n points in some metric space X, preprocess $\mathcal{P}$ so as to efficiently answer queries which require finding the closest point in $\mathcal{P}$ to a query point $q \in X$.

[2]The exact approximation factor is stated in the proof of Theorem 5.3.1.

The NNS problem is fundamental in computational geometry and appears, for example, in algorithms for information retrieval [BYRN99, Sal88, DDL$^+$90, BSMS95], databases and data mining [BO97, HT96], pattern recognition [CH67, DH73], and statistics and data analysis [DW82]. Its pervasiveness is due to the fact that it can be used as a means of indexing if the data is represented as points in some high dimensional metric space and similarity between data points is defined with respect to the underlying metric[3]. Thus, the problem is of particular interest in m-dimensional vector spaces under some l_p norm. There is a large literature on the nearest neighbor search problem, for example [DL76, Cla88, Mei93, AM92, Sam84, FBF77]; a survey can be found in [Tsa99]. For a small number of dimensions, the problem is well-solved (e.g., [DL76, Mei93, Ede87]), while for a large number of dimensions almost all known algorithms suffer from the "curse of dimensionality"[Cla94]. That is, either their space is exponential in the dimension, or their query time is not much better than a linear scan[4] over the data points [WSB98]. To overcome this curse of dimensionality one can consider approximate versions of the NNS problem. The *ε-approximate nearest neighbor search (ε-NNS)* problem asks for a $p \in \mathcal{P}$ such that for all $p' \in \mathcal{P}$ the distance between q and p is at most as large as $(1+\varepsilon)$ times the distance between q and p'. This problem has also been studied extensively, see for example [AMN$^+$98, Kle97, Cla94, IM98, KOR98].

The relation between NNS and the MI problem arises by the fact that the latter one can be considered as a nearest neighbor search problem in a "binary" form. Namely, the elements from $\mathcal{D}$ can be represented as binary vectors over $\mathcal{W}$. Similarity is then defined by the dot product between two vectors. Contrary to the formulation of the NNS problem, the dot product does not meet the requirements of a metric. Note that the NNS problem in the *Hamming cube* is equivalent to the MI problem since both problems can be reduced on each other. We will deal with these reductions in more detail in Chapter 2.

In [IM98], the approximate nearest neighbor search problem is tackled by reducing it to a problem called *point location in equal balls (PLEB)*. PLEB is the following problem: Given n points $P = \{p_1, \ldots, p_n\}$, a similarity measure ℓ, and two values s, t with $s > t$, devise a data structure which for any query point q does the following:

- if there is a point $p_i \in P$ with $\ell(p_i, q) \geq s$ then return "yes" and a point

[3] An *index* is a data structure that is built over a database in order to allow fast search in the database.

[4] By *linear scan* we denote an algorithm that calculates the distance between the query and each data point in order to determine an answer to a search problem.

$p'_i \in P$ such that $\ell(p'_i, q) \geq t$,

- if $\ell(p_i, q) < t$ for all $p_i \in P$ then return "no",
- if for the point $p_i \in P$ with $\ell(p_i, q)$ maximal among all points in P we have $t \leq \ell(p_i, q) \leq s$ then return either "yes" and a point $p'_i \in P$ with $\ell(p'_i, q) \geq t$, or return "no".

Thus, a point $p_i \in P$ with $\ell(p_i, q) \geq s$ is a "promise" that a point $p'_i \in P$ with $\ell(p'_i, q) \geq t$ is returned. Note that the above problem can be defined analogously for dissimilarity measures (e.g., any distance metric). Besides algorithms solving the ε-NNS problem, an algorithm solving PLEB under set resemblance measure is presented. (The resemblance[5] between two sets a and b is defined as $\frac{|a \cap b|}{|a \cup b|}$; in [Bro97, BGMZ97] this notion was introduced for documents.) This algorithm uses $O(nm + n^{1+\rho})$ space and $O(n^\rho)$ query time, where $\rho = (\ln r)/\ln \varepsilon r$. Now, most interesting for us is PLEB under dot product measure, which can also be solved by the above algorithm. Therefore, PLEB under dot product measure is reduced to PLEB under set resemblance measure by substituting the parameters s, t as follows: Given a query set $q \subseteq \{1, \ldots, m\}$. Assume $r \subseteq \{1, \ldots, m\}$ is a set such that $\frac{|r \cap q|}{|r \cup q|} \geq t$. Since

$$\frac{|r \cap q|}{|r \cup q|} = \frac{\mathbf{r} \cdot \mathbf{q}}{|\mathbf{r}|_1 + |\mathbf{q}|_1 - \mathbf{r} \cdot \mathbf{q}},$$

where $\mathbf{r}, \mathbf{q}$ denote binary vectors such that $r_i = 1$ (respectively, $q_i = 1$) if and only if $i \in r$ (respectively, $i \in q$), and $|\cdot|_1$ denotes the Hamming weight of a binary vector, we have

$$\mathbf{r} \cdot \mathbf{q} \geq \frac{t}{1+t}(|\mathbf{r}|_1 + |\mathbf{q}|_1).$$

Thus, if we look for an answer to PLEB under dot product measure with parameters s', t', we must determine a set r such that

$$t' \leq \frac{t}{1+t}(|\mathbf{r}|_1 + |\mathbf{q}|_1) \tag{1.1}$$

holds. This can be accomplished by splitting the data points into $O(\log m)$ groups of approximately the same weight and then determining an answer set in a group such that the weight of the vectors in this group satisfies (1.1). We want to stress that PLEB under dot product measure and the MI problem are different problems, since the former one asks for a set having an intersection of a fixed certain size with the query set, and not for a set having

[5]The set resemblance defined as above is also called the *Jaccard coefficient*.

an intersection of maximal size with the query set. Moreover, an answer to PLEB is not a k-approximate answer to the MI problem since the size of the intersection of such an answer and the query can be arbitrarily far away from the maximal intersection size.

Finally, we mention the *bit vector intersection problem* which is similar to the MI problem. Given a large collection of n sparse vectors and a parameter k, the task is to find all pairs of vectors with at least k ones in common. Assuming that the number of ones common to any two vectors is significantly less than k, except for an unknown number of pairs which is linear in n, then there exist two randomized algorithms solving the problem with high probability and in subquadratic (in n) expected time [KWZ95].

1.5 Notation and Basic Facts

1.5.1 Notation

Let a and b be numbers. By $a \cdot b$ we denote the product of a and b. To shorten notation, we will often omit the symbol $\cdot$ and just write ab instead of $a \cdot b$. By 2^S we denote the power set of a set S. Given two vectors $\mathbf{x} = \langle x_1, \ldots, x_m \rangle$ and $\mathbf{y} = \langle y_1, \ldots, y_m \rangle$. By $\mathbf{x} \cdot \mathbf{y}$ we denote the dot product $\sum_{i=1}^{m} x_i \cdot y_i$ of $\mathbf{x}$ and $\mathbf{y}$. Given two vectors $\mathbf{x}$ and $\mathbf{y}$ from the m-dimensional Hamming cube $C_m = \{0,1\}^m$, we denote by $h(\mathbf{x}, \mathbf{y})$ the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, that is, the number of positions on which they differ. Let X be a random variable. By $\mathbb{E}X$ we denote the expected value of X and by $\mathrm{Var}X$ we denote the variance of X. By $[A]$ we denote the indicator variable for the event A, that is, $[A] = 1$ if the event A happens and $[A] = 0$ otherwise. Let f, g be functions from $\mathbb{N}$ to $\mathbb{N}$. We use O-notation to describe asymptotic growth rates of functions:

- $f(n) \in O(g(n))$ if and only if
$$\exists c > 0 \, \exists n_0 \, \forall n \geq n_0 : f(n) \leq c \cdot g(n).$$
- $f(n) \in o(g(n))$ if and only if
$$\forall c > 0 \, \exists n_0 \, \forall n \geq n_0 : f(n) \leq c \cdot g(n).$$
- $f(n) \in \Omega(g(n))$ if and only if $g(n) \in O(f(n))$.
- $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$.
- $f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$.

For the sake of clarity, we will sometimes use $\tilde{O}$-notation to hide polylogarithmic terms, that is, $\tilde{O}(f) := \bigcup_{k>0} O(f \log^k f)$. For a more detailed treatment of O-notation we refer to [Knu97]. By *quasi-linear* we denote functions which are in $\Theta(n \log^k n)$ for a $k > 0$, where *near-linear* refers to functions which are in $\Theta(n^\theta)$ for all $\theta > 1$. All running times stated in this thesis hold for the *word RAM* model. A word RAM is a unit-cost random access machine with a word size of w bits, for some w, and an instruction set similar to that of present-day computers[6]. By log we always mean $\log_2$, while ln denotes $\log_e$. The *(collection) frequency* of a word is the overall number of occurrences of the word in a document (respectively, in a document collection). A table containing the words of a document (respectively, of a document collection) in descending order according to frequency (with or without frequencies) is called a *frequency table*. The *rank* of a word is the numerical position of the word in this table.

1.5.2 Probability Theory

Let X be a non-negative random variable. Then, for all $r \in \mathbb{R}$ with $r > 0$ the *Markov inequality* states that

$$\Pr(X \geq r) \leq \frac{\mathbb{E}X}{r}. \tag{1.2}$$

The *Chebyshev inequality* states that

$$\Pr(|X - \mathbb{E}X| \geq r) \leq \frac{\mathrm{Var}X}{r^2}. \tag{1.3}$$

Note that the Chebyshev inequality is also true if X takes values smaller than zero. We will need the following lemma.

Lemma 1.5.1. *Let $X_1, \ldots, X_n$ be independent non-negative random variables. Let $X(n) := \sum_{i=1}^{n} X_i$. If there exist strictly increasing functions $f, g : \mathbb{R} \to \mathbb{R}$ such that $f(n) - c \leq \mathbb{E}X(n)$ for some constant $c \geq 0$, $\mathrm{Var}X(n) \leq g(n)$, and $g(n) \in o(f^2(n))$, then for any $\varepsilon > 0$*

$$\lim_{n\to\infty} \Pr(f(n) - X(n) \geq \varepsilon f(n)) = 0.$$

Proof. Let n_0 such that $\varepsilon f(n_0) > c$ holds. Now, let $0 < \gamma_0 < \varepsilon$ such that $\varepsilon f(n_0) - c \geq \gamma_0 f(n_0)$. By Chebyshev's inequality (1.3)

$$\lim_{n\to\infty} \Pr(|X(n) - \mathbb{E}X(n)| \geq \gamma_0 f(n)) = 0$$

[6] We can assume that the word size is restricted to be at most $m^{O(1)}$ bits.

holds, which implies

$$\lim_{n\to\infty} \Pr\left(f(n) - X(n) \geq \varepsilon f(n)\right) = 0$$

for $X(n) < \mathbb{E}X(n)$. For $X(n) \geq \mathbb{E}X(n)$ the statement holds trivially. □

Let $X_1, \ldots, X_n$ be independent Bernoulli distributed random variables such that $\Pr(X_i = 1) = p_i$ and $\Pr(X_i = 0) = 1 - p_i$. Let $X := \sum_{i=1}^{n} X_i$. For all $0 < \delta \leq 1$ the following *Chernoff bound* holds:

$$\Pr\left(X \leq (1-\delta)\mathbb{E}X\right) \leq e^{-\mathbb{E}X\delta^2/2} \tag{1.4}$$

Let X be a random variable, let $\lambda > 0$, and let r be an arbitrary number. Then, the *exponential Chebyshev inequality* states that

$$\Pr\left(X \geq r\right) \leq \frac{\mathbb{E}e^{\lambda X}}{e^{\lambda r}}. \tag{1.5}$$

For more details on probability theory, see for example, [Loè77, SS01].

1.5.3 Inequalities

Let $a, b > 0$. Then

$$\left(1 - \frac{a}{b}\right)^b < e^{-a} \qquad \text{if} \quad a \leq b, \tag{1.6}$$

and

$$\left(1 - \frac{1}{ab}\right)^a \geq 1 - \frac{1}{b} \qquad \text{if} \quad a, b \geq 1\ . \tag{1.7}$$

Proof. Let $g(x) = \ln(1-x)/x$, $0 < x < 1$. Notice that g is a decreasing function. Inequality (1.6) follows from $g(a/b) \leq \lim_{x\to 0} g(x) = -1$ (note that this shows the inequality if $a < b$; if $a = b$, then the inequality follows immediately), while inequality (1.7) is equivalent to $g(1/b) \leq g(1/ab)$.

□

We will also use the fact that for all integers $n > 1$

$$\ln n < \sum_{k=1}^{n} \frac{1}{k} < \ln n + 1 \tag{1.8}$$

holds (see e.g., [GKP02]).

Chapter 2

Lower Bounds

We state our lower bounds in the context of the cell probe model. The bounds are derived from known bounds for the nearest neighbor search problem in the Hamming cube (see Section 1.4).

2.1 Preliminaries

In this chapter we demand that the input for the maximal intersection problem, that is the database and the query set, has a special form. Namely we demand that each set is represented by its *index vector*. The index vector of a set $d \in 2^{\mathcal{W}}$ is the unique vector $\mathbf{d} \in \{0,1\}^m$ such that $w_i \in d$ if and only if $d_i = 1$. Note that the intersection of two sets $d, \tilde{d} \in 2^{\mathcal{W}}$ is equal to the dot product of the corresponding index vectors $\mathbf{d}, \tilde{\mathbf{d}} \in \{0,1\}^m$, that is, $|d \cap \tilde{d}| = \mathbf{d} \cdot \tilde{\mathbf{d}}$.

2.2 The Cell Probe Model

The *cell probe model*, formulated by Yao and Fredman [cY81, Fre78], is a model for studying the complexity of data structure problems. By abuse of notation, we denote in the following by f a data structure problem and the associated function. A *data structure problem* f is the following: Given a finite set D of databases, a finite set Q of queries, and a finite set A of answers along with a function $f : D \times Q \to A$. We want to devise a method for storing any $\mathcal{D} \in D$ as a data structure in the memory of a word RAM such that, once a $\mathcal{D} \in D$ has been stored, for all $q \in Q$ the value $f(\mathcal{D}, q)$ can be computed efficiently. Such a method is called a *solution* to the data structure problem f.

A solution to f in the cell probe model with parameters s, b, and t is given by assigning to each $\mathcal{D} \in D$ a representation (data structure) $\phi_f(\mathcal{D}) \in R^s$, where $R = \{0,1\}^b$, and associating with each $q \in Q$ a decision tree over R^s of depth t. A *decision tree* is a rooted tree where each internal node is labeled with an index of a cell and has 2^b children. For each internal node, every outgoing edge is labeled with a value from R, whereas each value in R occurs exactly once. The leafs are labeled with values from A. A query q is processed by starting at the root node, reading the content $c \in R$ of the cell i (assuming that i is the label of the root node), and proceeding along the edge with label c. This process is continued until a leaf is reached.

Note that in the cell probe model only the space required for the data structure and the number of probes to it (the tree depth) is measured; all (other) computation is assumed to be for free. As consequence, a cell probe lower bound is a lower bound on the complexity of any algorithm for the problem as long as the algorithm is implemented on a word RAM where each instruction operates on a constant number of registers of size b. For a comprehensive survey about the cell probe model see [Mil99].

2.3 Reducing NNS in the Hamming Cube to the MI Problem

We now present a straightforward reduction from the nearest neighbor search problem in the Hamming cube to the maximal intersection problem.

Lemma 2.3.1. *Given a database $\mathcal{C} = \{c_1, \ldots, c_n\} \subseteq C_\ell$ and a query vector $\mathbf{q} \in C_\ell$. There exists a function $f : C_\ell \to \{0,1\}^{2\ell}$ with the following properties: A vector $\mathbf{c} \in \mathcal{C}$ is a nearest neighbor of $\mathbf{q}$ (with respect to the Hamming distance) if and only if the dot product between $\mathbf{f}(\mathbf{c}) \in \mathcal{D} = \{\mathbf{f}(\mathbf{c_1}), \ldots, \mathbf{f}(\mathbf{c_n})\}$ and $\mathbf{f}(\mathbf{q})$ is maximal among all elements from $\mathcal{D}$. Furthermore, f can be computed in linear time.*

Proof. Given $\mathbf{x} \in C_\ell$. We define $\mathbf{x}' = \mathbf{f}(\mathbf{x}) \in \{0,1\}^{2\ell}$ as follows:
For $i \in \{1, 3, 5, \ldots, 2\ell - 1\}$ set

$$x'_i x'_{i+1} = \begin{cases} 01 & : \quad x_{\lceil i/2 \rceil} = 0 \\ 10 & : \quad x_{\lceil i/2 \rceil} = 1 \end{cases}.$$

Then, $h(\mathbf{x}, \mathbf{y}) = \ell - \mathbf{f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{y})$ holds since every position at which $\mathbf{x}$ and $\mathbf{y}$ match increases $\mathbf{f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{y})$ by one. Every mismatch between $\mathbf{x}$ and $\mathbf{y}$ does not alter $\mathbf{f}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{y})$. Obviously, f can be computed in time which is linear in the input size. □

Note that the above reduction does not apply to the approximate versions of NNS and the MI problem. This can be seen as follows. Let $\mathbf{c} \in \mathcal{C}$ such that $\mathbf{f}(\mathbf{c}) \cdot \mathbf{f}(\mathbf{q})$ is maximal (i.e., $\mathbf{c}$ is a nearest neighbor of $\mathbf{q}$). Let $\mathbf{c}' \in \mathcal{C}$ such that $\mathbf{f}(\mathbf{c}')$ is a k-approximate answer to the MI problem for some $k > 1$, that is $\mathbf{f}(\mathbf{c}) \cdot \mathbf{f}(\mathbf{q}) \leq k(\mathbf{f}(\mathbf{c}') \cdot \mathbf{f}(\mathbf{q}))$. This inequality is equivalent to

$$h(\mathbf{c}', \mathbf{q}) \leq \frac{1}{k} h(\mathbf{c}, \mathbf{q}) + \frac{(k-1)\ell}{k} \ .$$

Thus, we cannot bound the Hamming distance of the approximate solution by the minimal Hamming distance multiplied by some constant.

Remark 1. Notice that the maximal intersection problem can also be reduced to the nearest neighbor search problem in the Hamming cube as follows: Elements from the database $\mathcal{D}$ are mapped by a function $g_{\mathcal{D}} : \{0,1\}^{\ell} \to C_{3\ell}$ such that $g_{\mathcal{D}}(0) = 110$, $g_{\mathcal{D}}(1) = 000$, and query elements are mapped by a function $g_q : \{0,1\}^{\ell} \to C_{3\ell}$ such that $g_q(0) = 101$, $g_q(1) = 000$. We have to distinguish between zeros in database and query elements so that every combination except two ones increases the Hamming distance by the same value. Then, a maximal dot product corresponds to a minimal Hamming distance. By the same argument as above, this reduction does only apply to the exact versions of the problems, and not to the approximate ones.

2.4 Lower Bounds for the Maximal Intersection Problem

First, we overview known cell probe lower bounds for NNS in the Hamming cube. We then show that these bounds also apply to the maximal intersection problem. Note that all lower bounds for NNS in the cube follow from lower bounds for the *λ-neighbor problem*, a decision version of NNS where the answer to a query is one if and only if there exists a point in the database at distance at most λ from the query point. Otherwise, the answer is zero. Obviously, NNS is at least as hard as the λ-neighbor problem.

Trivial Solutions. Consider the nearest neighbor search problem in the Hamming cube where the database has size n and each element has length m. There exist two trivial solutions to the problem. The first solution is to store the data without any preprocessing on it and to answer queries by computing the distance from a query point to each database element. Both, the storage requirement and search time are in $O(nm)$. Since in the Hamming cube the set of possible queries is finite, the second trivial solution is to compute and

to store in the preprocessing step the answer to each possible query point. Then, answering a query reduces to a look-up in the data structure which takes $O(m)$ time. The storage required is $O(2^m)$. Consequently, a *preferable* solution is one that uses $(nm)^{O(1)}$ storage and has $m^{O(1)}$ search time. In the following, we assume that $m \in \omega(\log n) \cap n^{o(1)}$ holds, since otherwise the problem becomes trivial (if $m = O(\log n)$ holds, then storing a table with answers to each query point requires $2^{O(\log n)} = n^{O(1)}$ storage; if $m = n^c$ for a $c > 0$, then the linear scan takes $m^{O(1)}$ search time).

Borodin et al.[BOR99] proved that any randomized two-sided error algorithm for the λ-neighbor problem using t probes, either uses $2^{\Omega(\log n \log m/t)}$ cells, or uses cells of size $\Omega(n^{1-\varepsilon}/t)$, where $\varepsilon > 0$. Hence, an algorithm that uses $n^{O(1)}$ cells of size $m^{O(1)}$ must make $\Omega(\log m)$ probes to the data structure.

This bound was improved by Barkol et al.[BR02]. It was shown that if the algorithm uses t probes, then either it uses $2^{\Omega(m/t)}$ cells, or it uses cells of size $n^{\Omega(1)}/t$. Consequently, using $n^{O(1)}$ cells of size $m^{O(1)}$ yields $t \in \Omega(m/\log n)$. Actually, this bound is an improvement only if $d \in \omega(\log^2 n)$ holds. Otherwise, the bound given by Borodin et al. is stronger.

Finally, this bound could be even more improved if the number of cells is restricted to $s = n^{1+o(1)}$. Then, it holds that $t \in \Omega(m/\log \frac{s \cdot m}{n})$ [PT06].

Lower bounds for the maximal intersection problem can be obtained by applying Lemma 2.3.1. Therefore, we define (analogously to the λ-neighbor problem) the *λ-intersection problem*. In this problem, the answer to a query has to be one if and only if the database $\mathcal{D}$ contains a set such that the intersection size between the query set and this set is at least λ. Otherwise, the answer is zero. We get the following theorem.

Theorem 2.4.1 (Lower bound for the λ-intersection problem)**.** *Let $m \in \omega(\log n) \cap n^{o(1)}$. Then any randomized two sided error algorithm in the cell probe model for the λ-intersection problem that uses at most $n^{O(1)}$ cells of size $m^{O(1)}$ must make $\Omega(\frac{m}{\log n})$ probes to the data structure.*

Proof. By Lemma 2.3.1 we can reduce the λ-neighbor problem in C_ℓ to the $(\ell - \lambda)$-intersection problem in $\{0,1\}^{2\ell}$. □

Clearly, the maximal intersection problem is at least as hard as the λ-intersection problem. Therefore, the lower bound applies to the maximal intersection problem as well. Note that in Theorem 2.4.1, we state a lower bound in case the number of cells is polynomially bounded in n and m. Actually, *all* lower bounds given by Barkol et al. also hold for the MI problem. We consider the bound stated above as some kind of "working version".

Further note that a lower bound for randomized algorithms is also a lower bound for deterministic algorithms.

Remark 2. So far, for arbitrary n and m (with the restriction that $m \in \omega(\log n) \cap n^{o(1)}$ holds) no preferable solution (in the above sense) to NNS in the Hamming cube is known. Moreover, it is believed that such a solution cannot exist [BOR99]. Thus, Theorem 2.4.1 indicates that we cannot expect that a preferable solution to the maximal intersection problem exists, since such a solution would imply a preferable solution to NNS as well.

Chapter 3

A Randomized Approach

In this chapter we present a simple randomized algorithm that solves an approximate version of the maximal intersection problem.

3.1 The Sampling Lemma

The analysis of many randomized algorithms for geometric optimization problems, as for example computing the smallest enclosing ball of n points in a m-dimensional space, is based on a simple identity, the so called *sampling lemma* [GW00]. Let S be a set of n elements and $\phi : 2^S \mapsto I$ be a function where I is some suitable set depending on the current problem. Consider the following sets:

$$V(R) = \{s \in S \setminus R \mid \phi(R \cup \{s\}) \neq \phi(R)\} \ ,$$
$$X(R) = \{s \in R \mid \phi(R \setminus \{s\}) \neq \phi(R)\} \ .$$

The set $V(R)$ is called the *violators* of R and the set $X(R)$ is called the *extreme* elements in R. Then

$$s \in V(R) \Leftrightarrow s \in X(R \cup \{s\}) \ . \tag{3.1}$$

For a random sample R of size r, that is, a set chosen uniformly at random from the set $\binom{S}{r}$ of all r-element subsets of S, consider the random variables $V_r : \binom{S}{r} \to \mathbb{N}, \ R \mapsto |V(R)|$ and $X_r : \binom{S}{r} \to \mathbb{N}, \ R \mapsto |X(R)|$ and their expected values $v_r := \mathbb{E}V_r$ and $x_r := \mathbb{E}X_r$. The sampling lemma states the following:

Lemma 3.1.1. *(Sampling lemma, [GW00])*
For $0 \leq r < n$*:*

$$v_r = \frac{x_{r+1}}{r+1} \cdot (n - r) \tag{3.2}$$

Proof.

$$\begin{aligned}
\binom{n}{r} \cdot v_r &= \binom{n}{r} \cdot \sum_{R \in \binom{S}{r}} V_r(R)\Pr(R) \\
&= \sum_{R \in \binom{S}{r}} V_r(R) \\
&= \sum_{R \in \binom{S}{r}} \sum_{s \in S \setminus R} [s \in V(R)] \\
&\overset{(3.1)}{=} \sum_{R \in \binom{S}{r}} \sum_{s \in S \setminus R} [s \in X(R \cup \{s\})] \\
&= \sum_{Q \in \binom{S}{r+1}} \sum_{s \in Q} [s \in X(Q)] \\
&= \binom{n}{r+1} \cdot x_{r+1}
\end{aligned}$$

Since $\binom{n}{r+1}/\binom{n}{r} = (n-r)/(r+1)$ the lemma follows. □

3.2 The Algorithm

The idea is to take a random sample of r sets from $\mathcal{D} \subseteq 2^{\mathcal{W}}$ and to determine an optimal answer among these sample sets, that is, a set which has a maximal intersection with the query set among the sample sets. (Note that there can exist several sets having the same maximal intersection size with the query; then, an arbitrary set with this property is returned.) Usually, an answer set determined by the above process has no intersection of maximal size with the query set among *all* database sets. However, we can bound the expected number of remaining sets that have a larger intersection by applying Lemma 3.1.1.

Lemma 3.2.1. *Let $q \subseteq \mathcal{W}$. Let $R \subseteq \mathcal{D}$ be a random sample of size r and let $d_a \in R$ be a set having a maximal intersection with q in R. Then, the expected number of sets from $\mathcal{D} \setminus R$ having a larger intersection with q is at most $(n-r)/(r+1)$.*

Proof. We define the function used for specifying the set $V(R)$ and $X(R)$ as follows:

$$\begin{aligned}
\phi_q : 2^{\mathcal{D}} &\to 2^{\mathcal{D}}, \\
R &\mapsto \{d \in \mathcal{D} \mid \exists d' \in R : |d \cap q| \leq |d' \cap q|\}
\end{aligned}$$

Our main observation is that the cardinality of $X(R)$ is 0 or 1. Assume that $|d \cap q| < |d_a \cap q|$ holds for all $d \in R \setminus \{d_a\}$. Then, $d_a \in X(R)$ since (at least) $d_a \notin \phi_q(R \setminus \{d_a\})$. Thus, $|X(R)| \geq 1$ holds (as we will see later, even $|X(R)| = 1$ holds). Now, assume there exists $d' \in R$ such that $|d_a \cap q| = |d' \cap q|$. Then $\phi_q(R \setminus \{x\}) = \phi_q(R)$ for $x \in \{d_a, d'\}$. Therefore, if there are at least two sets in R having the same maximal intersection size with q among the sets in R , then the set $X(R)$ is empty. Thus, $|X(R)| = 0$ or $|X(R)| = 1$. By Lemma 3.1.1

$$\mathbb{E}\left(\#\{d \in \mathcal{D} \setminus R \mid \forall d' \in R : |d' \cap q| < |d \cap q|\}\right) = v_r \leq \frac{n-r}{r+1}$$

holds. □

Let us consider the case that there exists a $d' \in R$ such that $q \subseteq d'$. Then, for all $d \in \mathcal{D}$ it holds that $\phi_q(R \cup \{d\}) = \phi_q(R)$, which implies that no violators exist (as required in this case).

Given Lemma 3.2.1, we get the following theorem.

Theorem 3.2.2. *Let $0 < r \leq n$ be an integer. Given a database $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ and a query set $q \subseteq \mathcal{W}$. There exists a randomized algorithm that returns an answer to the AMI problem such that an expected number of at most $\frac{n-r}{r+1}$ database sets have a larger intersection with q than this answer. The algorithm has a query time of $O(m \log m)$.*

Proof. The algorithm is obvious. It picks randomly r sets $R \subseteq \mathcal{D}$ and determines a set $d_a \in R$ having a maximal intersection with q in R. According to Lemma 3.2.1, an expected number of at most $\frac{n-r}{r+1}$ remaining sets have a larger intersection with q. Choosing r sets from the database can be done in time $O(1)$. The answer set d_a is determined by sorting the words of q in ascending order according to their index and then calculating the intersection size between each sample set and q by applying binary search. Overall, this can be done in time $O(m \log m)$. □

Remark 3. Note that we cannot state the approximation factor of the above algorithm. However, we can say the following: If we sort the database sets in descending order according to their intersection size with the query set, then the expected rank of the algorithm's answer in this sequence is $(n - r)/(r + 1) + 1 = \frac{n+1}{r+1}$ or better. Our experimental results presented on Table 6.4 show this fact.

Chapter 4

Maximal Intersection Queries in the Zipf Model

In this chapter we investigate the maximal intersection problem under the assumption that the rank-frequency distribution of the elements of the database can be approximated by Zipf's law. We present a deterministic algorithm solving the approximate maximal intersection problem in query time which is quasi-linear in the sum of n and m.

4.1 Preliminaries

Our main motivation for studying the maximal intersection problem were problems like text clustering, near-duplicate detection or the processing of search queries in web search engines. In these problems, the database mainly consists of natural language text documents. Therefore, in this chapter we deal with *documents* and *words* instead of sets and elements. We will use the term *document collection* (or just *collection*) to refer to a database of (textual) documents. Nevertheless, we want to stress that the results we present hold for every input following our model.

4.2 The Model

Consider a collection of n initially empty documents. Let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a set of words. We now describe a probabilistic process for generating a document collection which will be called the *Zipf model*. A document is generated by choosing words that will be contained in it. Each word is chosen independently of the other words and the word w_i is chosen with probability $1/i$. Every document is generated independently. Notice that a document

collection generated by the Zipf model (or, *following the Zipf model*) can contain documents d_i, d_j with $d_i = d_j$ for $i \neq j$. The expected number of words in a document generated by the Zipf model (or, *following the Zipf model*) is approximately equal to $\ln m$. This probabilistic approach was inspired by a recent survey of Newman [New03]. He gives a comprehensive survey about random graph models. Most similar to the Zipf model is the *configuration model*. In short, in the configuration model a degree distribution p_k is specified such that p_k is the fraction of vertices in a graph having degree k. Then, the degrees k_i of vertices $i = 1, \ldots, n$ are chosen from this distribution (i.e., giving each vertex i in the graph k_i stubs sticking out of it) and the vertices are connected at random. A document collection following the Zipf model can be considered as a bipartite random graph, where one vertex set represents the documents and the other set represents the words. In such a graph, the word degrees are distributed according to Zipf's law and the documents have (approximately) constant degree. Put it another way, in a document collection following the Zipf model the word frequencies are distributed according to Zipf's law. *Zipf's law* belongs to the family of discrete power laws, and is mainly applied in linguistics. It states that in natural language texts the frequency f of a word is inversely proportional to its rank r in the frequency table, that is,

$$f \propto \frac{1}{r},$$

which means there exists a constant c such that $f \cdot r \approx c$. The law was proposed in 1935 by George Kingsley Zipf[1]. It is an empirical law which can be observed in any large enough text. Table 4.1 is taken from [MS99] and shows an extract of the rank-frequency distribution of Mark Twain's *Tom Sawyer*. Although Zipf uncovered the law for languages in the first place, it also holds (approximately) for other real-world observations following a power law, for example city sizes, or lengths of rivers. Since the *collection frequency* (in the following just *frequency*) of a word w in a collection $\mathcal{D}$ following the Zipf model is defined as

$$\left|\{d \in \mathcal{D} \mid w \in d\}\right|,$$

we conclude that the expected frequency of the word w_i in the Zipf model is equal to n/i. The expected rank of w_i is exactly the i-th value among those of all words. Therefore, the word frequencies are indeed distributed according to Zipf's law. As mentioned above, Zipf's law describes the rank-frequency distribution of words in natural language texts. Since some of

[1]Presumably, J. B. Estoup noticed the law before Zipf [Est16].

Table 4.1: Empirical evaluation of Zipf's law on *Tom Sawyer*

Word	Freq.	Rank	$f \cdot r$	Word	Freq.	Rank	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

our motivating applications also deal with natural language texts, we can state that the Zipf model agrees with real life at least by the rank-frequency distribution of words. For more details about Zipf's law see [Zip49, MS99].

Remark 4. By defining the probability of the word w_i to be contained in a document as $1/i$, the set $2^{\mathcal{W}}$ yields a probability space where a document d is an event occurring with probability $\Pr(d) = \left(\prod_{w_i \in d} \frac{1}{i}\right)\left(\prod_{w_i \notin d} 1 - \frac{1}{i}\right)$.

Remark 5. Note that sorting the words in a document collection $\mathcal{D} \subseteq 2^{\mathcal{W}}$ following the Zipf model in descending order according to their frequency yields the same order as sorting them in ascending order according to their index.

For further considerations we introduce the following definitions:

Definition 4.2.1. Let $r \geq 0$. Given a document collection $\mathcal{D} \subseteq 2^{\mathcal{W}}$ and a query document $q \subseteq \mathcal{W}$. By *r-match* we denote a document from the collection $\mathcal{D}$ that contains at least r words of q, that is, the size of the intersection of this document and the query document q is at least r.

Definition 4.2.2. Let $r \geq 0$. Given a document collection $\mathcal{D} \subseteq 2^{\mathcal{W}}$ and a query document $q \subseteq \mathcal{W}$. Assume that the words of q are sorted in ascending order according to their index. By *r-prefix match* we denote a document from the collection $\mathcal{D}$ that contains at least the first r words of q.

Definition 4.2.3. We partition the set of words as follows. For $i \in \{1, \ldots, \lceil \ln m \rceil - 1\}$ the group P_i consists of the words $w_{\lceil e^{i-1} \rceil}$ up to $w_{\lfloor e^i \rfloor}$. The group $P_{\lceil \ln m \rceil}$ consists of the words $w_{\lceil e^{\lceil \ln m \rceil - 1} \rceil}$ up to w_m.

$$\underbrace{w_1\ w_2}_{P_1} \quad \underbrace{w_3\ w_4\ w_5\ w_6\ w_7}_{P_2} \quad \ldots$$

We say that a document $d \in 2^{\mathcal{W}}$ is *regular* if it contains exactly one word from each group.

Remark 6. Note that the groups do not overlap since x and e^x cannot be both algebraic for $x > 0$ [Wal69]. Note also that the expected number of words a document contains from each group is approximately one.

Definition 4.2.4. Let $0 < \delta < 1$. We say that a document $d \in 2^{\mathcal{W}}$ is (δ, n)-*generic* if for all $i \geq \delta\sqrt{2 \ln n}$ the following holds:

$$|d \cap (P_1 \cup \ldots \cup P_i)| \geq (1 - \delta)i$$

This means that a (δ, n)-generic document contains for all $i \geq \delta\sqrt{2 \ln n}$ at least $(1 - \delta)i$ words w_j with $j \leq e^i$.

Lemma 4.2.5. *Let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. Let $d \in 2^{\mathcal{W}}$ be a random document following the Zipf model. For sufficiently large n, m the probability that d is (δ, n)-generic is greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1 - c)$.*

Proof. First, we consider a fixed $i \geq 1$ and let X be a random variable denoting the expected number of words in d up to the word $w_{\lfloor e^i \rfloor}$. In order to prove the lemma we need that $i < \mathbb{E}X = \sum_{k=1}^{\lfloor e^i \rfloor} 1/k$ holds. We have

$$i < \ln\lfloor e^i \rfloor + 0.5$$

since $e^{0.5} > 1.5$ and therefore $e^i < e^{0.5}\lfloor e^i \rfloor$. Since Euler's constant γ is greater than 0.5 and $\mathbb{E}X = \ln\lfloor e^i \rfloor + \gamma + O(1/\lfloor e^i \rfloor)$ (see e.g., [GKP02]), we conclude that

$$i < \ln\lfloor e^i \rfloor + 0.5 < \mathbb{E}X \tag{4.1}$$

holds. Now the Chernoff bound (1.4) yields that the probability that d contains less than $(1 - \delta)i$ words up to the word $t_{\lfloor e^i \rfloor}$ is smaller than $e^{-i\delta^2/2}$. This holds since $i < \mathbb{E}X$ and therefore

$$\Pr\left(X \leq (1 - \delta)i\right) \leq \Pr\left(X \leq (1 - \delta)\mathbb{E}X\right) \leq e^{-\mathbb{E}X\delta^2/2} < e^{-i\delta^2/2}\ .$$

So the probability that d is not (δ, n)-generic for large n is bounded by

$$\sum_{i\geq 0} c^i - \sum_{i=0}^{\lfloor \delta\sqrt{2\ln n}\rfloor -1} c^i = \sum_{i\geq \lfloor \delta\sqrt{2\ln n}\rfloor} c^i = c^{\lfloor \delta\sqrt{2\ln n}\rfloor} \cdot \sum_{i\geq 0} c^i$$
$$= \frac{c^{\lfloor \delta\sqrt{2\ln n}\rfloor}}{1-c} < \frac{c^{\sqrt{2}\,\delta\sqrt{\ln n}-1}}{1-c}$$
$$< \frac{c^{1.3\,\delta\sqrt{\ln n}}}{1-c}\ .$$

This holds because $1.3 < \sqrt{2}$ and n is large. Overall, the probability that for all $i \geq \delta\sqrt{2\ln n}$ the document d contains at least $(1-\delta)i$ words w_j with $j \leq e^i$ is greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$. Note that we need m to be sufficiently large so that at least $\lfloor e^{\lceil \delta\sqrt{2\ln n}\rceil}\rfloor$ words exist. □

Lemma 4.2.6. *Let $d \in 2^{\mathcal{W}}$ be a random document following the Zipf model and let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. For sufficiently large n, m the following holds: If we insert the first (regarding the word order by frequency) $\lceil \delta\sqrt{2\ln n}\rceil$ missing words to d then*

$$\Pr\left(\forall i \leq \sqrt{2\ln n} :\ |d \cap (P_1 \cup \ldots \cup P_i)| \geq i\right) > 1 - \frac{c^{1.3\,\delta\sqrt{\ln n}}}{1-c} \tag{4.2}$$

Proof. For all $i \geq \delta\sqrt{2\ln n}$ Lemma 4.2.5 states that for sufficiently large n, m the probability that d is (δ, n)-generic is greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$. Since $\delta i \leq \lceil \delta\sqrt{2\ln n}\rceil$ for all $i \leq \sqrt{2\ln n}$ and since we insert the first $\lceil \delta\sqrt{2\ln n}\rceil$ missing words to d, for sufficiently large n, m the probability that $|d \cap (P_1 \cup \ldots \cup P_i)| \geq i$ holds for all $\delta\sqrt{2\ln n} \leq i \leq \sqrt{2\ln n}$ is also greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$. Note that by inserting the first $\lceil \delta\sqrt{2\ln n}\rceil$ missing words to d, also for all $i \leq \delta\sqrt{2\ln n}$ it is true that $|d \cap (P_1 \cup \ldots \cup P_i)| \geq i$. Overall, inequality (4.2) holds. □

We now introduce a threshold to give statements about the most probable size of a maximal intersection:

$$s = s_n := \sqrt{2\ln n}$$

Theorem 4.2.7 (Threshold theorem for the Zipf model)**.** *Let $m \geq \lfloor e^{\lfloor s\rfloor -2}\rfloor$ and let $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ be a document collection following the Zipf model.*

1. *Let $0 < \delta < 1$ be fixed. Let $\zeta = 2 + \lceil \delta s \rceil$ and $c = e^{-\delta^2/2}$. Given a query document $q \subseteq \mathcal{W}$ following the Zipf model. Then for sufficiently large n the following holds: The probability that there exists a $(\lfloor s \rfloor - \zeta)$-*prefix *match is greater than $1 - c^{\delta\sqrt{\ln n}}/(1-c)$. Thus, the probability tends to one as $n \to \infty$.*

2. *Let $\varepsilon > 0$ be fixed. Given a query document $q \subseteq \mathcal{W}$ following the Zipf model. Then the probability that there exists a $\lceil (1+\varepsilon)s \rceil$-match tends to zero as $n \to \infty$.*

Proof. 1. Let $d_R \in 2^{\mathcal{W}}$ be a fixed regular document (Definition 4.2.3). The probability that a random document from $2^{\mathcal{W}}$ following the Zipf model contains the first $\lfloor s \rfloor - 2$ words of d_R is at least

$$\frac{1}{\lfloor e \rfloor} \cdot \ldots \cdot \frac{1}{\lfloor e^{\lfloor s \rfloor - 2} \rfloor} > \frac{1}{e} \cdot \ldots \cdot \frac{1}{e^{\lfloor s \rfloor - 2}} = \frac{1}{\exp\left(\sum_{i=1}^{\lfloor s \rfloor - 2} i\right)} > \frac{1}{\exp\left((\lfloor s \rfloor - 1)^2/2\right)} > \frac{1}{\exp\left((s^2 - 2s + 1)/2\right)} = \frac{\exp\left(s - 1/2\right)}{n} .$$

Note that $\exp(s - 1/2) < n$ since the above probability is less than one. This means that the probability that there exists no document in $\mathcal{D}$ containing the first $\lfloor s \rfloor - 2$ words of d_R is no more than

$$\left(1 - \frac{\exp\left(s - 1/2\right)}{n}\right)^n < \exp\left(-\exp\left(s - \frac{1}{2}\right)\right),$$

which follows from inequality (1.6). So with probability greater than

$$1 - \exp\left(-\exp\left(s - \frac{1}{2}\right)\right)$$

there exists a document in $\mathcal{D}$ having the first $\lfloor s \rfloor - 2$ words of d_R. Consider the query q. Assume we insert the *first* $\lceil \delta s \rceil$ missing words to q. Then, Lemma 4.2.6 implies that for large n the probability that for each $i \leq s$ the query q contains at least i words w_j with $j \leq e^i$ is greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$. This means that the probability that for each $i \leq s$ the query contains at least as many words as d_R among the first i groups is also greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$, see Figure 4.1. Thus, the probability that there exists a document in $\mathcal{D}$ matching the first $\lfloor s \rfloor - 2$ words of the query q is greater than

$$\left(1 - \frac{c^{1.3\,\delta\sqrt{\ln n}}}{1-c}\right)\left(1 - \exp\left(-\exp\left(s - \frac{1}{2}\right)\right)\right) .$$

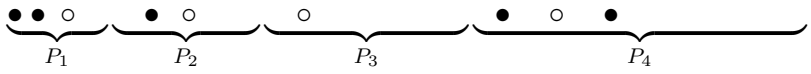


Figure 4.1: "Regularization" of the query ($d_R = \circ$, $q = \bullet$)

For large n, this product is at least $1 - c^{\delta\sqrt{\ln n}}/(1-c)$. It remains to notice that by removing the imaginary inserted $\lceil \delta s \rceil$ words from q we still match $\lfloor s \rfloor - 2 - \lceil \delta s \rceil = \lfloor s \rfloor - \zeta$ words. This completes the proof of part one.

2. At first we consider the query q as a fixed document and let d be a random document following the Zipf model. For every word $w_j \in q$ define the random variable

$$W_j := \begin{cases} 1 & : \quad \text{if} \quad w_j \in d, \\ 0 & : \quad \text{else.} \end{cases}$$

Note that these random variables are pairwise independent since in the Zipf model words are chosen independently and every document is also generated independently. Let $W = \sum_{j:w_j \in q} W_j$. The main idea is to evaluate the Laplace transform (with parameter $\lambda > 0$) of W (i.e., of the intersection size between q and d)

$$\begin{aligned} \mathbb{E}\exp\left(\lambda \left|q \cap d\right|\right) &= \mathbb{E}\left(e^{\lambda W}\right) = \mathbb{E}\left(\exp\left(\lambda \sum_{j:w_j \in q} W_j\right)\right) \\ &= \mathbb{E}\left(\prod_{j:w_j \in q} e^{\lambda W_j}\right) = \prod_{j:w_j \in q} \mathbb{E}e^{\lambda W_j} \\ &= \prod_{j:w_j \in q} \left(1 - \frac{1}{j} + \frac{1}{j}e^{\lambda}\right) \leq \prod_{j:w_j \in q} \left(1 + \frac{e^{\lambda}}{j}\right) \\ &= \prod_{\substack{j:w_j \in q \\ j > e^{\lambda}}} \left(1 + \frac{e^{\lambda}}{j}\right) \cdot \prod_{\substack{j:w_j \in q \\ j \leq e^{\lambda}}} \frac{e^{\lambda}}{j}\left(1 + \frac{j}{e^{\lambda}}\right). \end{aligned}$$

Using the Taylor series expansion of $\ln(1+x) = \sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^k}{k} \leq x$

for $x \in (-1, 1]$, it follows that

$$\begin{aligned}\ln \mathbb{E}\exp\left(\lambda|q \cap d|\right) &\leq \sum_{\substack{j:w_j \in q \\ j \geq e^\lambda}} \frac{e^\lambda}{j} + \sum_{\substack{j:w_j \in q \\ j \leq e^\lambda}} (\lambda - \ln j) + \sum_{\substack{j:w_j \in q \\ j \leq e^\lambda}} \frac{j}{e^\lambda} \\ &= e^\lambda T_1 + \lambda T_2 - T_3 + e^{-\lambda} T_4,\end{aligned}$$

where

$$T_1 = \sum_{\substack{j:w_j \in q \\ j \geq e^\lambda}} \frac{1}{j}, \qquad T_2 = \left|q \cap \left\{w_j \mid 1 \leq j \leq e^\lambda\right\}\right|,$$

$$T_3 = \sum_{\substack{j:w_j \in q \\ j \leq e^\lambda}} \ln j, \qquad T_4 = \sum_{\substack{j:w_j \in q \\ j \leq e^\lambda}} j.$$

Let $0 < \gamma < \frac{1}{5}$. We will assume that the following four *regularity conditions* are verified. (In fact, the probability that each of these conditions is true tends to one if we consider q as a random document following the Zipf model and let $\lambda \to \infty$; the proof follows later.)

$$e^\lambda T_1 \leq \gamma\lambda, \qquad T_2 \leq (1+\gamma)(\lambda+1),$$

$$T_3 \geq (1-\gamma)\frac{\lambda^2}{2}, \qquad e^{-\lambda} T_4 \leq \gamma\lambda.$$

Under these regularity conditions we obtain for sufficiently large λ

$$\begin{aligned}\ln \mathbb{E}\exp\left(\lambda|q \cap d|\right) &\leq 2\gamma\lambda + (1+\gamma)\lambda(\lambda+1) - (1-\gamma)\frac{\lambda^2}{2} \\ &\leq \underbrace{3\gamma\lambda}_{\leq \gamma\lambda^2/2} + 3\gamma\frac{\lambda^2}{2} + \frac{\lambda^2}{2} + \underbrace{\lambda}_{\leq \gamma\lambda^2/2} \\ &\leq (1+5\gamma)\frac{\lambda^2}{2}\ .\end{aligned}$$

Assume that $\{d_j \mid 1 \leq j \leq n\}$ is a sample of n independent documents distributed according to the Zipf model. Then by the exponential Chebyshev inequality (1.5), for any $r > 0$ we have

$$\begin{aligned}\Pr\left(\max_{j \leq n} |q \cap d_j| \geq r\right) &\leq n\ \Pr\left(|q \cap d| \geq r\right) \\ &\leq n\ \frac{\mathbb{E}\exp\left(\lambda|q \cap d|\right)}{e^{\lambda r}} \\ &\leq n\ \exp\left((1+5\gamma)\frac{\lambda^2}{2} - \lambda r\right).\end{aligned}$$

By choosing

$$r = (1 + 5\gamma)\sqrt{2 \ln n}$$

and

$$\lambda = \sqrt{2 \ln n}$$

we obtain $(1 + 5\gamma)\lambda^2 = \lambda r$, hence

$$\begin{aligned} \Pr\left(\max_{j \le n} |q \cap d_j| \ge r\right) &\le n \, \exp\left(-(1 + 5\gamma)\frac{\lambda^2}{2}\right) \\ &= n \, \exp\left(-\ln n - 5\gamma \ln n\right) \\ &= \frac{1}{n^{5\gamma}} \; \to 0 \end{aligned}$$

for $n \to \infty$ (note that $n \to \infty$ implies $\lambda \to \infty$). Part 2 of Theorem 4.2.7 follows as $r = (1 + 5\gamma)\sqrt{2 \ln n} = (1 + \varepsilon)s$ for $\varepsilon = 5\gamma$.

Finally, we present the regularity conditions proof. Therefore, let the query q be randomly chosen according to the Zipf model. We need the following lemma.

Lemma 4.2.8. *For sufficiently large λ the following inequalities hold:*

a) $\mathbb{E}T_1 \le (e^\lambda + 1)e^{-2\lambda}$

b) $\mathbb{E}T_2 \le \lambda + 1$

c) $\mathrm{Var}T_2 \le \lambda + 1$

d) $\frac{\lambda^2}{2} - \left(\frac{\ln^2 2}{2} + \frac{1}{e}\right) < \mathbb{E}T_3$

e) $\mathbb{E}T_3 \le \frac{\lambda^2}{2}$

f) $\mathrm{Var}T_3 \le \frac{\lambda^3}{3}$

g) $\mathbb{E}T_4 \le e^\lambda$

Proof. a) Let

$$X_j = \begin{cases} 1/j & : \text{ if } \; w_j \in q, \\ 0 & : \text{ else.} \end{cases}$$

Then,

$$\mathbb{E}T_1 = \mathbb{E}\sum_{j \ge e^\lambda} X_j = \sum_{j \ge e^\lambda} \frac{1}{j^2} \le \lim_{B \to \infty} \int_{e^\lambda}^{B} \frac{1}{x^2}\, dx + \frac{1}{e^{2\lambda}} = (e^\lambda + 1)e^{-2\lambda}.$$

b) Let

$$Y_j = \begin{cases} 1 & : \quad \text{if} \quad w_j \in q, \\ 0 & : \quad \text{else.} \end{cases}$$

Now,

$$\mathbb{E}T_2 = \mathbb{E}\sum_{j\le e^\lambda} Y_j = \sum_{j\le e^\lambda} \frac{1}{j} \le \int_1^{e^\lambda} \frac{1}{x}\,dx + 1 = \lambda + 1.$$

c) For the variance of T_2, we get

$$\begin{aligned} \mathrm{Var}T_2 = \mathbb{E}T_2^2 - (\mathbb{E}T_2)^2 &= \mathbb{E}\left(\sum_{j\le e^\lambda} Y_j\right)^2 - \left(\sum_{j\le e^\lambda} \frac{1}{j}\right)^2 \\ &= \mathbb{E}\sum_{j\le e^\lambda} Y_j^2 + \underbrace{2\sum_{j\le e^\lambda}\sum_{j<k\le e^\lambda} \mathbb{E}Y_j\mathbb{E}Y_k - \left(\sum_{j\le e^\lambda} \frac{1}{j}\right)^2}_{<0} \\ &\le \mathbb{E}\sum_{j\le e^\lambda} Y_j \le \lambda + 1 \end{aligned}$$

Note that Y_j, Y_k are pairwise independent for $j \neq k$.

d) By

$$\mathbb{E}T_3 = \sum_{j\le e^\lambda} \frac{\ln j}{j} > \int_2^{e^\lambda} \frac{\ln x}{x}\,dx - \frac{1}{e} = \frac{\lambda^2}{2} - \left(\frac{\ln^2 2}{2} + \frac{1}{e}\right)$$

we get the lower bound on $\mathbb{E}T_3$. Note that the function $\ln(x)/x$ has a maximum at e on $[2, \infty)$.

e) The upper bound for $\mathbb{E}T_3$ is calculated in the following way. Let

$$Z_j = \begin{cases} \ln j & : \quad \text{if} \quad w_j \in q, \\ 0 & : \quad \text{else.} \end{cases}$$

Thus $\mathbb{E}T_3 = \mathbb{E}\sum_{j\le e^\lambda} Z_j = \sum_{j\le e^\lambda} \frac{\ln j}{j}$. Let $2 < a < e^\lambda$ be an integer. Then

$$\sum_{j\le e^\lambda} \frac{\ln j}{j} \le \int_a^{e^\lambda} \frac{\ln x}{x}\,dx + \sum_{j=2}^{a} \frac{\ln j}{j} = \frac{\ln^2 e^\lambda}{2} - \frac{\ln^2 a}{2} + \sum_{j=2}^{a} \frac{\ln j}{j}\ .$$

Note that $\ln(x)/x$ has a maximum at e. For $a \geq 21$ the inequality

$$\sum_{j=2}^{a} \frac{\ln j}{j} \leq \frac{\ln^2 a}{2}$$

holds. Therefore,

$$\sum_{j \leq e^\lambda} \frac{\ln j}{j} \leq \frac{\ln^2 e^\lambda}{2} \ .$$

For $\lambda \geq 4$ the inequality $e^\lambda > 54$ holds. Thus, let $a = 21$ and $\lambda \geq 4$. Then $a < e^\lambda$ and by the above considerations $\mathbb{E}T_3 \leq \frac{\lambda^2}{2}$.

f) In a similar manner,

$$\mathrm{Var}T_3 = \mathbb{E}T_3^2 - (\mathbb{E}T_3)^2 \leq \sum_{j \leq e^\lambda} \frac{\ln^2 j}{j} \leq \frac{\lambda^3}{3}$$

if $a \geq 3416$, that is, if $\lambda \geq 9$ (note that $\ln^2(x)/x$ has a maximum at e^2).

g) Finally, let

$$W_j = \begin{cases} j & : \quad \text{if} \quad w_j \in q, \\ 0 & : \quad \text{else.} \end{cases}$$

We have $\mathbb{E}T_4 = \mathbb{E}\sum_{j \leq e^\lambda} W_j = \sum_{j \leq e^\lambda} 1 \leq e^\lambda$.

This completes the proof of Lemma 4.2.8.

Using Lemma 4.2.8, we can finish the proof of the second part of Theorem 4.2.7. Let the query q be randomly chosen according to the Zipf model. Then, by Markov's inequality (1.2)

$$\Pr\left(T_1 \geq \frac{\gamma\lambda}{e^\lambda}\right) \leq \frac{e^\lambda + 1}{e^\lambda \gamma\lambda} \ .$$

Therefore, as $\lambda \to \infty$ this probability tends to zero and the first regularity condition holds. The second regularity condition holds since by Chebyshev's inequality (1.3)

$$\begin{aligned} \Pr\left(T_2 \geq (1+\gamma)(\lambda+1)\right) &\overset{\mathbb{E}T_2 \leq \lambda+1}{\leq} \Pr\left(T_2 - \mathbb{E}T_2 \geq \gamma(\lambda+1)\right) \\ &\leq \frac{1}{\gamma^2(\lambda+1)} \end{aligned}$$

tends to zero as $\lambda \to \infty$. Note that T_2 takes only non-negative values. Thus, we only have to consider the case $T_2 \geq \lambda + 1$. Since $\frac{\lambda^2}{2} - C < \mathbb{E}T_3 \leq \frac{\lambda^2}{2}$ the third condition holds by applying Lemma 1.5.1. The fourth condition is true since by Markov's inequality the probability

$$\Pr\left(T_4 \geq \gamma\lambda e^{\lambda}\right) \leq \frac{1}{\gamma\lambda}$$

also tends to zero as $\lambda \to \infty$. Thus, all four regularity conditions hold and we are done.

□

Let us summarize the statement of Theorem 4.2.7. Consider a graph where the probability curves for r-match and r-prefix match are shown (Figure 4.2). Both curves have a similar structure: The probability is close to

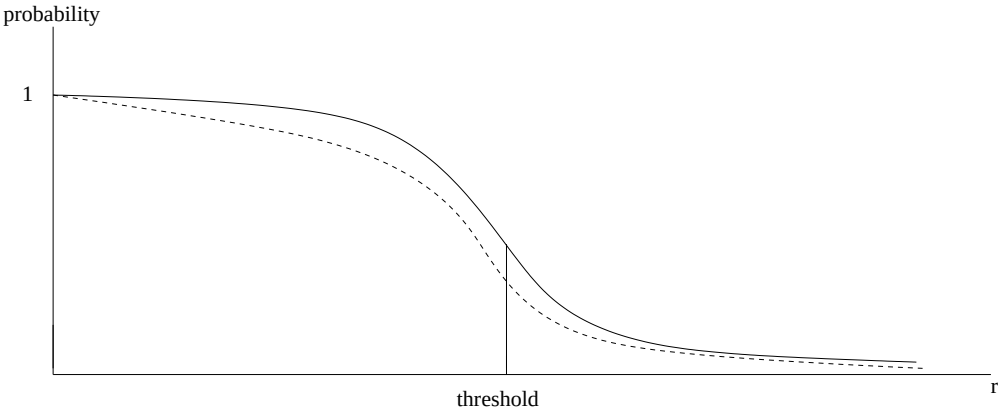


Figure 4.2: Exemplary probability curves for r-match (the solid line) and r-prefix match (the dashed line)

one for small r, but at some "threshold value" it falls to nearly zero. Now, Theorem 4.2.7 states that the threshold values of the two kinds of matches are close to each other. And this fact is extremely important, since it yields an efficient method for solving the approximate version of the maximal intersection problem: Compute a prefix match whose size is only slightly smaller ($\approx (1-\delta)s$) than the threshold values. According to the theorem, such a prefix match exists with high probability and, moreover, is with high probability almost optimal. This holds since the probability that there exists a match whose size is (significantly) larger ($> (1+\varepsilon)s$) than the threshold values is small. Thus, the above "threshold phenomenon" justifies to search only for a prefix match. In the next section we explain the corresponding algorithm in detail and show that its query time is quasi-linear in the sum of n and m.

4.3 The Algorithm

Using Theorem 4.2.7, we get the following result.

Theorem 4.3.1. *Let $0 < \delta < 1$ and $0 < \gamma < \frac{1}{5}$ be fixed. Let $m \in \omega(\log n)$ and $c = e^{-\delta^2/2}$.*[2] *Given a document collection $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ and a query document $q \subseteq \mathcal{W}$ both following the Zipf model. Then, there is an algorithm for the AMI problem which for sufficiently large n returns with probability greater than $1 - c^{\delta\sqrt{\ln n}}/(1-c) - 1/n^{5\gamma}$ at least a $\frac{1+\Gamma(\gamma,n)}{1-\Delta(\delta,n)}$-approximate answer, where $\Gamma(\gamma, n) \to 5\gamma$, $\Delta(\delta, n) \to \delta$ as $n \to \infty$. The algorithm has $\tilde{O}(nm)$ preprocessing time. The average case has $\tilde{O}(\log m + n)$ query time, while the worst case has a query time of $\tilde{O}(m + n)$. The storage required is $n^{1+o(1)}$.*

Proof. The algorithm is based on the usage of an inverted index. An inverted index is a data structure which for every word stores a list of the documents containing it. A query is answered by intersecting the document lists of the query words. Clearly, this approach only yields an answer if there exists a document in the database which contains (at least) the same words as the query. Thus, the problem is to choose the "right" query words for intersection. To solve this issue, we apply Theorem 4.2.7. Let $s = s_n = \sqrt{2 \ln n}$ and $\zeta = 2 + \lceil \delta s \rceil$. Since the theorem implies that a prefix match of size $\lfloor s \rfloor - \zeta$ is almost optimal, it is enough to intersect only the document lists of the *first* $\lfloor s \rfloor - \zeta$ query words. This can be done in the following way. We represent the document lists by a $\lfloor e^{\lfloor s \rfloor - 2} \rfloor \times n$ binary matrix, where the entry (i, j) is one if and only if the word w_i is contained in the document d_j (see Figure 4.3). Note that it suffices to use only the first $\lfloor e^{\lfloor s \rfloor - 2} \rfloor$ words of $\mathcal{W}$ as

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_{e^s} \end{array} \begin{array}{c} \begin{array}{ccccc} d_1 & d_2 & d_3 & \cdots & d_n \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 1 & \cdots & 0 \end{pmatrix} \end{array}$$

Figure 4.3: Binary matrix representing word-document relations

row indices since with probability greater than $1 - c^{1.3\,\delta\sqrt{\ln n}}/(1-c)$ the first $\lfloor s \rfloor - \zeta$ words of the query are among these words. Calculating the intersection of document lists reduces to a *binary and* of matrix rows. The answer is any document from this intersection. Note that the intersection is non-empty

[2] To be more precise, $m \geq \lfloor e^{\lfloor s \rfloor - 2} \rfloor$.

Preprocessing on $\mathcal{D} \subseteq 2^{\mathcal{W}}$
Generate a binary matrix representing word-document relations.

Processing a query $q \subseteq \mathcal{W}$
Intersect the matrix rows corresponding to the first query words.

Figure 4.4: Algorithm for the k-AMI problem in the Zipf model

with probability greater than $1-c^{\delta\sqrt{\ln n}}/(1-c)$. An overview of the algorithm is shown on Figure 4.4. In order to generate the matrix, we first sort the words of each document according to their index. This takes $O(nm \log m)$ time. Generating the matrix can be done in $C \cdot n \cdot n^{\varepsilon}$ time for all $\varepsilon > 0$, thus in $n^{1+o(1)}$ time (note that for all $\varepsilon > 0$ there exists a value n such that $e^s < n^{\varepsilon}$). Storing the matrix requires $n^{1+o(1)}$ space. For query processing, we sort the words of the query according to their index and then intersect the matrix rows belonging to the first $\lfloor s \rfloor - \zeta$ query words. In the average case this takes $O(\log m \log\log m + n\sqrt{\log n}) \subseteq \tilde{O}(\log m + n)$ time (in this case, a query contains approximately $\ln m$ words), and $O(m \log m + n\sqrt{\log n}) \subseteq \tilde{O}(m+n)$ time in the worst case. The $\sqrt{\log n}$ factor follows from the fact that we intersect $(\lfloor s \rfloor - \zeta) \in O(\sqrt{\log n})$ rows. The stated approximation factor (quality) is an upper bound of the quotient of the size of a maximal match and the size of a prefix match, which is returned by the above algorithm. Thus,

$$\frac{\lceil (1+5\gamma)s \rceil}{\lfloor s \rfloor - 2 - \lceil \delta s \rceil} \leq \frac{(1+5\gamma)s + 1}{s - 4 - \delta s} = \frac{1 + 5\gamma + s^{-1}}{1 - \delta - 4s^{-1}}$$

for sufficiently large n. The product

$$\left(1 - c^{\delta\sqrt{\ln n}}/(1-c)\right) \cdot \left(1 - 1/n^{5\gamma}\right) > 1 - c^{\delta\sqrt{\ln n}}/(1-c) - 1/n^{5\gamma}$$

is the probability that the returned prefix match is an approximate answer of at least this quality. This probability tends to one as $n \to \infty$. □

Remark 7. If we search for a document that has a *maximal* common prefix with the query (not "only" of size $\lfloor s \rfloor - \delta$), then the worst case query complexity changes to $\tilde{O}(m + nm)$ and the storage required is $O(nm)$. The average case and preprocessing complexity remains the same up to logarithmic factors.

In order to classify the complexity of the above algorithm, we compare it with the complexity of brute-force search. Brute-force search calculates the intersection size between the query and each document from the

database. If we sort the words of each document in a preprocessing step (which can be done in $O(nm \log m)$ time), then, for a document collection following the Zipf model, answering a query has an average case complexity of $O(\log m \log \log m + nm)$ and a worst case complexity of $O(m \log m + nm)$. Thus, the above algorithm performs faster in each case since we demand $m \geq \lfloor e^{\lfloor s \rfloor - 2} \rfloor$, that is, $m \in \omega(\log n)$. However, one should note that brute-force search returns an exact answer (i.e., a maximal match), while the above algorithm returns an approximate answer.

Since natural language texts are of particular interest for the Zipf model (for these texts, Zipf's law holds), we want to examine the complexity in this context in more detail. Beside Zipf's law, in natural language texts also *Heaps' law* holds [Hea78, MRS08]. This law predicts the vocabulary size of a text as a function of the text size. That is $v = C \cdot t^{\gamma}$ holds, where v denotes the vocabulary size, t the text size and C and $0 < \gamma < 1$ are constants depending on the text (typically, $10 \leq C \leq 100$ and $\gamma \approx 0.5$). Assuming that each document has length l and considering the concatenation of all documents as a (new) single document, we get $m = C \cdot (l \cdot n)^{\gamma} \in O(n^{\gamma})$. Thus, the preprocessing time of the above algorithm "reduces" to $O(n^{1+\gamma} \log n)$, and the average and worst case query time to $O(n\sqrt{\log n})$ respectively.

An important property of the above algorithm is the fact that it does not suffer from the curse of dimensionality (cf. Section 1.4). This holds since its space requirement is near-linear in n and its query time is quasi-linear in the sum of n and m, but not linear in $n \cdot m$.

4.3.1 A Different Approach

Instead on an inverted index, the algorithm can also be based on the usage of a tree. Therefore, we represent each document by its *index vectors*, that is a binary vector of length m such that position i is one if and only if the word w_i is contained in the document. The main idea is to store these vectors in a binary search tree such that each path corresponds to at least one vector (since duplicate documents are allowed, a path can correspond to several vectors). Then, one has to determine a path in this tree that matches $\lfloor s \rfloor - \zeta$ ones with a query. This is accomplished by applying a backtracking algorithm which traverses the tree in depth-first order. According to Theorem 4.2.7, it is enough to descend up to the depth $\lfloor e^{\lfloor s \rfloor - 2} \rfloor$ (thus, generating a tree based on the words w_1 up to $w_{\lfloor e^{\lfloor s \rfloor - 2} \rfloor}$ suffices; note that in this tree, different documents can correspond to the same path). Since a possible answer document must contain the first words of the query, but can also contain additional words, backtracking is only invoked at positions (levels) where the index vector of the query is zero. The algorithm stops when a path

is determined such that $\lfloor s \rfloor - \zeta$ ones of the query vector are matched (note that such a path exists with probability greater than $1 - c^{\delta\sqrt{\ln n}}/(1-c)$). The answer is any document from this path.

The document tree can be generated in $O(nm \log m)$ time and queries are answered in $O(\log m \log\log m + n^\theta)$ average and $O(m \log m + n^\theta)$ worst case query time for all $\theta > 1$ (note that $O(2^{e^s})$, the time backtracking requires in the worst case on a *complete* binary tree of depth e^s, is greater than $ne^s \in n^{1+o(1)}$). Hence, the asymptotic complexity of the tree-based approach is worse than that of the matrix-based approach.

Remark 8. Note that for documents from $2^{\mathcal{W}}$ following the Zipf model the frequency table is well-known. Namely, the word w_i is at rank i. For "real-life" document collections this is not the case. Thus, we have to generate the frequency table before we can apply the above algorithm. In Chapter 6 we deal with this task in more detail.

Remark 9. All statements presented in this chapter only hold for sufficiently large values of n and m. For example, the value of n must be (approximately) at least 10^{68} if $\delta = 0.7$ so that each theorem (respectively, lemma) holds. Then, Theorem 4.2.7 states that with probability greater than 0.46 there exists a prefix match of size 2. However, the values given by the statements are worst-case estimations. That is, the algorithm's performance (in terms of quality and running time) can be much better in practice than one would expect from the above statements. (For example, despite the fact that the satisfiability problem is NP-complete, today propositional formulas containing hundreds and even thousands of variables are efficiently tested for satisfiability, see e.g., [GW99].) Our experimental results presented in Chapter 6 indicate that this is indeed the case.

4.4 Generalization of the Zipf Model

In this section we generalize the threshold theorem of Section 4.2 in case the word frequencies are distributed according to a *Mandelbrot distribution*. Additionally, we derive a theoretical explanation for the so called *stop words*, that is, words which are too frequent and therefore appear to be of little value in reflecting the content of a document.

4.4.1 The Mandelbrot Distribution and Stop Words

According to Zipf's law, a rank-frequency plot of the words of a text on logarithmic axes should be a straight line with slope -1. However, most

empirical graphs differ from that line, especially for words of low and high ranks. To achieve a closer fit, Benoît Mandelbrot derived the more general formula

$$f = A \cdot (B + r)^{-\rho},$$

where A, B and ρ are parameters depending on the text [Man65, MS99]. If the rank-frequency correlation of the words of a text follows a formula of the above type, we say the words are distributed according to a *Mandelbrot distribution*[3]. In the following let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a finite set of words. We assume that a document collection is generated by the same process as in Section 4.2, but the probability of the word w_i, $i \in \{1, \ldots, m\}$ to be contained in a document is $\alpha/(\beta + i)$ for some constants $\alpha > 0$ and $\beta \geq 0$ (if $\alpha/(\beta + i) > 1$ then the probability for w_i to be contained in a document is set to 1). The expected frequency of w_i in the collection is $n \cdot \alpha/(\beta + i)$. Therefore, the word frequencies are distributed according to a Mandelbrot distribution with $A = n \cdot \alpha$, $B = \beta$ and $\rho = 1$. We call this process the *Mandelbrot model*. As mentioned before, stop words are words that occur very often, and, in general, do not carry meaning in natural language (e.g., a, the, by). The Mandelbrot model yields a natural explanation for stop words. Namely, the stop words are the set

$$\{w_i \mid i \leq \alpha - \beta\} ,$$

that is, words that occur in every document.

4.4.2 Thresholds in the Mandelbrot Model

In this section we give a statement about the probabilities of intersecting sets analogously to Theorem 4.2.7. If $\alpha - \beta \geq 1$, then in the Mandelbrot model stop words occur (namely, there are $\alpha - \beta$ stop words) and every stop word occurs in every document. Thus, stop words are irrelevant for the maximal intersection problem and can be excluded from further considerations. *Relevant* words are the words w_i where $i > \alpha - \beta$. For the remainder of this chapter we assume that α and β are integer numbers where $\alpha \geq \beta$ and $\alpha \geq 1$. The probability that a relevant word w_i is contained in a document is $\alpha/(\alpha + j)$ with $j = i - \alpha + \beta$. Note that $j \geq 1$ holds. By abuse of notation we define $\mathcal{W} := \{w_i \mid 1 \leq i \leq m_r\}$ to be the set of *relevant* words (i.e., without stop words)[4]. Thus, the probability that a relevant word $w_i \in \mathcal{W}$ is contained in a document is $\alpha/(\alpha + i)$. Note that by redefining $\mathcal{W}$ to be

[3]Sometimes also called Zipf-Mandelbrot distribution

[4]Note that $m = m_r + \alpha - \beta$.

the set of relevant words we ensure that in the following all documents in $2^{\mathcal{W}}$ contain only relevant words, and no stop words.

Remark 10. The set $2^{\mathcal{W}}$ yields a probability space where a document d is an event occurring with probability $\Pr(d) = \left(\prod_{w_i \in d} \frac{\alpha}{\alpha+i}\right)\left(\prod_{w_i \notin d} 1 - \frac{\alpha}{\alpha+i}\right)$.

As in the Zipf model, we introduce a threshold:

$$s = s_n := \sqrt{2\alpha \ln n}$$

For the following proofs, we partition the set $\mathcal{W}$ as follows (clearly, the group sizes depend on α).

Definition 4.4.1. We define

$$P_1 := \left\{ w_1 \, , \ldots , \, w_{\lfloor \alpha\left(\exp\left(\frac{1}{\alpha}+1\right)-1\right)+1 \rfloor} \right\}.$$

For $2 \leq i \leq \lfloor s \rfloor - 2\alpha - 1$ we define

$$P_i := \left\{ w_{\lceil \alpha\left(\exp\left(\frac{i-1}{\alpha}+1\right)-1\right)+1 \rceil} \, , \ldots , \, w_{\lfloor \alpha\left(\exp\left(\frac{i}{\alpha}+1\right)-1\right)+1 \rfloor} \right\}.$$

We say that a document $d \in 2^{\mathcal{W}}$ is *α-regular* if it contains exactly one word from each group P_i for all $i \in \{1, \ldots, \lfloor s \rfloor - 2\alpha - 1\}$ (in general, an α-regular document can contain words w_i with $i \geq \lceil \alpha(\exp(\frac{\lfloor s \rfloor - 2\alpha - 1}{\alpha} + 1) - 1) + 1 \rceil$).

Remark 11. Notice that only for $m_r \geq \lfloor \alpha \cdot \exp(\lfloor s \rfloor / \alpha - 1 - \frac{1}{\alpha}) - \alpha + 1 \rfloor$ there exist $\lfloor s \rfloor - 2\alpha - 1$ groups. Further notice that the groups do not overlap since $i/\alpha + 1$ is rational for $i \in \mathbb{N} \setminus \{0\}$, so $\exp(i/\alpha + 1)$ cannot be an integer [Wal69]. Obviously, the group P_1 is not empty. For $i \geq 2$ a group P_i is not empty since

$$\lceil \alpha \left(\exp((i-1)/\alpha + 1) - 1\right) + 1 \rceil \leq \lfloor \alpha \left(\exp\left(i/\alpha + 1\right) - 1\right) + 1 \rfloor .$$

This holds because

$$\alpha \left(\exp\left(\frac{i}{\alpha} + 1 \right) - 1 \right) + 1 - \alpha \left(\exp\left(\frac{i-1}{\alpha} + 1 \right) - 1 \right) - 1 \geq 1$$

is equivalent to

$$\exp\left(\frac{i}{\alpha} + 1 \right) \cdot \alpha \left(1 - \exp\left(-\frac{1}{\alpha} \right) \right) \geq 1 \ .$$

The last inequality is true since $\exp\left(\frac{i}{\alpha} + 1\right) > e > 2.7$ for $i \geq 1$, $g(\alpha) = \alpha(1 - \exp(-1/\alpha))$ is monotonically increasing on $[1, \infty)$, and $g(1) > 0.63$.

Definition 4.4.2. Let $0 < \delta < 1$. We say that a document $d \in 2^{\mathcal{W}}$ is (δ, α, n)-*generic* if for all $i \geq \delta s$ the following holds:

$$|d \cap (P_1 \cup \ldots \cup P_i)| \geq (1-\delta)i$$

This means that for all $i \geq \delta s$ a (δ, α, n)-generic document contains at least $(1-\delta)i$ words w_j with $j \leq \alpha \left(\exp\left(i/\alpha + 1\right) - 1\right) + 1$.

Lemma 4.4.3. *Let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. Let $d \in 2^{\mathcal{W}}$ be a random document following the Mandelbrot model. For sufficiently large n, m_r the probability that d is (δ, α, n)-generic is greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$.*

Proof. First, we consider a fixed i and let X be a random variable denoting the expected number of words in d up to the word $w_{\lfloor \alpha(\exp(i/\alpha+1)-1)+1 \rfloor}$. Then

$$\begin{aligned}
i &= \alpha(\ln \alpha + \frac{i}{\alpha} + 1 - \ln \alpha - 1) \\
&= \alpha \left(\ln \left(\alpha \cdot \exp\left(\frac{i}{\alpha} + 1 \right) \right) - \ln \alpha - 1 \right) \\
&< \alpha \left(\ln \left(\left\lfloor \alpha \cdot \exp\left(\frac{i}{\alpha} + 1 \right) \right\rfloor + 1 \right) - \ln \alpha - 1 \right) \\
&< \alpha \left(\sum_{k=1}^{\lfloor \alpha \cdot \exp(i/\alpha+1) \rfloor + 1} \frac{1}{k} \quad - \quad \sum_{k=1}^{\alpha} \frac{1}{k} \right) .
\end{aligned}$$

This holds since $\lfloor \alpha \cdot \exp\left(i/\alpha + 1\right) \rfloor + 1$ is an integer > 1 and therefore we can apply (1.8). Now

$$\begin{aligned}
& \alpha \left(\sum_{k=1}^{\lfloor \alpha \cdot \exp(i/\alpha+1) \rfloor + 1} \frac{1}{k} \quad - \quad \sum_{k=1}^{\alpha} \frac{1}{k} \right) \\
= \alpha & \sum_{k=\alpha+1}^{\lfloor \alpha \cdot \exp(i/\alpha+1) \rfloor + 1} \frac{1}{k} \\
= & \sum_{k=1}^{\lfloor \alpha(\exp(i/\alpha+1)-1)+1 \rfloor} \frac{\alpha}{\alpha + k} = \mathbb{E}X \ .
\end{aligned}$$

Overall, we have $i < \mathbb{E}X$. The Chernoff bound (1.4) yields that the probability that d contains less than $(1-\delta)i$ words up to the word $w_{\lfloor \alpha(\exp(i/\alpha+1)-1)+1 \rfloor}$ is smaller than $e^{-i\delta^2/2}$, since

$$\Pr\left(X \leq (1-\delta)i\right) \leq \Pr\left(X \leq (1-\delta)\mathbb{E}X\right) \leq e^{-\mathbb{E}X\delta^2/2} < e^{-i\delta^2/2}$$

for $i < \mathbb{E}X$. So the probability that d is not (δ, α, n)-generic for large n is bounded by

$$\begin{aligned}\sum_{i\geq 0} c^i - \sum_{i=0}^{\lfloor \delta\sqrt{2\alpha \ln n}\rfloor -1} c^i &= \sum_{i\geq \lfloor \delta\sqrt{2\alpha \ln n}\rfloor} c^i = c^{\lfloor \delta\sqrt{2\alpha \ln n}\rfloor} \cdot \sum_{i\geq 0} c^i \\ &= \frac{c^{\lfloor \delta\sqrt{2\alpha \ln n}\rfloor}}{1-c} \\ &< \frac{c^{\sqrt{2}\,\delta\sqrt{\alpha \ln n}-1}}{1-c} \\ &< \frac{c^{1.3\,\delta\sqrt{\alpha \ln n}}}{1-c}\ .\end{aligned}$$

This holds because $1.3 < \sqrt{2}$ and n is large. Overall, the probability that for all $i \geq \delta s$ the document d contains at least $(1-\delta)i$ words w_j with $j \leq \alpha\left(\exp\left(i/\alpha + 1\right) - 1\right) + 1$ is greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$. Note that we need m_r to be sufficiently large so that at least

$$\left\lfloor \alpha\left(\exp\left(\frac{\lceil \delta s\rceil}{\alpha} + 1\right) - 1\right) + 1 \right\rfloor$$

words exist. □

Lemma 4.4.4. *Let $d \in 2^{\mathcal{W}}$ be a random document following the Mandelbrot model and let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. For sufficiently large n, m_r the following holds: If we insert the first (regarding the word order according to indices) $\lceil \delta s\rceil$ missing words to d then*

$$\Pr\left(\forall i \leq s:\ |d \cap (P_1 \cup \ldots \cup P_i)| \geq i\right) > 1 - \frac{c^{1.3\,\delta\sqrt{\alpha \ln n}}}{1-c} \tag{4.3}$$

Proof. For all $i \geq \delta s$ Lemma 4.4.3 states that for sufficiently large n, m_r the probability that d is (δ, α, n)-generic is greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$. Since $\delta i \leq \lceil \delta s\rceil$ for all $i \leq s$ and since we insert the first $\lceil \delta s\rceil$ missing words to d, for sufficiently large n, m_r the probability that $|d \cap (P_1 \cup \ldots \cup P_i)| \geq i$ holds for all $\delta s \leq i \leq s$ is also greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$. Note that by inserting the first $\lceil \delta s\rceil$ missing words to d, also for all $i \leq \delta s$ it is true that $|d \cap (P_1 \cup \ldots \cup P_i)| \geq i$. Overall, inequality (4.3) holds. □

Next follows our main theorem for the Mandelbrot model.

Theorem 4.4.5 (Threshold theorem for the Mandelbrot model)**.** *Let* $m_r \geq \lfloor \alpha \cdot e^{\lfloor s \rfloor/\alpha - 1 - 1/\alpha} - \alpha + 1 \rfloor$ *and let* $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ *be a document collection following the Mandelbrot model.*

1. *Let* $0 < \delta < 1$ *be fixed. Let* $\zeta = 2\alpha + 1 + \lceil \delta s \rceil$ *and* $c = e^{-\delta^2/2}$*. Given a query document* $q \subseteq \mathcal{W}$ *following the Mandelbrot model. Then for sufficiently large* n *the following holds: The probability that there exists a* $(\lfloor s \rfloor - \zeta)$-prefix *match is greater than* $1 - c^{\delta\sqrt{\alpha \ln n}}/(1-c)$*. Thus, the probability tends to one as* $n \to \infty$*.*

2. *Let* $\varepsilon > 0$ *be fixed. Given a query document* $q \subseteq \mathcal{W}$ *following the Mandelbrot model. Then the probability that there exists a* $\lceil (1+\varepsilon)\sqrt{\alpha} s \rceil$*-match tends to zero as* $n \to \infty$*.*

Proof. 1. Let $d_R \in 2^{\mathcal{W}}$ be a fixed α-regular document (Definition 4.4.1). The probability that a random document from $2^{\mathcal{W}}$ following the Mandelbrot model contains the first $\lfloor s \rfloor - 2\alpha - 1$ words of d_R is at least

$$\frac{\alpha}{\left\lfloor \alpha \left(\exp\left(\frac{1}{\alpha} + 1\right) - 1 \right) + 1 \right\rfloor + \alpha} \cdot \ldots \cdot \frac{\alpha}{\left\lfloor \alpha \left(\exp\left(\frac{\lfloor s \rfloor - 2\alpha - 1}{\alpha} + 1 \right) - 1 \right) + 1 \right\rfloor + \alpha}.$$

This probability is greater than

$$\frac{\alpha}{\alpha \cdot \exp\left(1/\alpha + 1\right) + 1} \cdot \ldots \cdot \frac{\alpha}{\alpha \cdot \exp\left((\lfloor s \rfloor - 2\alpha - 1)/\alpha + 1\right) + 1} > \frac{1}{\exp\left(1/\alpha + 3/2\right)} \cdot \ldots \cdot \frac{1}{\exp\left((\lfloor s \rfloor - 2\alpha - 1)/\alpha + 3/2\right)}.$$

This inequality holds since

$$\exp\left(\frac{i}{\alpha} + 1\right) + 1/\alpha < \exp\left(\frac{i}{\alpha} + 1\right) \cdot \sqrt{e}$$

for $i \geq 1$ and $\alpha \geq 1$. The last product is equal to

$$\frac{1}{\exp\left(1/\alpha \sum_{i=1}^{\lfloor s \rfloor - 2\alpha - 1} i + (\lfloor s \rfloor - 2\alpha - 1) \cdot 3/2\right)} >$$

$$\frac{1}{\exp\left(1/\alpha \cdot (\lfloor s \rfloor - 2\alpha)^2/2 + (\lfloor s \rfloor - 2\alpha - 1) \cdot 3/2\right)} >$$

$$\frac{1}{\exp\left(1/\alpha \cdot (s - 2\alpha)^2/2 + (s - 2\alpha - 1) \cdot 3/2\right)} =$$

$$\frac{\exp\left(1/2s + \alpha + 3/2\right)}{n} .$$

Note that

$$\exp\left(\frac{1}{2}s + \alpha + \frac{3}{2}\right) < n$$

since the probability that a document contains the first $\lfloor s \rfloor - 2\alpha - 1$ words of d_R is less than one. This means that the probability that there exists no document in $\mathcal{D}$ containing these first words is no more than

$$\left(1 - \frac{\exp\left(1/2s + \alpha + 3/2\right)}{n}\right)^n < \exp\left(-\exp\left(\frac{1}{2}s + \alpha + \frac{3}{2}\right)\right) ,$$

which follows from inequality (1.6). So with probability greater than

$$1 - \exp\left(-\exp\left(\frac{1}{2}s + \alpha + \frac{3}{2}\right)\right)$$

there exists a document in $\mathcal{D}$ having the first $\lfloor s \rfloor - 2\alpha - 1$ words of d_R. Consider the query q. Assume we insert the *first* $\lceil \delta s \rceil$ missing words to q. Then, Lemma 4.4.4 implies that for large n the probability that for each $i \leq s$ the query q contains at least i words w_j with $j \leq \alpha(\exp(i/\alpha + 1) - 1) + 1$ is greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$. This means that the probability that for each $i \leq s$ the query contains at least as many words as d_R among the first i groups is also greater than $1 - c^{1.3\,\delta\sqrt{\alpha \ln n}}/(1-c)$ (see Figure 4.1). Thus, the probability that there exists a document in $\mathcal{D}$ matching the first $\lfloor s \rfloor - 2\alpha - 1$ words of the query q is greater than

$$\left(1 - \frac{c^{1.3\,\delta\sqrt{\alpha \ln n}}}{1-c}\right)\left(1 - \exp\left(-\exp\left(\frac{1}{2}s + \alpha + \frac{3}{2}\right)\right)\right) .$$

For large n, this product is at least $1 - c^{\delta\sqrt{\alpha \ln n}}/(1-c)$. It remains to notice that by removing the imaginary inserted $\lceil \delta s \rceil$ words from q we still match $\lfloor s \rfloor - 2\alpha - 1 - \lceil \delta s \rceil = \lfloor s \rfloor - \zeta$ words and we are done.

2. At first we consider the query q as a fixed document and let d be a random document following the Mandelbrot model. Again, the idea is to evaluate the Laplace transform (with parameter $\lambda > 0$) of the intersection size between q and d

$$\begin{aligned}
\mathbb{E}\exp\left(\lambda |q \cap d|\right) &= \prod_{j:w_j \in q} \left(1 - \frac{\alpha}{\alpha+j} + \frac{\alpha}{\alpha+j} e^{\lambda}\right) \\
&\leq \prod_{j:w_j \in q} \left(1 + \frac{\alpha}{\alpha+j} e^{\lambda}\right) \\
&= \prod_{\substack{j:w_j \in q \\ \alpha+j > \alpha e^{\lambda}}} \left(1 + \frac{\alpha e^{\lambda}}{\alpha+j}\right) \cdot \prod_{\substack{j:w_j \in q \\ \alpha+j \leq \alpha e^{\lambda}}} \frac{\alpha e^{\lambda}}{\alpha+j} \left(1 + \frac{\alpha+j}{\alpha e^{\lambda}}\right).
\end{aligned}$$

It follows that

$$\begin{aligned}
\ln \mathbb{E}\exp\left(\lambda |q \cap d|\right) &\leq \sum_{\substack{j:w_j \in q \\ \alpha+j \geq \alpha e^{\lambda}}} \frac{\alpha e^{\lambda}}{\alpha+j} + \sum_{\substack{j:w_j \in q \\ \alpha+j \leq \alpha e^{\lambda}}} \left(\ln\left(\alpha e^{\lambda}\right) - \ln(\alpha+j)\right) \\
&\quad + \sum_{\substack{j:w_j \in q \\ \alpha+j \leq \alpha e^{\lambda}}} \frac{\alpha+j}{\alpha e^{\lambda}} \\
&= \alpha e^{\lambda} T_1 + (\ln \alpha + \lambda) T_2 - T_3 + \frac{1}{\alpha e^{\lambda}} T_4,
\end{aligned}$$

where

$$T_1 = \sum_{\substack{j:w_j \in q \\ \alpha+j \geq \alpha e^{\lambda}}} \frac{1}{\alpha+j}, \qquad T_2 = \left|q \cap \left\{w_j \mid \alpha + j \leq \alpha e^{\lambda}\right\}\right|,$$

$$T_3 = \sum_{\substack{j:w_j \in q \\ \alpha+j \leq \alpha e^{\lambda}}} \ln(\alpha+j), \qquad T_4 = \sum_{\substack{j:w_j \in q \\ \alpha+j \leq \alpha e^{\lambda}}} \alpha + j.$$

Let $0 < \gamma < \frac{1}{4}$. We will assume that the following four *regularity conditions* are verified. (In fact, the probability that each of these conditions is true tends to one if we consider q as a random document

following the Mandelbrot model and let $\lambda \to \infty$; the proof follows later.)

$$\alpha e^{\lambda} T_1 \leq \gamma\lambda, \qquad T_2 \leq (1+\gamma)\alpha\lambda,$$

$$T_3 \geq (1-\gamma)\alpha \left(\frac{\ln^2 \left(\alpha e^{\lambda}\right)}{2} \right), \qquad \frac{1}{\alpha e^{\lambda}} T_4 \leq \gamma\lambda.$$

Under these regularity conditions we obtain for sufficiently large λ

$$\begin{aligned}
\ln \mathbb{E}\exp\left(\lambda |q \cap d|\right) &\leq 2\gamma\lambda + (\ln\alpha + \lambda)(1+\gamma)\alpha\lambda - (1-\gamma)\alpha \left(\frac{\ln^2(\alpha e^{\lambda})}{2} \right) \\
&= 2\gamma\lambda + (\ln\alpha + \lambda)(1+\gamma)\alpha\lambda \\
&\qquad + (\gamma - 1)\alpha \left(\frac{(\ln\alpha + \lambda)^2}{2} \right) \\
&\leq (1+3\gamma)\alpha\frac{\lambda^2}{2} + \underbrace{2\gamma\lambda(1+\alpha\ln\alpha)}_{\leq \gamma\alpha\frac{\lambda^2}{2}} \\
&\leq (1+4\gamma)\alpha\frac{\lambda^2}{2} \ .
\end{aligned}$$

The following argumentation is (except for the implications of α) almost identical to the analogous part in the proof of Theorem 4.2.7. Assume that $\{d_j \mid 1 \leq j \leq n\}$ is a sample of n independent documents distributed according to the Mandelbrot model. Then by the exponential Chebyshev inequality (1.5), for any $r > 0$ we have

$$\begin{aligned}
\Pr\left(\max_{j \leq n} |q \cap d_j| \geq r \right) &\leq n \ \Pr\left(|q \cap d| \geq r\right) \\
&\leq n \ \frac{\mathbb{E}\exp\left(\lambda |q \cap d|\right)}{e^{\lambda r}} \\
&\leq n \ \exp\left((1+4\gamma)\alpha\frac{\lambda^2}{2} - \lambda r \right).
\end{aligned}$$

By choosing

$$r = (1+4\gamma)\alpha\sqrt{2\ln n}$$

and

$$\lambda = \sqrt{2\ln n}$$

we obtain $(1+4\gamma)\alpha\lambda^2 = \lambda r$, hence

$$\begin{aligned}\Pr\left(\max_{j\le n}|q\cap d_j| \ge r\right) &\le n\,\exp\left(-(1+4\gamma)\alpha\frac{\lambda^2}{2}\right)\\ &= n\,\exp\Big(-\ln n - \Big((1+4\gamma)\alpha - 1\Big)\ln n\Big)\\ &= \frac{1}{n^{(1+4\gamma)\alpha-1}} \;\to 0\end{aligned}$$

for $n \to \infty$, as required (note that $n \to \infty$ implies $\lambda \to \infty$). Part 2 of Theorem 4.4.5 follows as $r = (1+4\gamma)\alpha\sqrt{2\ln n} = (1+\varepsilon)\sqrt{\alpha}\cdot s$ for $\varepsilon = 4\gamma$.

Finally, we present the regularity conditions proof. Therefore, let the query q be randomly chosen according to the Mandelbrot model. We need the following lemma.

Lemma 4.4.6. *For sufficiently large λ the following inequalities hold:*

a) $\mathbb{E}T_1 \le 2e^{-\lambda}$

b) $\mathbb{E}T_2 \le \alpha\lambda$

c) $\mathrm{Var}T_2 \le \alpha\lambda$

d) $\alpha\frac{\ln^2(\alpha e^\lambda)}{2} - \alpha\left(\frac{\ln^2(\alpha+1)}{2} + \frac{1}{e}\right) < \mathbb{E}T_3$

e) $\mathbb{E}T_3 \le \alpha\frac{\ln^2(\alpha e^\lambda)}{2}$

f) $\mathrm{Var}T_3 \le \alpha\frac{\ln^3(\alpha e^\lambda)}{3}$

g) $\mathbb{E}T_4 < \alpha^2 e^\lambda$

Proof. The proof is similar to the proof of Lemma 4.2.8. Thus, we only present the calculations of expectations and variances.

a)

$$\begin{aligned}\mathbb{E}T_1 &= \sum_{j:\alpha+j\ge\alpha e^\lambda}\frac{\alpha}{(\alpha+j)^2} = \alpha\sum_{k\ge\alpha e^\lambda}\frac{1}{k^2}\\ &\le \lim_{B\to\infty} 2\alpha\int_{\alpha e^\lambda}^{B}\frac{1}{x^2}\,dx = 2\alpha\cdot\frac{1}{\alpha e^\lambda} = 2e^{-\lambda}.\end{aligned}$$

b) Let

$$X_j = \begin{cases} 1 & : \text{ if } \quad w_j \in q,\\ 0 & : \text{ else.}\end{cases}$$

Now,

$$\mathbb{E}T_2 = \mathbb{E}\sum_{j:\alpha+j\le\alpha e^\lambda} X_j = \sum_{j:\alpha+j\le\alpha e^\lambda} \frac{\alpha}{\alpha+j} = \alpha\sum_{k=\alpha+1}^{\lfloor\alpha e^\lambda\rfloor}\frac{1}{k} \le \alpha\int_\alpha^{\alpha e^\lambda}\frac{1}{x}\,dx$$
$$= \alpha\left(\ln\left(\alpha e^\lambda\right) - \ln\alpha\right) = \alpha\lambda\ .$$

c) The variance of T_2 is calculated as follows:

$$\mathrm{Var}T_2 = \mathbb{E}T_2^2 - (\mathbb{E}T_2)^2 = \mathbb{E}\left(\sum_{j:\alpha+j\le\alpha e^\lambda} X_j\right)^2 - \left(\sum_{j:\alpha+j\le\alpha e^\lambda}\frac{\alpha}{\alpha+j}\right)^2$$
$$= \mathbb{E}\sum_{j:\alpha+j\le\alpha e^\lambda} X_j^2$$
$$+ \underbrace{2\sum_{j:\alpha+j\le\alpha e^\lambda}\ \sum_{k:\alpha+j<\alpha+k\le\alpha e^\lambda}\mathbb{E}X_j\mathbb{E}X_k - \left(\sum_{j:\alpha+j\le\alpha e^\lambda}\frac{\alpha}{\alpha+j}\right)^2}_{<0}$$
$$\le \mathbb{E}\sum_{j:\alpha+j\le\alpha e^\lambda} X_j \le \alpha\lambda$$

Note that X_j, X_k are pairwise independent for $j \neq k$.

d) The lower bound on $\mathbb{E}T_3$ is

$$\mathbb{E}T_3 = \sum_{j:\alpha+j\le\alpha e^\lambda}\frac{\alpha\ln(\alpha+j)}{\alpha+j} = \alpha\sum_{k=\alpha+1}^{\lfloor\alpha e^\lambda\rfloor}\frac{\ln k}{k}$$
$$> \alpha\left(\int_{\alpha+1}^{\alpha e^\lambda}\frac{\ln x}{x}\,dx - \frac{1}{e}\right) = \alpha\frac{\ln^2(\alpha e^\lambda)}{2} - \alpha\left(\frac{\ln^2(\alpha+1)}{2} + \frac{1}{e}\right).$$

The integral is at most as large as the sum since $\ln(x)/x$ has a maximum at e on $[2,\infty)$.

e) Let $\alpha+1 < b < \alpha e^\lambda$ be an integer. Then

$$\mathbb{E}T_3 = \sum_{j:\alpha+j\le\alpha e^\lambda}\frac{\alpha\ln(\alpha+j)}{\alpha+j} = \alpha\sum_{k=\alpha+1}^{\lfloor\alpha e^\lambda\rfloor}\frac{\ln k}{k} \le$$
$$\le \alpha\left(\int_b^{\alpha e^\lambda}\frac{\ln x}{x}\,dx + \sum_{k=\alpha+1}^{b}\frac{\ln k}{k}\right) \le \alpha\frac{\ln^2(\alpha e^\lambda)}{2}\ .$$

The last inequality holds for sufficiently large λ, see the proof of Lemma 4.2.8.

f) The upper bound on $\mathrm{Var}T_3$ is calculated by the same techniques as the upper bounds on $\mathrm{Var}T_2$ and $\mathbb{E}T_3$:

$$\mathrm{Var}T_3 \leq \sum_{j:\alpha+j\leq\alpha e^{\lambda}} \frac{\alpha \ln^2(\alpha+j)}{\alpha+j} \leq \alpha \left(\frac{\ln^3(\alpha e^{\lambda})}{3}\right)$$

for sufficiently large λ, see the proof of Lemma 4.2.8.

g) Finally,

$$\mathbb{E}T_4 = \sum_{j:\alpha+j\leq\alpha e^{\lambda}} \alpha \leq \alpha\left(\alpha e^{\lambda} - \alpha\right) < \alpha^2 e^{\lambda}.$$

This completes the proof of Lemma 4.4.6.

By Lemma 4.4.6 and Lemma 1.5.1 it follows by the same argumentation as in the proof of Theorem 4.2.7 that for a query q randomly chosen according to the Mandelbrot model the four regularity conditions hold as $\lambda \to \infty$.

□

4.4.3 Discussion

In the previous section, we transferred the threshold theorem of the Zipf model to the Mandelbrot model. Like the Mandelbrot model itself, the threshold depends on the parameter α. As α increases, the size of a prefix match and the upper bound on the intersection size increase as well. Contrary to the Zipf model, these two values are not "close" in the Mandelbrot model. They differ about a factor of $\sqrt{\alpha}$. This is due to the fact that α affects the probability if a word is contained in a document. As α increases, for *all* words this probability increases as well. Thus, it seems that considering only the "first" words of a query does not suffice for determining a match of almost maximal size. However, our experimental results indicate that in real applications this fact is not as critical as one may expect.

As indicated in Section 4.4.1, Mandelbrot's formula $f = A \cdot (B+r)^{-\rho}$ is a generalization of Zipf's law (just let $B = 0$ and $\rho = 1$, then Mandelbrot's formula degenerates to Zipf's law). Analogously, the Mandelbrot model is a generalization of the Zipf model, or in other words, the Zipf model is a special case of the Mandelbrot model. Let $\alpha = 1$ and $\beta = 0$, then the formula $\alpha/(\beta+i)$ degenerates to $1/i$. And indeed, the intersection sizes for a prefix match stated in Theorem 4.2.7 and 4.4.5 differ only by one. This difference is due to the fact that, in contrast to the Zipf model, in the Mandelbrot model we exclude stop words, and for $\alpha = 1$, $\beta = 0$ there exists exactly one stop word. As expected, the upper bounds coincide.

4.4.4 The Algorithm

Given Theorem 4.4.5, we get the following result.

Theorem 4.4.7. *Let $0 < \delta < 1$ and $0 < \gamma < \frac{1}{4}$ be fixed. Let $m \in \omega(\log n)$ and $c = e^{-\delta^2/2}$.*[5] *Given a document collection $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq 2^{\mathcal{W}}$ and a query document $q \subseteq \mathcal{W}$ both following the Mandelbrot model. Then, there is an algorithm for the AMI problem which for sufficiently large n returns with probability greater than $1 - c^{\delta\sqrt{\alpha \ln n}}/(1-c) - 1/n^{(1+4\gamma)\alpha-1}$ at least a $\sqrt{\alpha} \cdot \frac{1+\Gamma(\gamma,n)}{1-\Delta(\delta,n)}$-approximate answer, where $\Gamma(\gamma, n) \to 4\gamma$, $\Delta(\delta, n) \to \delta$ as $n \to \infty$. The algorithm has $\tilde{O}(nm)$ preprocessing time. The average case has $\tilde{O}(\log m + n)$ query time, while the worst case has a query time of $\tilde{O}(m+n)$. The storage required is $n^{1+o(1)}$.*

Proof. Based on Theorem 4.4.5 we apply the same algorithm as in the Zipf model. Let $s = s_n = \sqrt{2\alpha \ln n}$. The size of the matrix must be adapted to $\lfloor \alpha(\exp(\lfloor s \rfloor/\alpha - 1 - 1/\alpha) - 1) + 1 \rfloor \times n$. Since in the Mandelbrot model the average document length is also in $O(\log m)$, and for each $\varepsilon > 0$ there exists a value n such that $e^{s/\alpha} < n^{\varepsilon}$, the complexity of the algorithm remains the same. The approximation factor follows since

$$\frac{\lceil (1+4\gamma)\sqrt{\alpha} s \rceil}{\lfloor s \rfloor - 2\alpha - 1 - \lceil \delta s \rceil} \leq \frac{(1+4\gamma)\sqrt{\alpha} s + 1}{s - 2\alpha - 3 - \delta s} = \frac{(1+4\gamma)\sqrt{\alpha} + s^{-1}}{1 - (2\alpha+3)s^{-1} - \delta}$$

for sufficiently large n. Note that the probability that the algorithm returns an answer of at least this quality tends to one as $n \to \infty$. □

Remark 12. All statements in this chapter were stated under the assumption that no document contains stop words. Clearly, in a real setting we have to remove stop words before we can apply the above algorithm. In Chapter 6 we discuss this issue in more detail.

[5] To be more precise, $m \geq \lfloor \alpha \cdot e^{\lfloor s \rfloor/\alpha - 1 - 1/\alpha} \rfloor + 1 - \beta$.

Chapter 5

Maximal Intersection Queries in the Hierarchical Schemes

In this chapter we investigate a second input distribution. Again, we show that a threshold theorem holds. Based on this theorem we state a deterministic algorithm solving the approximate maximal intersection problem in query time which is polylogarithmic in the database size.

5.1 Preliminaries

As in Chapter 4, we deal with *documents* and *words* instead of sets and elements. However, the results hold for every input following the model presented subsequently.

5.2 The Model

Let $k > 2$ be an integer and let $\mathcal{W} = \{w_1, \ldots, w_m\}$ be a finite set of $m = (2^k - 1) \cdot k$ different words. We assume that $n = 2^k$ holds. A document collection $\mathcal{D}$ consists of n documents where every document $d \in \mathcal{D}$ is an element of $2^{\mathcal{W}}$ with $|d| = k$. A *hierarchical scheme* H of height k is a table with k levels, level 1 to level k. For $1 \leq i \leq k$ level i is divided into 2^{i-1} cells, cell $C_{i,1}$ to cell $C_{i,2^{i-1}}$. For $2 \leq l \leq k$ we say that cell $C_{l-1,j}$, $1 \leq j \leq 2^{l-2}$, is *above* cell $C_{l,j'}$, $1 \leq j' \leq 2^{l-1}$, if and only if $\lceil j'/2 \rceil = j$. Every cell contains k words. Each word occurs in exactly one cell. A document collection based on a hierarchical scheme H can be generated as follows: Every document is generated independently. To generate a document, one chooses a random cell on level k and marks it. Then, for $l = k, \ldots, 2$ one marks the cell on level $l-1$ that is above the already marked cell on level l. Finally, one chooses a random

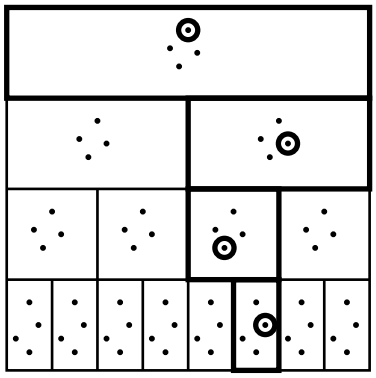


Figure 5.1: A hierarchical scheme for $k = 4$

word in every marked cell. Note that a document collection generated by the above process (or, *following a hierarchical scheme* H) can contain documents d_i, d_j with $d_i = d_j$ for $i \neq j$. Clearly, every document corresponds to a unique sequence of k cells. We call such a sequence a *cell path*. In a hierarchical scheme, at most $k \cdot (2k)^{k-1}$ different documents can exist. In Figure 5.1 a hierarchical scheme for $k = 4$ and a document with corresponding cell path is depicted.

Remark 13. The hierarchical schemes are motivated by the hierarchical structure of the *domain name system*. Consider for example the web address `http://www.fmi.uni-stuttgart.de`. Then, "`de`" would be located at the cell on level 1, "`uni-stuttgart`" at a cell on level 2 and so on.

Remark 14. Zipf's law can be observed in a hierarchical scheme. To be more precise, the following holds: For all words it holds that the product of expected frequency and expected rank is approximately the same. Indeed, the expected frequency of a word on level i is given by the formula $2^k/(2^{i-1} \cdot k)$. The expected rank of such a word is given by the formula $(2^{i-1}-1) \cdot k + 2^{i-2} \cdot k$. Hence, the product between frequency and frequency rank (divided by 2^k) is equal to

$$\frac{2^k}{2^{i-1} \cdot k} \cdot \left(\frac{3}{2} \cdot 2^{i-1} - 1 \right) \cdot \frac{k}{2^k} = \frac{3}{2} - \frac{1}{2^{i-1}},$$

which means for all $i \geq 1$ the value lies in the interval $[0.5, 1.5)$.

Definition 5.2.1. Let $r \geq 0$. Given a document collection $\mathcal{D}$ and a query document q. By *r-match* we denote a document from the collection $\mathcal{D}$ that contains at least r words of q.

Definition 5.2.2. Let $r \geq 0$. Given a document collection $\mathcal{D}$ and a query document q both following a hierarchical scheme H. Assume that the words

of q are ordered according to their respective level in H. By *r-prefix match* we denote a document from the collection $\mathcal{D}$ that contains at least the first r words of q.

This time we introduce two thresholds to give statements about the most probable size of a maximal intersection:

$$s = \frac{k}{1+\log k} \qquad \text{and} \qquad s' = \frac{k}{\log k}\ .$$

Theorem 5.2.3 (Threshold theorem for the hierarchical schemes)**.** *Let $k \geq 9$ be an integer and $2 \leq \gamma < s$. Let $\mathcal{D}$ be a document collection following a hierarchical scheme H.*

1. *Given a query document following H. Then the probability that there exists a $\lfloor s-\gamma \rfloor$-*prefix *match is greater than $1-2^{-(2k)^\gamma}$. Thus, the probability tends to one as $n \to \infty$.*

2. *Given a query document following H. Then the probability that there exists a $\lceil s'+\gamma \rceil$-match is smaller than $2/k^{\gamma-1}$, that is, tends to zero as $n \to \infty$.*

Proof. 1. The number of different prefixes of length $\lfloor s-\gamma \rfloor$ in H is at most

$$k(2k)^{s-\gamma-1} < 2^{(1+\log k)(s-\gamma)} = 2^{(1+\log k)\left(k/(1+\log k)-\gamma\right)} = 2^k \cdot (2k)^{-\gamma}.$$

So the probability that a random document from $2^{\mathcal{W}}$ following H does not match a prefix of length $\lfloor s-\gamma \rfloor$ of any document from $\mathcal{D}$ is smaller than

$$\left(1-\frac{(2k)^\gamma}{2^k}\right)^{2^k} < e^{-(2k)^\gamma} < 2^{-(2k)^\gamma}.$$

This inequality follows from inequality (1.6) since $(2k)^\gamma \leq 2^k$ holds for $2 \leq \gamma < s$ and $k \geq 9$. Therefore, the probability that there exists a document in $\mathcal{D}$ with the same prefix as q of length $\lfloor s-\gamma \rfloor$ is greater than $1-2^{-(2k)^\gamma}$.

2. Let d be a random document from $2^{\mathcal{W}}$ following H. Let $t \geq \lceil s'+\gamma \rceil$ be the last level where the words of d and q match. We want to estimate the probability that q matches at least $\lceil s'+\gamma \rceil$ words at arbitrary positions with d. The probability that d and q correspond to the same cell path up to level t is 2^{1-t}. The probability that at least $\lceil s'+\gamma \rceil$ words

are matched at some fixed cells is at most $1/k^{\lceil s'+\gamma \rceil} \cdot ((k-1)/k)^{t-\lceil s'+\gamma \rceil}$. An upper bound for the number of different possibilities of choosing at least $\lceil s'+\gamma \rceil$ out of t cells is 2^t (consider $\sum_j \binom{t}{j} = 2^t$). The factor $((k-1)/k)^{t-\lceil s'+\gamma \rceil}$ is smaller than one. Overall, the probability that q matches at least $\lceil s'+\gamma \rceil$ words at arbitrary positions with d is smaller than

$$k \cdot 2^t \cdot \left(\frac{1}{k}\right)^{\lceil s'+\gamma \rceil} \cdot 2^{1-t} \;=\; 2 \cdot k \cdot \left(\frac{1}{k}\right)^{\lceil s'+\gamma \rceil} \;=\; 2 \cdot \left(\frac{1}{k}\right)^{\lceil s'+\gamma \rceil - 1}.$$

The factor k in the above equation arises from the fact that we need to consider all possible levels for the last matched position t. Thus, the probability that no document matches at least $\lceil s'+\gamma \rceil$ words at arbitrary positions with q is at least

$$\left(1 - 2 \cdot \left(\frac{1}{k}\right)^{\lceil s'+\gamma \rceil - 1}\right)^{2^k} \geq \left(1 - 2 \cdot \left(\frac{1}{k}\right)^{s'+\gamma-1}\right)^{2^k} = \left(1 - \frac{2}{2^k \cdot k^{\gamma-1}}\right)^{2^k} \geq 1 - \frac{2}{k^{\gamma-1}},$$

which follows from inequality (1.7) (note that (1.7) can be applied since $2^k \geq 1$ and $k^{\gamma-1}/2 \geq 1$ holds for $k \geq 9$ and $\gamma \geq 2$). So the probability that there exists a document in $\mathcal{D}$ that matches at least $\lceil s'+\gamma \rceil$ words of q is smaller than $2/k^{\gamma-1}$.

□

5.3 The Algorithm

Let $\gamma = 2$. Applying Theorem 5.2.3, we get the following result.

Theorem 5.3.1. *Let* H *be a hierarchical scheme. Given a document collection* $\mathcal{D} = \{d_1, \ldots, d_n\} \subset 2^{\mathcal{W}}$ *and a query document* $q \subseteq \mathcal{W}$ *both following* H*. Then, there exists an algorithm for the AMI problem which for sufficiently large* n, m *returns with probability greater than* $1 - 1/n^{4 \log n} - 2/\log n$ *at least a* $2 \cdot \frac{1+\varepsilon_n}{1-\varepsilon_n}$*-approximate answer, where* $\varepsilon_n > 0$*. The algorithm has a preprocessing time of* $\tilde{O}(n)$*, a query time of* $O(\log^2 n)$*, and requires* $\tilde{O}(n)$ *storage.*

Proof. According to Theorem 5.2.3, a maximal prefix match is an almost optimal answer. Thus, we determine such a match as follows: In a preprocessing step, we build a binary search tree T of height k such that the nodes

Preprocessing on $\mathcal{D} \subset 2^{\mathcal{W}}$
Build a binary search tree corresponding to H.

Processing a query $q \subseteq \mathcal{W}$
Find a maximal prefix match by searching through the tree according to the query's cell sequence.

Figure 5.2: Algorithm for the k-AMI problem in the hierarchical schemes

of T correspond to the cells of H. In every node, we store a list with k entries of the form $(w, \mathcal{L}_w)$, where w is a word located in the cell corresponding to the node, and $\mathcal{L}_w$ is a list of documents containing w. We process a query by determining its cell sequence (i.e., sorting its words according to H) and then searching through T according to this sequence. At each node, we check if the document list $\mathcal{L}_w$ of the current query word w is empty. If it is not empty, we continue the search. Otherwise, or if we reach a leaf, we have found a maximal prefix match. See figure 5.2 for an overview of the algorithm. The preprocessing takes $O(n \log^2 n) \subseteq \tilde{O}(n)$ time if we assume that H is given such that the cell of a word can be determined in $O(\log m) = O(\log n)$ time (for example, if an AVL tree storing the words and their corresponding cells is given additionally to H). The space required for T is $O(n \log n) \subseteq \tilde{O}(n)$. Clearly, processing a query can be done in $O(\log^2 n)$ time. In order to calculate the approximation factor we consider the following quotient:

$$\begin{aligned}
\frac{\lceil s' + 2 \rceil}{\lfloor s - 2 \rfloor} &\leq \frac{k/\log k + 3}{k/(1 + \log k) - 3} \\
&= \frac{(k + 3\log k)(1 + \log k)}{k \log k - 3(1 + \log k)\log k} \\
&< \frac{(k + 3\log k) 2 \log k}{k \log k - 3 \cdot 2 \log k \cdot \log k} \\
&= \frac{k + 3\log k}{\frac{k}{2} - 3\log k} = \frac{2 \cdot \left(1 + \frac{3\log k}{k}\right)}{1 - \frac{6 \log k}{k}} \ .
\end{aligned}$$

Note that $\frac{3\log k}{k}$ and $\frac{6\log k}{k}$ tend to zero as $k = \log n \to \infty$. The probability that the algorithm returns an answer of the above quality is greater than $(1 - 2^{-4k^2}) \cdot (1 - 2/k)$. □

We now discuss the performance of the above algorithm. Its space requirement is linear in m and its query complexity is polylogarithmic in n.

Thus, for inputs following a hierarchical scheme we have a solution to the AMI problem avoiding the curse of dimensionality (cf. Chapter 1).

Chapter 6

Experimental Results

In this chapter we evaluate the Zipf/Mandelbrot model on a real-world data set. We describe how to apply the proposed algorithm and analyze its performance in terms of quality and running time. In addition, we compare it to the randomized algorithm described in Chapter 3. The data set used is a collection consisting of 2659 scientific articles about various topics in computer science. Note that we only evaluate the Zipf/Mandelbrot model and not the hierarchical schemes, since most applications for the maximal intersection problem deal with natural language texts (e.g., text clustering, near-duplicate detection), which follow Zipf's law.

6.1 Experimental Setting

Filtering. Before we perform any preprocessing steps on the collection, we filter the texts. First, we remove special characters (e.g., @, &, ?) and convert all words to lowercase. Second, we remove stop words. As discussed in Section 4.4.2, *stop words* are not relevant for intersection sizes. We use a stop word list consisting of 571 words[1]. This list is part of the SMART (System for the Mechanical Analysis and Retrieval of Text) software, which is an information retrieval system developed at Cornell University [SMA]. Given the nature of our collection, we extend the list by the words "theorem", "lemma", and "proof". These three words occur very frequently in most texts and carry (almost) no meaning with regard to the content of a text.

In [MNF58] Miller et al. established empirically that the average length of a stop word is 3.13 letters. Hence, in addition to the words of the stop word list, we remove all words whose length is less than five.

[1] Words which are naturally contained in this list are articles, prepositions and conjunctions.

Unlike our strict definition of a stop word in Section 4.4.1, most words we filter do not occur in every document from the collection. However, they occur in the majority. Moreover, filtering significantly reduces the document sizes, which in turn reduces the time needed for further processing.

Similarity Measure. Usually, a natural language text is a *multiset* of words. This fact is not consistent with our definition of a document as a set of words. Therefore, we drop word multiplicity and measure similarity between two documents by the number of their common words, that is, the size of their common vocabulary (in the following called *intersection size*). In [CGS03], the following definition of non-exact duplicate documents is given: Two documents are duplicates if they retain much of the same language and if at least 80% of the words of one document are contained in the other. Thus, if we consider only the common vocabulary it is similar to this definition of non-exact duplicates. Moreover, our experimental results give evidence that using the intersection size as defined above as a similarity measure is reasonable, see Table 6.1. Following the discussion in Section 4.4.2, we measure similarity based on the filtered texts (i.e., without stop words in particular).

A small detail in this context is the following. Since on average, a document contains 682 different words after filtering, we use (if available) the 1000 most frequently occurring words of each document for the calculation of the intersection sizes.

Collection Parameters and the Frequency Table. We process 2659 articles from various computer science conferences and journals. The language of all articles is English. The vocabulary size of the collection is 308090. The average document size is 6835 before filtering. It reduces to 2399 after filtering. On average, a document contains 682 *different* words after filtering.

In order to apply the algorithm given in Theorem 4.3.1 (in the following, we call this algorithm the *Zipf algorithm*[2]), we have to count the word frequencies in our collection and to generate a frequency table. Since a natural language text is a multiset of words, there are two different ways of determining the word frequencies: One can count the number of documents in which a word occurs, or one can count the total number of a word's occurrences in each document and then add these values up. We call the former counting method *binary counting* and the latter one *absolute counting*. Figure 6.1 shows for both counting methods the rank-frequency distributions of the computer science collection. The graph shows the rank on the x-axis versus

[2]Actually, we apply the algorithm given in Theorem 4.4.7. Since the only difference to Theorem 4.3.1 is the matrix size, we refer to it as the Zipf algorithm anyway.

Table 6.1: Sample queries and corresponding answers of the Zipf algorithm. Every answer is a maximal match.

Query	Answer
T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, P. White: *Testing Random Variables for Independence and Identity.* 42nd FOCS, 2001	T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, P. White: *Testing that distributions are close.* 41st FOCS, 2000
M. Babaioff, R. Lavi, E. Pavlov: *Single-Value Combinatorial Auctions and Algorithmic Implementation in Undominated Strategies.* JACM, 2009	P. Briest, P. Krysta, B. Vöcking: *Approximation Techniques for Utilitarian Mechanism Design.* 37th STOC, 2005
S. Arora, E. Chlamtac, M. Charikar: *New Approximation Guarantee for Chromatic Number.* 38th STOC, 2006	S. Arora, S. Rao, U. Vazirani: *Expander Flows, Geometric Embeddings and Graph Partitioning.* 36th STOC, 2004
R. Alur, P. Černý, S. Chaudhuri: *Model Checking on Trees with Path Equivalences.* 13th TACAS, 2007	R. Alur, S. Chaudhuri, P. Madhusudan: *A Fixpoint Calculus for Local and Global Program Flows.* 33rd POPL, 2006

the frequency on the y-axis, using logarithmic scales. Both curves shows the rank-frequency distribution *with* stop words.

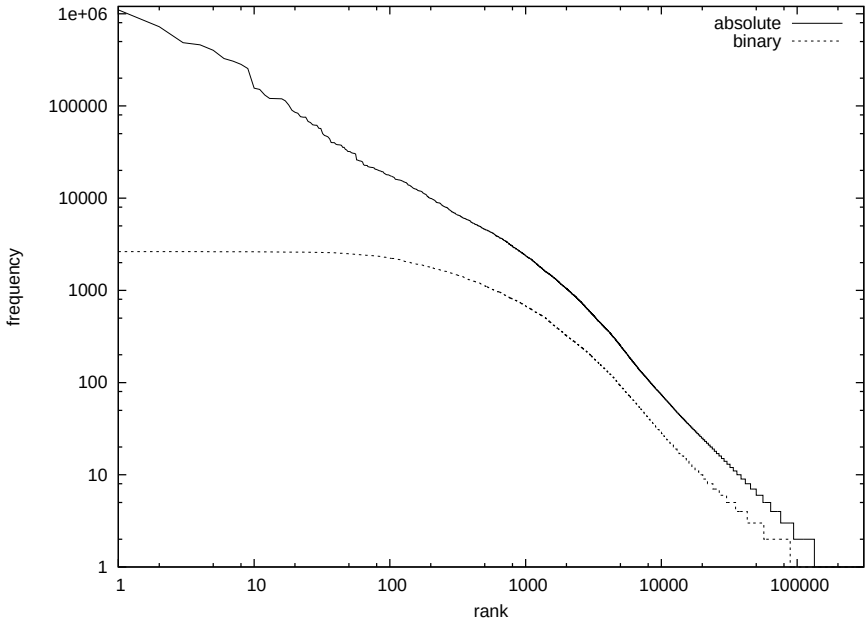


Figure 6.1: Rank-frequency distributions of the computer science collection for binary counting (the dashed line) and absolute counting (the solid line)

Applying binary counting is conform to the Zipf/Mandelbrot model. The corresponding curve (the dashed line) can be approximated by the function $f(x) = 730705/(232 + x)$, which yields $\alpha \approx 274$ ($f(x) = n \cdot \alpha/(\beta + x)$, see Section 4.4.1). The fact that the curve is a horizontal line for the first x-values shows the existence of stop words. Accordingly to the values of α and β, there should be 42 stop words. And indeed, if we examine the first words, we find the first non-stop word at rank 39.

Due to the fact that a document contains on average 682 different words after filtering, we restrict the maximal number of words a document can contribute to the frequency table to the 1000 most frequently occurring ones.

Implementation Details. We implemented the matrix-based and tree-based variant of the Zipf algorithm. A crucial issue are the number of rows and the tree depth. This number is given by Theorem 4.4.5 (our collection contains stop words). However, the statement of this theorem only holds for large values of n, and for our "relative" small collection it would imply a negative number of rows and a negative tree depth. Thus, we set these

values to 1000. As we will see later, the average search depth achieved by the tree-based variant during the search procedure is 122. The index of the last matrix row considered can even be smaller. Therefore, 1000 is sufficiently large.

Implementing the matrix variant is straightforward. The search procedure terminates if the intersection of the document lists has size one, or if it has size zero (in this case, the intersection of the previous step is taken and the answer is determined by calculating the intersection size between the query and each document contained in this intersection).

The tree implementation stores at each node a list of documents corresponding to the path leading to this node. The search procedure traverses the tree via backtracking in depth-first order. The answer is determined by calculating the intersection size between the query and each document from *each* path that matches a maximal number of words with the query. It is necessary to consider each such path in order to obtain the same results as the matrix variant. An important detail in this context is the fact that from such a path we consider all documents corresponding to the node where the last query word is matched.

6.2 Test Results

All tests were performed on a system with 4 AMD Opteron™ processors with 2.6 gigahertz and 24 gigabyte main memory. The algorithm removing stop words and special characters was implemented in Perl, all other algorithms were implemented in C++. In Table 6.2 we list the results of the matrix-based and tree-based variant. The table shows the results for binary counting. All results are averaged over 100 queries, which were randomly chosen from the computer science collection. The parameter *Rank* denotes the answer's position in a document list sorted in decreasing order according to intersection size with the query. In order to lessen the effect of outliers, we also list the median position (the value in brackets). We do the same for the average number of documents which are used for determining the answer (the parameter *Documents*).

The quality the Zipf algorithm achieves is 1.28. It is the quotient of the average of the maximal intersection sizes and the average of the intersection sizes of the answers. In order to compare the quality with the value stated in Theorem 4.4.7, we also calculate it based on the matching size relative to the first (according to the frequency table) query words. Then, the resulting value is 4.5. The theorem states that it should be roughly $\sqrt{\alpha}$, which is $\sqrt{274} \approx 16$. Thus, the actual quality is better by approximately a factor

Table 6.2: Experimental results of the Zipf algorithm on the computer science collection. Data: 2559 computer science articles. The frequency table is based on **binary** counting. All results are averaged over 100 queries.

Variant	**Matrix**	**Tree**
Rank (median)	48 (25)	
Maximum	3	
Intersection size		
answer	271	
only the first words (prefix)	77	
average	152	
maximal	346	
Quality	1.28	
Search depth	-	122
Documents (median)	1 (2)	
Backtracking steps	-	62
Time (in sec:msec)		
Generating the data structure	0:527	1:109
Processing a query	0:341	0:8

of 3.5. Moreover, if we take into account that the statement only holds for sufficiently large n, m, we see that the Zipf algorithm performs significantly better in practice than one would expect.

An interesting point is the fact that, despite the better asymptotic query complexity of the matrix-based variant, the query processing of the tree-based implementation is forty times faster than that of the matrix-based implementation. This may indicate that the document tree has significantly less than n leaves. The fact that on average only 62 backtracking steps are invoked support this assumption.

Table 6.3 shows the results if the frequency table is based on absolute counting. Clearly, the average and maximal intersection size remain the same (the *average intersection size* denotes the average size of the intersection of the query with the database). As expected, the results are slightly worse

Table 6.3: Experimental results of the Zipf algorithm on the computer science collection. Data: 2559 computer science articles. The frequency table is based on **absolute** counting. All results are averaged over 100 queries.

Variant	**Matrix**	**Tree**
Rank (median)	64 (21)	
Maximum	5	
Intersection size		
answer	267	
only the first words (prefix)	56	
average	152	
maximal	346	
Quality	1.3	
Search depth	-	102
Documents (median)	1 (1)	
Backtracking steps	-	96
Time (in sec:msec)		
Generating the data structure	0:530	1:93
Processing a query	0:346	0:8

than those of binary counting since absolute counting fits the model worse. However, a maximum is found five times instead of three times. Again, a query is processed faster by the tree-based implementation (factor 41). The greater number of backtracking steps can be explained by the fact that on average the intersection size with respect to the first query words becomes smaller, while the search depth remains nearly the same. This results in more positions in the document tree where backtracking has to be invoked.

In Table 6.4 we list the results of the randomized algorithm which is presented in Chapter 3. Again, the values are averaged over the same 100 queries that we used before. For sample sets of size 50, the randomized algorithm yields similar results as the Zipf algorithm. The running times are almost the same in this case. Using sample sets of size 200 yields better results, especially with regard to the answer's rank. However, a maximum is

Table 6.4: Experimental results of the randomized algorithm. Data: 2559 computer science articles. All results are averaged over 100 queries.

Sample set	**10**	**50**	**200**
Rank (median)	262 (217)	60 (39)	12 (8)
Maximum	0	2	9
Intersection size	217	252	276
Quality	1.59	1.37	1.25
Time (in msec)	1	8	29

only returned in nine cases and, compared to the tree variant, the running times are slower.

In the next section, we improve the Zipf algorithm such that it finds a maximum more often and, in general, documents having a larger intersection with the query.

6.2.1 Improvements

Based on the good results the randomized algorithm yields for large sample sets, we analyze the performance of the matrix-based variant if the search stops as soon as the intersection of the document lists contains C or less documents. For $C = 200$ the results improve considerably. For binary counting, we achieve an average and a median rank of three, while the quality improves to 1.11. The cases we find a maximum increases to 31. Compared to the standard variant, we find ten times as often a maximum. Compared to the randomized algorithm using a sample sets of size 200, this means more than three times as often a maximum. For absolute counting, the results are even better. The average rank becomes three (median: two). The quality improves to 1.08. Most notable, we find in 49 cases a maximum. Therefore, "picking" documents according to the first words they have in common with the query yields much better results than picking them at random. On average, the search stops for binary counting after intersecting the document lists of the first 25 words, and after 13 for absolute counting. It remains to notice that the query time does not increase observably.

Besides this approach, the most obvious way to improve the algorithm's results may be to allow errors in the prefix match. This is realized by shifting the search range, that is, to start the search not only at the first rank in

the frequency table, but also at later ranks. As answer, we then take the maximum of the different search runs. For our tests, we use three runs and shift the range by 20 ranks each time. Under binary counting, we achieve an average rank of 24 (median: 12) and find a maximum in nine cases. The quality improves to 1.23. The results for absolute counting are almost similar. Interestingly, even this variant of the algorithm has to calculate the intersection between the query and only three documents on average. As expected, the running times triple.

6.2.2 Duplicates

In the following section we examine the performance of the Zipf algorithm in the area of duplicate detection. First, we give an (informal) definition of what constitutes a duplicate (note that there exists no standard definition), and then we explain the different tests we run. The results are shown in Table 6.5.

A *duplicate* (for the sake of clarity, we also refer to an *exact duplicate* when we mean a duplicate) consists of the same text as the original with or without formatting differences like file format, white spaces or paragraph order. A *near duplicate* contains roughly the same semantic content as the original. This informal notion can be captured by the mathematical concept of resemblance introduced in [Bro97].

In the first test, we analyze the algorithm's performance on exact duplicates. We take the same queries as before and include them in the database. Since our algorithm does not depend on any of the formatting issues mentioned above, we expect good results. And indeed, the variant without any improvements finds all duplicates.

In the second test, we use near duplicates. We take 15 conference and corresponding journal articles by Madhu Sudan and use each type as query (clearly, the counterpart of the respective type is contained in the database). It turned out that for articles where the conference and journal version differ significantly in their lengths, the respective counterpart is not found. A reason for this behavior may be that for these queries the length difference results in different vocabularies. In particular, the most frequent words are different. Thus, the query and its counterpart have different index vectors.

For the third test, we generate two query sets with different degrees of modification from the originals. We use almost the same approach as [CFGM02] to generate the sets. For the first one, we modify every fifth word of the original text by choosing a random number smaller than ten. If the number is at least five, we replace the word by a word chosen at random from the frequency table. Otherwise it is deleted. The second set is generated by

modifying every third word in the same manner. On average, the duplicate and the original differ in 24% of their words in the first set, and in 39% in the second one. Note that "real" near duplicates usually do not differ that much from the originals. A drawback of this generation method is the fact that in real near duplicates it could be the case that in the whole text one or more words are consistently replaced by different words. Such a modification possibly has a strong influence on our algorithm. As expected, the results are better if fewer words are altered. The matrix-based variant together with the early abort strategy (answer set ≤ 200) yields the best results. In at least three-fourths of all cases the duplicate is returned. This should be compared with the randomized algorithm using sample sets of size 200. The probability that this algorithm finds a duplicate is approximately 0.075.

In general, we observe that absolute counting yields better results. For finding duplicates, *keywords* (i.e., words describing the topic or content of a text) seem to be important. For example, absolute counting ranks keywords like algorithm, graph or automata higher than binary counting. On the other hand, binary counting ranks words occurring in many documents, like references or introduction, higher. This fact may explain the better results absolute counting achieves at duplicate detection. Here, adapting the stop word list could improve the results.

Finally, we apply the Zipf algorithm to *short* queries. That is, we take title and abstract (without stop words and special characters) from our previous query documents as new queries. On average, these queries consist of 50 different words. Already the standard variant of the Zipf algorithm yields almost all corresponding full texts (98 out of 100), independently of the underlying counting method. An interesting observation for these queries is the fact that the query time of the tree-based variant is 3386 μsec for binary and 2807 μsec for absolute counting, which is notably shorter than the time needed for other types of queries.

6.3 Discussion

Clearly, the implementation of the Zipf algorithm depends on the underlying document collection. Parameters like the number of words each document contributes to the frequency table and the number of matrix rows (respectively, the tree depth) has to be adapted to different collections as well as the parameters of the proposed improvements. These parameters and also the number of matrix rows/the tree depth should be determined by trying different values for a sample set of queries.

Besides these numerical parameters, assembling a "good" stop word list

Table 6.5: Duplicate detection using the Zipf algorithm (without improvements / early matrix abort (≤ 200) / range shift)

	Binary	**Absolute**
Exact	100	100
Conference	6 / 6 / 6	6 / 6 / 7
Journal	2 / 8 / 2	4 / 5 / 6
5th word	41 / 90 / 49	50 / 85 / 70
3rd word	30 / 78 / 37	43 / 73 / 54
Abstract	98 / 99 /98	98 / 99 / 98

also depends on the collection. For example, we extended the SMART list by the words "theorem", "lemma", and "proof", which should in other collections not be treated as stop words. In general, choosing the right stop words seems to be a sensitive issue.

The good results our algorithm achieves for short queries indicate that it could be applied in applications like web search engines or electronic archives, since queries occurring there are similar to the short queries (title, abstract) we used.

The fact that we found all exact duplicates opens another angle for applying our method. The idea is to use a document's index vector as its *fingerprint*. If these fingerprints have a constant size (in our tests 1000), checking if a document collection of size n with vocabulary of size m contains duplicates can be done in time $O(nm \log m)$ by generating the fingerprints and inserting them into a binary search tree (the time for inserting vectors of a constant length into a tree is linear in the number of vectors). Note that a naive approach that compares each document with every other document takes $O(n^2)$ time. Thus, for natural language texts our approach is faster since $m \in O(n^\gamma)$, with $0 < \gamma < 1$ (see Heaps' law, Section 4.3).

Chapter 7

Conclusion and Open Questions

In this thesis we have studied the maximal intersection (MI) problem. We provided cell probe lower bounds corresponding to the lower bounds for NNS in the Hamming cube via reducing this problem to the MI problem. These bounds hold if the vocabulary size is asymptotically smaller than every polynomial function which operates on the database size. We also argued that we cannot expect that a so called preferable solution to the MI problem exists, since such a solution would also imply a preferable solution to NNS, which so far is not known and believed to be nonexistent among researchers.

We presented a randomized algorithm that yields an approximate solution to the MI problem. The algorithm takes a random sample from the database and determines an optimal solution subject to this sample. Its running time only depends on the vocabulary size, and not on the database size.

Our main results are two new randomized input models along with an efficient (quasi-linear or polylogarithmic query time) deterministic algorithm for each model solving the approximate version of the MI problem. The first model, called the Zipf model, is based on the fact that in natural language texts words follow Zipf's law, an empirical law stating that the frequency of any word is inversely proportional to its rank in the frequency table. The second model, called the hierarchical schemes, is based on the assumption that the elements from the vocabulary can be arranged in a binary tree-like table. We proved that in both models a "threshold phenomenon" on the probabilities of intersecting sets holds, which led to the above mentioned algorithms.

Finally, we evaluated the Zipf algorithm on a real document collection consisting of natural language texts. We explained how real documents have to be processed and discussed implementation details. The experimental results we obtained were significantly better than we would have expected from our theoretical analysis. By modifying the algorithm we were able to

obtain even better results. It turned out that our methods are particularly effective in the area of duplicate detection.

We conclude this thesis by pointing out open questions:

- Could the quality of the Zipf algorithm be improved by adapting the stop word list during execution? Could the algorithm learn which words should be treated as stop words?
- Are there stronger lower bounds for the MI problem, especially without the restriction on the vocabulary size?
- Our algorithms yield an answer which meets a guaranteed approximation factor only with high probability. Does there exist an efficient algorithm yielding a guaranteed approximation every time?
- Are there other reasonable input models or assumptions about the input leading to efficient solutions?
- Most important: Does there exist an *efficient* algorithm solving the exact version of the MI problem (i.e., faster than a linear scan over the database)?

Bibliography

[AM92] Pankaj K. Agarwal and Jiří Matoušek. Ray shooting and parametric search. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 517–526, New York, NY, USA, 1992. ACM.

[AMN+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

[BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[BO97] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Tucson (United States)*, pages 357–368. ACM, 1997.

[BOR99] Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing, Atlanta (United States)*, pages 312–321. ACM, 1999.

[BR02] Omer Barkol and Yuval Rabani. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *J. Comput. Syst. Sci.*, 64(4):873–896, 2002. Conference version appeared at STOC'00, ACM Press, 388–396 (2000).

[Bro97] A. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.

[BSMS95] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART: TREC 4. In *Proceedings of the Fourth Text Retrieval Conference*, 1995.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[CFGM02] Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[CGS03] Jack G. Conrad, Xi S. Guo, and Cindy P. Schriber. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management, New Orleans (United States)*, pages 443–452. ACM, 2003.

[CH67] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[Cla88] Kenneth L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17(4):830–847, 1988.

[Cla94] Kenneth L. Clarkson. An algorithm for approximate closest-point queries. In *SCG '94: Proceedings of the tenth annual symposium on Computational geometry*, pages 160–164, New York, NY, USA, 1994. ACM.

[cY81] Andrew Chi chih Yao. Should tables be sorted. *J. Assoc. Comput. Mach*, 28(3):615–628, 1981.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc., 1973.

[DL76] David P. Dobkin and Richard J. Lipton. Multidimensional searching problems. *SIAM J. Comput.*, 5(2):181–186, 1976.

[DW82] L. Devroye and T. J. Wagner. Nearest neighbor methods in discrimination. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2. North-Holland, 1982.

[Ede87] Herbert Edelsbrunner. *Algorithms in combinatorial geometry.* Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[Est16] J. B. Estoup. *Gammes Sténographiques.* Paris, 4th edition, 1916.

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, 1977.

[Fre78] Michael L. Fredman. Observations on the complexity of generating quasi-gray codes. *SIAM J. Comput*, 7(2):134–146, 1978.

[GKP02] R. Graham, D. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, 2002.

[GW99] Ian P. Gent and Toby Walsh. The search for satisfaction. Technical report, Department of Computer Science, University of Strathclyde, Scotland, 1999.

[GW00] Bernd Gärtner and Emo Welzl. A simple sampling lemma: Analysis and applications in geometric optimization. *Discr. Comput. Geometry*, 25:569–590, 2000.

[Hea78] H. S. Heaps. *Information retrieval: Computational and theoretical aspects.* Academic Press, 1978.

[HLLN09] Benjamin Hoffmann, Mikhail Lifshits, Yury Lifshits, and Dirk Nowotka. Maximal intersection queries in randomized input models. *Theor. Comp. Sys.*, 46(1):104–119, 2009.

[HLN07] Benjamin Hoffmann, Yury Lifshits, and Dirk Nowotka. Maximal intersection queries in randomized graph models. In Volker Diekert, Mikhail V. Volkov, and Andrei Voronkov, editors, *CSR*, volume 4649 of *Lecture Notes in Computer Science*, pages 227–236. Springer, 2007.

[HT96] Trevor Hastie and Robert Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(6):607–616, 1996.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM.

[Kle97] Jon M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 599–608, New York, NY, USA, 1997. ACM.

[Knu97] Donald E. Knuth. *The Art of Computer Programming 1: Fundamental Algorithms*. Addison-Wesley, 3rd edition, 1997.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 614–623, New York, NY, USA, 1998. ACM.

[KWZ95] R. M. Karp, O. Waarts, and G. Zweig. The bit vector intersection problem. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 621, Washington, DC, USA, 1995. IEEE Computer Society.

[Loè77] Michael Loève. *Probability Theory I*. Springer, 4th edition, 1977.

[Man65] B. Mandelbrot. *Information theory and psycholinguistics*. Basic Books, 1965. Reprinted as: B. Mandelbrot: R. C. Oldfield and J. C. Marshall (eds.), Language, Penguin Books (1968).

[Mei93] S. Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2):286–303, 1993.

[Mil99] Peter Bro Miltersen. Cell probe complexity - a survey. In *In 19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 1999. Advances in Data Structures Workshop, Chennai, (India)*, 1999.

[MNF58] George A. Miller, E. B. Newman, and Elizabeth A. Friedman. Length-frequency statistics for written english. *Information and Control*, 1(4):370–389, 1958.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MS99] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[PT06] Mihai Patrascu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. In *FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, Berkley (United States)*, pages 646–654. IEEE Computer Society, 2006.

[Reu08] Thomson Reuters 2008 Annual Report, 2008. http://ar.thomsonreuters.com/. Cited 29 October 2009.

[Sal88] Gerald Salton. *Automatic Text Processing*. Addison-Wesley, 1988.

[Sam84] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984.

[SMA] SMART ftp site. ftp://ftp.cs.cornell.edu/pub/smart/. Cited 29 October 2009.

[SS01] Thomas Schickinger and Angelika Steger. *Diskrete Strukturen 2: Wahrscheinlichkeitstheorie und Statistik*. Springer, 2001.

[Tsa99] Panayiotis Tsaparas. Nearest neighbor search in multidimensional spaces. Technical report, Department of Computer Science, University of Toronto, Canada, 1999.

[Wal69] Rolf Wallisser. Zur Transzendenz der Werte der Exponentialfunktion. *Monatshefte für Mathematik*, 73:449–460, 1969.

[WSB98] Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[Zip49] G. K. Zipf. *Human behavior and the principle of least effort.* Addison-Wesley, 1949.

Index