Statistical Properties of Query Strings

P. Oscar Boykin, Jesse S.A. Bridgewater, Vwani P. Roychowdhury

January 28, 2004

Abstract

We study the statistical properties of query search strings in the Gnutella peer-to-peer network and find that the distribution of query frequency (how often queries are made) closely follows a lognormal distribution. We find evidence for an upper bound on query cache performance and extend previous studies of Gnutella query caching to model effectiveness of such schemes.

1 Introduction

The Gnutella file sharing system is an open architecture allowing peers to connect to one another and search for files, and download them from one another. Every node in the network is a so-called servant, having the properties of a server and a client. Each servant performs broadcast search queries and receives query hits in response to queries. Queries have a finite time-to-live (TTL). Servants rebroadcast the queries , decrementing the TTL each time until the $TTL = 0$ at which time the query is not rebroadcast.

Since the fundamental operation of Gnutella, which is a database search, is a broadcast operation, it is of great interest to develop techniques that reduce the need to make such broadcasts. By studying the statistics of query frequency, we are able to make predictions about the efficacy of simple caching. For each query that comes to a node, if that node has previously cached the result of that query, there is no need to broadcast it on, which results in a bandwidth savings for the node in question. In this work, we want to study the probability that a query is found in the cache as a function of the size of the cache. Ideally, the probability of finding a query in the cache will grow rapidly as a function of size.

1.1 Previous work

Much work has focused on the distribution of web requests in both proxy-cache architectures and search engines. From the early web caching studies in the mid 1990s it has been observed that the frequency-rank relationship of file requests to web servers is a Zipf-like power-law[6][2]. Markatos points out that caching in P2P networks has significant differences from web caching[9], but he also argues that even simple web-inspired cache strategies can improve performance. The efficiency of his caching schemes were estimated using a Gnutella data set. In a separate study of web caching based on search engine queries to the EXCITE web search engine, he demonstrates that the rank-frequency plot follows a power-law very convincingly[8].

K. Sripanidkulchai attempts to apply the same method to the study of Gnutella queries[11]. However unlike in the search engine study, significant deviations from the power-law model exist in the Gnutella query data if one tries to use the same rank-frequency plot method that was successful in analyzing web searches.

Both of the above works gave empirical results that implied that caching of query hits might drastically reduce the need for widely broadcasting queries. Those results are based purely on their recorded datasets. By contrast, we develop a model for query frequency, and then do analysis to study the efficacy of query caching. Our approach gives some possible insight into how asymptotic behavior might look as the query rate grows.

1.2 Summary of Results

While previous results have been empirical examinations of query caching schemes, our approach is to develop a model of query frequency distribution and use this model to predict the scaling properties of a simple query cache. Often query frequency plots are presented as rank-frequency plot, such as figure 1 for our Gnutella data. However, the rank-frequency curve is not of direct interest to caching schemes. Instead we examine the distribution of query frequency: if we make t query observations, what is the probability that a query will be observed k out of the t times? Queries which are of no use to our cache, will occur only one time, and hence, no caching scheme can make use of them. So, a question of fundamental importance is what is the probability that a random query string will occur only one time, which we refer to as p_1. In Gnutella, we will see that over the entire range of data, p_1 is always observed to be greater than 1/2.

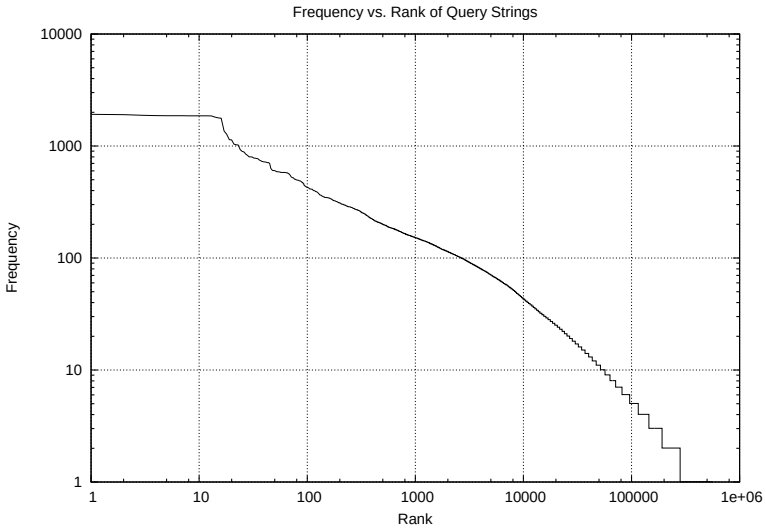

Figure 1: Rank-Frequency plot for our Gnutella Query Data Set

A simple caching strategy remembers the previous N distinct queries, and inserts the next non-cached query at the head of the queue, possibly pushing a query off the queue's tail if the cache is full. This is identical to Most Recently Used (MRU) caching except that MRU caching reorders the cache when non-distinct queries arrive so that the newest query is always at the head of the queue. We say the probability that the next query is *not* in the cache, as a function of N, is q_N. How q_N scales will depend on the distribution of query frequency. We offer a model for how q_N scales for the observed queries in Gnutella. Breslau et. al.[3] discuss some of the common cache replacement algorithms employed for web caching.

Finally, we obtain results which show that for Gnutella, q_N approximately follows a power-law, which limits the efficiency of query caching: some benefit is obtained by a relatively small cache, however making cache efficiency much larger requires great increases in cache size, as we see in table 1

1.3 Gnutella Data Gathering

The decentralized architecture makes it quite easy for any node to collect a large dataset by simply recording all of the query hits that pass through that servant. We performed this procedure on a modified version of gtk-gnutella[7] over a period of 12 hours on January 13, 2003. During this period, we recorded 3,240,649 queries,[1] 578,764 of which were distinct query strings. This indicates that the data set collected approximately 75 queries per second.

[1]This number excludes cryptographic hash queries which search for exact files and are not keyword searches

Cache size	Hit rate
256	21.6%
2480	26.3%
7280	36.1%
19,800	50.4%
49,200	66.3%
115,000	77.7%
259,000	85.7%

Table 1: Summary of expected cache hit rates for various size caches in Gnutella

2 Query Frequency Modeling

In this section we present results on various statistics about query strings in the Gnutella file sharing system. Each query is represented by a string. If two queries are represented by the same string, we say these queries are identical, otherwise we say these two queries are distinct. A trace of queries from a system will have t total queries with N distinct queries, and $t - N$ repeated queries, each of which are identical to one of the N distinct queries.

When we compute, over the entire data set, the average number of times a distinct query string was seen, we find that the average number of appearances is 5.17. Put another way, with probability almost exactly half, a distinct query string will only be seen once, however, given that it is seen 2 or more times, the average is 9.3.

As discussed above, researchers in this area have been concerned with the probability of each query. This is typically presented as a rank-frequency plot. The highest rank query is the query which appears the most times. The noted linguist Zipf pointed out that when words from a text are plotted in this way, the result follows a power-law of approximately $F \propto R^{-1}$. This simple result does not however appear to be a very good model for Gnutella query strings as seen in previous studies[11] as well as our own analysis. An alternate approach pursued in this study is to look at the frequency distribution, or how many queries have each frequency. We find using this approach that the frequency distribution exhibits excellent agreement with the lognormal distribution.

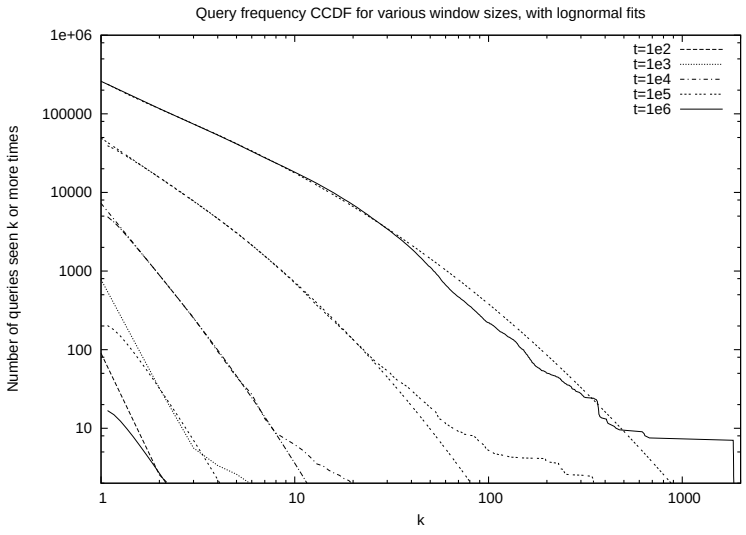

Figure 2: CCDF of frequency of Gnutella query strings

Figure 2 shows that this very closely matches the lognormal distribution. In particular, figure 2 shows that for this data set, the probability that query occurs k or more times follows[2]

$$CCDF(k) \quad = \quad (1 - \Phi(\frac{\ln(\frac{k-\mu}{m})}{\sigma}))$$

with parameters $\mu = 1.0$, and m and σ given in table 2 This distribution is called lognormal. To get the probability of seeing a query k times, $p_k = CCDF(k) - CCDF(k+1)$. As k gets a bit larger, we may approximate the difference as a derivative to obtain:

$$p_k \quad \approx \quad \frac{e^{-\frac{\ln^2((k-\mu)/m)}{2\sigma^2}}}{(k-\mu)\sigma\sqrt{2\pi}}$$

The mean of this distribution is

$$\langle k \rangle = me^{\sigma^2/2} + \mu \ . \tag{1}$$

While we fit the data to a lognormal to get two parameters m, σ for each number of distinct queries seen, N, we do not have any model for a rule for how m, σ might scale with N. The total number of queries t is clearly related to the average number of times a query is observed: $t = \langle k \rangle N$.

We note that in figure 2, there are usually "anomalous" high frequency queries, i.e. we consistently see slightly more queries in the tail than we would expect according to the model. By examining these strings in particular, we believe that these queries are the result of a individuals or unusual clients which are sending queries on a massively repeated basis. The top few highest frequency queries in the data were closely related to one another, and seemed to be precise file names for related content. On the other hand, the queries which ranked just behind these, seemed to be more general search keywords, and the sorts of queries that are popularly seen in search engines[3]. Hence, we believe that the very highest frequency terms operate with a mechanism fundamentally different of the normal query formation mechanism. A similar effect was observed in rank-frequency studies of Gnutella queries[11].

Lognormal distributions have been observed in a wide range of computer systems and there has been recent interest in modeling the formation of lognormal behavior in engineered systems. In particular many types of network traffic [4] and file size distributions [10, 5] follow lognormal distributions. In the case of file size distributions, there are at least two different proposed models of file generation that match the observed distributions[10, 5]. It would be of interest to formulate models of query formation which agree with the observed frequency distribution. Such models should give an idea for how we can expect m and σ to scale.

Although no generating model is currently available that yields the lognormal query frequency CCDF, there is a connection between the distributions of the word counts in Gnutella query strings and books titles. In figure 3 we see that the distribution on number of words in a query follows closely $p_l \sim e^{-0.27l}$.

Is the length distribution related to anything underlying the Gnutella system? We suggest that the queries follow closely the structure of titles in general. Project Gutenberg [1] is an effort to bring books that have no copyright or an expired copyright into digital form encoded as ASCII text. By studying the distributions on the number of words in the titles that they make available (5861 in the title list we use), we see that it is very similar to Gnutella query length distribution. Figure 4 shows that the number of words in the titles of the books in Project Gutenberg follows closely $p_l \sim e^{-0.22l}$.

Note that the maximum length in both cases is very close. In the case of Gnutella, the maximum was 46. In the case of Project Gutenberg, the maximum is 51. Additionally, figure 3 shows that queries of high length are less frequent than the exponential law would predict. Of course, the Gnutella data appears to be a much better fit to the law, but the data set is also much larger. Recall that the number of queries in the data set is 2.99×10^6 compared to 5861 in the case of Project Gutenberg.

[2] $\Phi(x) = \int_{-\infty}^{x} \frac{e^{-t^2}}{\sqrt{2\pi}} dt$

[3] Top recording artists names, "mp3", and various sexually oriented queries

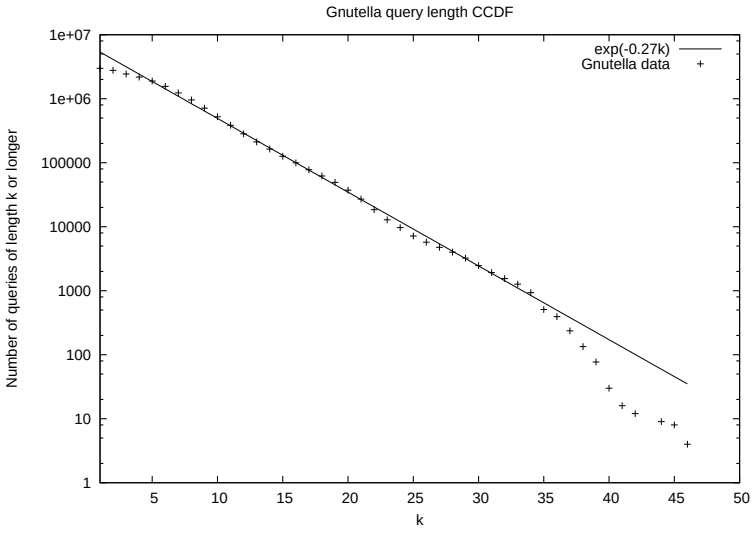

Figure 3: CCDF of number of words in Gnutella queries

3 Caching in time sensitive systems

The problems we have looked at in the previous two sections are related to caching in a model that is relevant in p2p systems, namely the case where a result has a short lifespan, and that the caching system will be large enough to hold all results with have not yet expired. In this case, there is no need for any cache replacement strategy as you can store all previous results which are still valid.

A natural question in this setting, is how effective will caching be? In fact, if the query rate is such that you can only store the last N elements before they are no longer "fresh", then the probability that a query is not in the cache is the probability of seeing a unique query q_N (see section 5.1).

Relating this problem to section 5.2, for a cache that has access to the last t queries, we only have to cache N queries, and we hit the cache $t - N$ times. So, a system which is efficient for caching has an N_t which grows very slowly, or equivalently, q_N which goes to zero very quickly.

The last column of table 4 gives the size N we need in order to hit the cache with probability q. We would like to see that cache size does not increase dramatically as $q \to 0$. Exponential distributions scale logarithmically in $1/q$. Power-law distributions scale polynomially in $1/q$. So, uniform and exponential distributions allow for very efficient caching: if you want q to be exponentially small, then N needs to scale as a constant size or linearly in the uniform and exponential cases respectively. For the power-law case, an exponentially small q requires an exponentially large N.

3.1 Caching of Gnutella query hits

Figure 6, we see q_N for Gnutella. As we see by the close inverse polynomial approximation, and the arguments in the end section 5.2, caching in Gnutella is close to caching in systems with power-law distributions on queries. Despite the fact that q_N does not go to zero as fast as we may hope, practically speaking it allows for caching which can significantly reduce the communications requirements of the Gnutella system. Table 1 summarizes expected cache hit rates for various cache sizes.

Our results show that even very small caches produce high hit rates, a finding that matches the conclusions of previous authors [11, 9]. Modest sized query caches ($N = 5,000 - 50,000$) produce hit rates of roughly 50 percent, while using a much larger cache will raise the hit rate to around 90 percent. However because of the high degree of time localization of queries, a higher hit rate does not translate to better system performance. This is obvious because as cached queries age, the probability of obtaining erroneous cache hits is likely to increase significantly [9]. To be precise, an erroneous cache hit would return a query hit that refers to

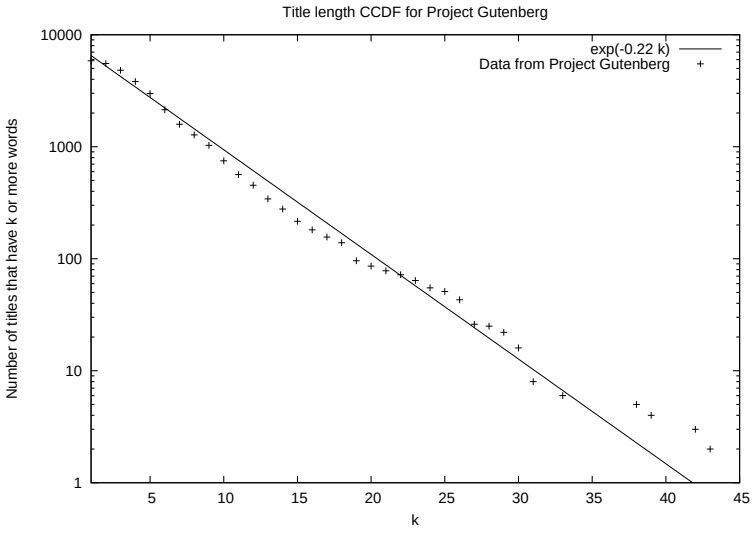

Figure 4: CCDF of number of words in Project Gutenberg titles

content that no longer exists at the node referred to in the query hit, or to content associated with a node that is no longer connected to the network. A good general approach is to constrain how many hits, N, are cached as well as how long a hit may remain in the cache. The expiration of cached query hit validity is closely related to the lifetimes of content-bearing nodes in the network, so statistical measures of node lifetimes may alone provide adequate guidance for cache validity time limits. Additionally empirical study of how long query hits remain valid in file sharing systems might shed light on how long a cache can hold results before assuming them to be invalid.

4 Probability that a distinct query occurs exactly once

When considering effectiveness of caching it is important to know the probability that randomly selected query will be in a set of cached queries. In particular, if we remember the last N distinct queries, we might ask what fraction of those N were only seen 1 time. Looking to the Gnutella data, we plot this fraction in figure 5.

According to the lognormal model, if we choose a query string uniformly from the set of all distinct query strings, the probability that it appears only once is

$$p_1 = CCDF(1) - CCDF(2) = \Phi(\frac{\ln(\frac{1}{m})}{\sigma}). \tag{2}$$

In figure 5 we assume $\sigma = aN^b + c$, and that $m = 0.6$, and we find parameters a, b, c which come closest to the observed data. The fit for σ does not match the results from figure 2. This is because the CCDF fit fits over all frequencies, not just for the frequency 1. Figure 5 has a lognormal where $\sigma = 2.8 \times 10^{-4} N^{0.82} + 0.68$.

Note that the fit parameters in table 2 has $m < 1$, so we immediately see that since $\Phi(x) \geq 1/2$ for all $x \geq 0$, that $p_1 \geq 1/2$. We do not know the scaling of m and σ with N, however six orders of magnitude, we don't see one with $m > 1$. If there is some reason that $m < 1$ for all N, it means that more than half the distinct queries will be seen only once. In any case, as long as $\log(1/m)/\sigma$ is close to zero, p_1 is close to $1/2$.

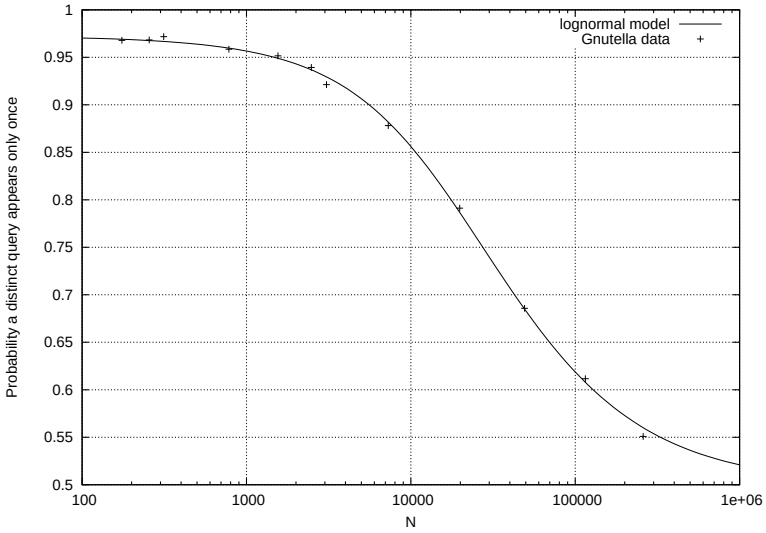

Figure 5: The fraction of distinct queries which are only seen 1 time: p_1

t	N	m	σ	p_1 data	p_1 model
10^3	782	0.42	1.21	0.97	0.84
10^4	7.3×10^3	0.40	1.38	0.88	0.83
10^5	4.9×10^4	0.69	1.73	0.69	0.62
10^6	2.6×10^5	0.81	2.28	0.55	0.55

Table 2: Fit parameters for the lognormals in figure 2, along with probabilities of seeing a query one time

5 Analytic models

In this section we use analytic techniques to compare various probability distributions for queries to what we see with Gnutella.

5.1 Probability of seeing unique queries

We assume that each query, Q_i, has an independent probability p_i, and that we order the queries from most likely to least likely, such that $p_i \geq p_{i+1}$. After we have seen N unique queries, the probability of seeing a unique query we denote as q_N. The quantity $1 - q_N$ may be calculated by computing the probability of all the different ways that one may see N distinct queries. If we see that the first N queries, $Q_1, Q_2, \ldots, Q_N$ are also the most likely queries by assumption, then we see that:

$$\begin{aligned} 1 - q_N &\leq \sum_{i=1}^{N} p_i \\ q_N &\geq \sum_{i=N+1}^{\infty} p_i \\ &\geq \int_{N+1}^{\infty} p_i di \end{aligned}$$

We use the fact that p_i is a decreasing function to bound the sum with an integral. In cases where the sum is difficult, but the integral is easy, this will be useful.

p_i	q_N
$1/M$	$1 - \frac{N}{M}$
$(e^{1/\mu} - 1)e^{-i/\mu}$	$e^{-N/\mu}$
$\frac{1}{\zeta(\gamma) i^\gamma}, \gamma > 1$	$\frac{1}{(N+1)^{\gamma-1}}$

Table 3: Summary of probability of seeing a unique query for various distributions

5.2 Rate of distinct queries

As we monitor the past t queries, how many *distinct* queries (N) have we seen? In other words, given access to t queries, N are distinct, and $t - N$ are repeated from the N distinct queries. What follows is an analysis of three different probability distributions and the rate unique query growth[4]

A difference equation for the expectation value of N is the following:

$$E(N_{t+1}) = E(N_t) + q_{N_t} \tag{3}$$

Dropping the expectation value, and approximating the difference as a differential we obtain $\frac{dN}{dt} = q_N$, which gives the solution, $\int_0^n \frac{dN}{q_N} = t$.

q_N	N_t	$N(q)$
$1 - \frac{N}{M}$	$M(1 - e^{-t/M})$	$M(1 - q)$
$e^{-N/\mu}$	$\mu \ln(1 + t/\mu)$	$\mu \ln \frac{1}{q}$
$\frac{1}{(N+1)^{\gamma-1}}$	$(\gamma t + 1)^{1/\gamma} - 1$	$(\frac{1}{q})^{\frac{1}{\gamma-1}} - 1$

Table 4: Summary of rate of distinct queries, N_t, and the required cache size, $N(q)$, for query hit probability q for various distributions

Approximate solutions of this problem for uniform, exponential, and power-law distributions are given in table 4.

5.3 Fraction of Unique Queries

Using our lognormal model for query frequency introduced in section 2, we can predict the query hit rate for a simple query caching scheme. If our cache remembers the last N distinct queries, which came from a stream of t total queries, we can ask what is the probability that the next query will be among the N in our cache. In order to answer the previous question we first answer the following: when we select a query string uniformly from a set of t observed query strings, what is the probability that the query is seen only once. Note, that this is not the same question that we asked in section 5.1 where we restrict the set of queries to the N *distinct* queries. In the set of t observed queries, distinct queries which appear more often are more likely to be selected. The probability of selection a query which appears only once in the t observed

[4]In fact, this is related to the so-called birthday problem: assuming that every birthday is equally likely, how many people does one need before you are likely to have a pair that has the same birthday. This occurs when $t - N = 1$, or that there is one repeated birthday.

queries we denote as g_t. We can calculate g_t from the frequency distribution, it is the ratio of the number of observed queries which appear once, over the total number of observed queries:

$$g_t = \frac{p_1}{\sum_k k p_k} = \frac{p_1}{\langle k \rangle}$$

According to the model:

$$g_t = \frac{\Phi(\frac{\log(1/m_t)}{\sigma_t})}{m_t e^{\sigma_t^2/2} + 1}$$

Instead of considering total observed queries t, and look at the number of distinct queries N, one finds: $g_N = \sum_t g_t P(t|N)$. In figure 6 we see g_N from the data along with a fit to the function $1/(1.20+5.21\times10^{-4}N^{0.748})$. The fit in figure 6 is not directly based on our lognormal models, hence a model that can account for m and σ would be of great value.

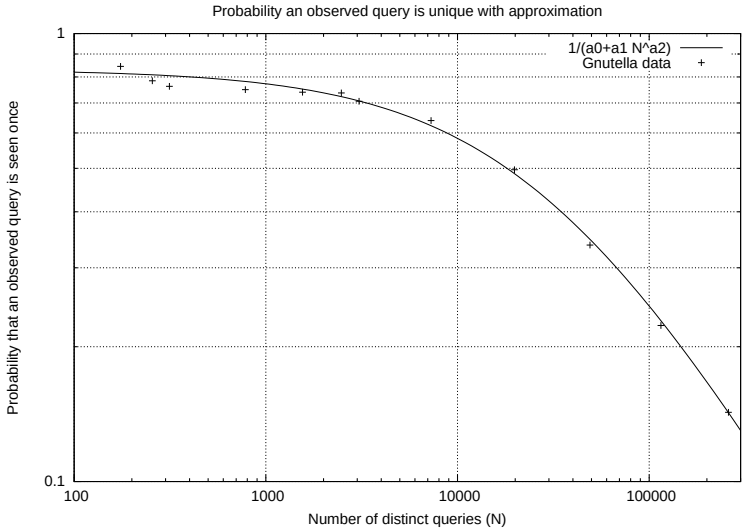

Figure 6: The fraction of observed queries which are only seen 1 time: g_N

5.4 Relating Query Hit Rate to Fraction of Unique Queries

To compute the hit rate of a the simple caching scheme, g_t, introduced in section 5.3, is not directly applicable. The query hit rate is itself a probability: given that we have $t-1$ total observed queries, what is the probability that the next one will be a *new* query. We will call this query hit rate q_t.

Intuitively, it may seem that g_t and q_t are related. In fact, since g_t is an estimate of the probability of a query being unique over t queries, and q_t is what is the probability that the t^{th} query will be unique, in so far as the the previous history is a good indicator for the future, it would seem that g_t and q_t are equal. In the following we see that indeed g_t and q_t are equal in many cases, including the case where the queries are drawn independently.

First we introduce some new notation. We define a string of t queries $Q_0, Q_1, \ldots, Q_{t-1}$ as a vector $\vec{Q_t}$. We introduce two vector functions, Y and X which count *unique* queries: $X(\vec{Q_t})$ is the number of unique queries in the vector $\vec{Q_t}$, or how many of the distinct queries appear only one time; $Y(\vec{Q_t})$ is 1 if Q_0 is unique, and 0 otherwise. A shift operator on the query vector is represented as:

$$S_i(Q_0, Q_1, \ldots, Q_{t-1}) = (Q_{i \pmod t}, Q_{1+i \pmod t}, \ldots, Q_{i+t-1 \pmod t})$$

Clearly, $X(\vec{Q}_t) = \sum_{i=0}^{t-1} Y(S_i(\vec{Q}_t))$ and $X(\vec{Q}_t) = X(S_i(\vec{Q}_t))$. The probability of drawing a particular string of t queries is $P(\vec{Q}_t)$. Now we may write g_t and q_t in these new terms:

$$\begin{aligned} q_t &= \sum_{\vec{Q}_t} P(\vec{Q}_t) Y(\vec{Q}_t) \\ g_t &= \sum_{\vec{Q}_t} P(\vec{Q}_t) \frac{X(\vec{Q}_t)}{t} \end{aligned}$$

where the sum is over all possible sets of t queries. Note that we do not restrict the set of all possible queries to a finite set. Assume that:

$$P(\vec{Q}_t) = P(S_i(\vec{Q}_t))$$

This is of course true if the queries are drawn independently, in which case, the order of the queries does not matter. Applying the above:

$$\begin{aligned} |q_t - g_t| &= |\sum_{\vec{Q}_t} P(\vec{Q}_t) \left(Y(\vec{Q}_t) - \frac{X(\vec{Q}_t)}{t} \right)| \\ &= |\frac{1}{t} \sum_i \sum_{\vec{Q}_t} P(\vec{Q}_t) \left(Y(\vec{Q}_t) - \frac{X(\vec{Q}_t)}{t} \right)| \\ &= |\frac{1}{t} \sum_i \sum_{\vec{Q}_t} P(S_i(\vec{Q}_t)) \left(Y(S_i(\vec{Q}_t)) - \frac{X(S_i(\vec{Q}_t))}{t} \right)| \\ &= |\frac{1}{t} \sum_i \sum_{\vec{Q}_t} P(\vec{Q}_t) \left(Y(S_i(\vec{Q}_t)) - \frac{X(\vec{Q}_t)}{t} \right)| \\ &= |\sum_{\vec{Q}_t} P(\vec{Q}_t) \left((\frac{1}{t} \sum_i Y(S_i(\vec{Q}_t))) - \frac{X(\vec{Q}_t)}{t} \right)| \\ &= |\sum_{\vec{Q}_t} P(\vec{Q}_t) \left((\frac{X(\vec{Q}_t)}{t}) - \frac{X(\vec{Q}_t)}{t} \right)| \\ &= 0 \end{aligned}$$

5.5 Comparing Model and Data

In this section we assume that for Gnutella, $q_N = g_N$ and we see how N should depend on t in that case. We find excellent agreement between the model and the data, indicating that for Gnutella, q_N and g_N are indeed very close. In figure 7 we see the data from Gnutella along with an analytic solution based on a polynomial fit of q_N from figure 5, which gives: $1.20N + \frac{5.21 \times 10^{-4}}{1.748} N^{1.748} = t$.

From the lognormal model presented in section 2 we see that: $q_N = \frac{p_1}{t/N}$. We can solve the differential equation exactly in a some cases. In the case where p_1 is constant, we have:

$$\begin{aligned} \frac{dN}{dt} &= \frac{p_1}{t/N} \\ \frac{dN}{N} &= \frac{p_1 dt}{t} \\ \ln(N/N_0) &= p_1 \ln(t/t_0) \\ N &= N_0 (t/t_0)^{p_1} \end{aligned}$$

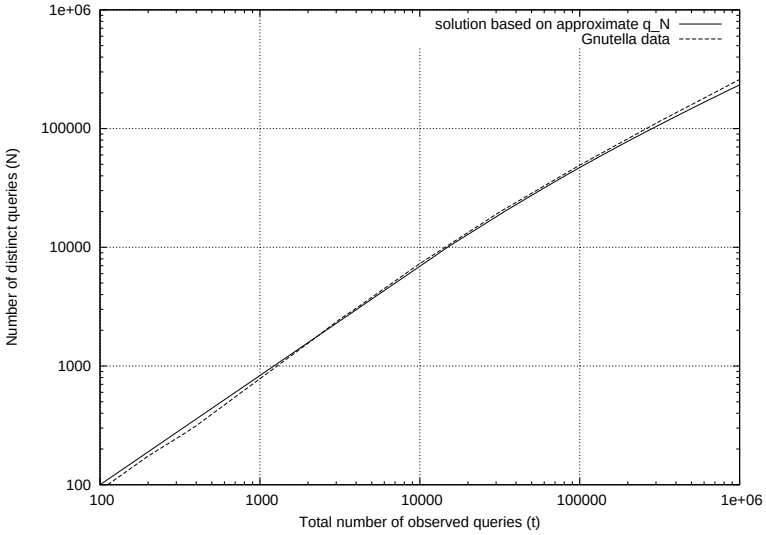

Figure 7: N_t: distinct queries vs. total queries

It is interesting to note, that if the lognormal model scales well, and if $\ln(1/m) << \sigma$ for large N, as in figure 5, then we can expect p_1 to go to 1/2. This would imply that in the limit $N \propto \sqrt{t}$ and $q_N \propto 1/N$.

We know that p_1 for Gnutella is not constant, but regions of figure 5 are approximately flat on a linear-log plot. We can also solve the differential equation in the case that $p_1 = \alpha + \beta \ln t$:

$$\begin{aligned} \frac{dN}{dt} &= \frac{p_1}{t/N} \\ \frac{dN}{N} &= \frac{\alpha + \beta \ln t \; dt}{t} \\ \ln(N/N_0) &= \alpha \ln(t/t_0) + \frac{\beta}{2}(\ln^2 t - \ln^2 t_0) \end{aligned}$$

Which gives quadratic solutions for $\ln N$ in $\ln t$. Since p_1 is a decreasing function of N and hence t, we would expect β to be negative giving only negative curvature for any plot of $\ln N$ vs $\ln t$ and while figure 7 does not have a large curvature, there is some negative curvature apparent which is in agreement with this analytic solution.

6 Conclusion

This study provide further evidence for modest caching proposals in p2p networks such as Gnutella. We suggest an alternative approach to previous caching studies in p2p networks by examining the distribution of frequencies rather that the rank-frequency relationship. In taking this approach we observe excellent agreement with a lognormal distribution. So far a generating model for the lognormal behavior is not obvious, however the similar length distributions of the number of words in Gnutella query strings and book titles suggests a direction for future study of a generating model.

References

[1] Project gutenberg. http://promo.net/pg/.

[2] Virgílio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. Characterizing reference locality in the WWW. In *Proceedings of the IEEE Conference on Parallel and Distributed Information Systems (PDIS)*, Miami Beach, FL, 1996.

[3] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM (1)*, pages 126–134, 1999.

[4] Allen B. Downey. Evidence for long-tailed distributions in the internet. http://citeseer.nj.nec.com/downey01evidence.html, 2001.

[5] Allen B. Downey. The structural cause of file size distributions. In *SIGMETRICS/Performance*, pages 328–329, 2001.

[6] Steven Glassman. A caching relay for the World Wide Web. *Computer Networks and ISDN Systems*, 27(2):165–173, 1994.

[7] Y. Grossel and R. Manfredi. Gtk-gnutella. http://gtk-gnutella.sourceforge.net/.

[8] Evangelos P. Markatos. On caching search engine query results. *Computer Communications*, 24(2):137–143, 2001.

[9] Evangelos P. Markatos. Tracing a large-scale peer to peer system: an hour in the life of gnutella. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[10] M. Mitzenmacher. Dynamic models for file sizes and double pareto distributions. http://citeseer.nj.nec.com/mitzenmacher01dynamic.html, 2002.

[11] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. http://www.cs.cmu.edu/ kunwadee/research/p2p/gnutella.html, February 2001.