Efficient Peer-to-Peer Semantic Overlay Networks based on Statistical Language Models

Alessandro Linari *
University of Bologna
Viale Risorgimento, 2 - 40136
Bologna, Italy
alinari@deis.unibo.it

Gerhard Weikum
Max-Planck Institute for Informatics
Stuhlsatzenhausweg 85, 66123
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

In this paper we address the query routing problem in peer-to-peer (P2P) information retrieval. Our system builds up on the idea of a Semantic Overlay Network (SON), in which each peer becomes neighbor of a small number of peers, chosen among those that are most similar to it. Peers in the network are represented by a statistical Language Model derived from their local data collections but, instead of using the non-metric Kullback-Leibler divergence to compute the similarity between them, we use a symmetrized and "metricized" related measure, the square root of the Jensen-Shannon divergence, which let us map the problem to a metric search problem. The search strategy exploits the triangular inequality to efficiently prune the search space and relies on a priority queue to visit the most promising peers first. To keep communications costs low and to perform an efficient comparison between Language Models, we devise a compression technique that builds on Bloom-filters and histograms and we provide error bounds for the approximation and a cost analysis for the algorithms used to build and maintain the SON.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Design

Keywords

Semantic Overlay Network, Peer-to-Peer, Language Models, Metric Space, Nearest Neighbor Search

*On leave at MPII.

P2PIR'06, November 11, 2006, Arlington, Virginia, USA.

1. INTRODUCTION

1.1 Motivation

Peer-to-peer (P2P) networks potentially offer major advantages for large-scale decentralized search of information such as file sharing or Web search [33]. We envision an architecture where every peer has a powerful local search engine, with its own crawler, indexer, and query processor. Such a peer can compile its own content from thematically focused crawls and other sources, and make this content available in a P2P overlay network. Search requests issued by a peer can first be executed locally, on the peer's locally indexed content, with a strong potential for personalization and other advanced IR techniques. In cases when the recall of the local search result is unsatisfactory, the query may be forwarded to a small set of other peers that are expected to provide thematically relevant, high-quality and previously unseen results. Deciding on this set of target peers is the *query routing* problem in a P2P search network (aka. collection selection).

A practically viable query routing strategy needs to consider the similarity of peers in terms of their thematic profiles, the overlap of their contents, the potential relevance of a target peer for the given query, and also the costs of network communication. Many proposals have been made in the literature, for example: globally available term statistics about the peers' contents [8, 16, 27, 2], epidemic routing using gossiping strategies [18], routing indices with peer summaries from local neighborhoods [10, 21], statistical synopses such as Bloom filters or hash sketches maintained in a directory based on distributed hash tables (DHT) [5, 28, 32], randomized expander graphs with low-diameter guarantees [25, 26] and randomized rendezvous [31], clustering of thematically related peers [11, 12, 23], superpeer-based hierarchical networks [24, 22], cost/benefit optimization based on coarse-grained global knowledge [29, 30], and many more.

Typically, query routing decisions would be made at query run-time, when the query is issued at some peer. But many of the above methods involve directory lookups, statistical computations, and multi-hop messages; so it is desirable to precompute some basic elements of query routing decisions and amortize this information over many queries. A technique for doing this is to encode a similarity-based precomputed binary relation among peers into a so-called *semantic overlay network (SON)* [10, 1]. Each peer P becomes directly connected to a small number of peers that are likely to be good routing targets for many of P's queries. At query

run-time, the query router would consider only the SON neighbors of the query initiator and select a subset of these based on a more detailed analysis of similarity, overlap, networking costs, etc.

From a given peer's viewpoint, one particularly intriguing choice of SON neighbors are those peers with the lowest *Kullback-Leibler (KL) divergence* [9] regarding the term-frequency distributions in the peers' locally indexed contents. This information-theoretic measure captures the general thematic similarity of two peers in a natural manner and is well-founded in the recent work on statistical language models for IR [20]. In a language model (LM), queries and documents are viewed as samples generated from an underlying probability distribution over terms. When we assume that the contents of a peer represents this distribution and assume that queries are generated according to a multinomial model, the best peer for a query is the one that has the highest likelihood of generating the query; this is mathematically equivalent, in terms of peer ranking, to the lowest KL divergence between the query LM and the peer LM. Finally, if we do not have a specific query at hand at the time we want to precompute the similarity of two peers, we consider the contents of the query-initiating peer as a substitute for the query itself and replace the query LM by that peer's contents LM.

LM's have become state-of-the-art practice for many IR tasks, and they have already been proposed also for P2P IR by [24]. In conventional IR settings, with one LM for each document and one LM for each query, parameter estimation and smoothing are very critical parts of LM-based approaches; details of smoothing techniques are often decisive for search result quality [35]. In contrast, a P2P setting provides a much more convenient and robust environment for LM's, as the peers have rich and naturally available sources for parameter estimation and smoothing. Instead of a document LM we can use the entire contents of a peer as the basis of the LM or as a background corpus for smoothing, and instead of a query LM we can leverage the user-behavior history (query logs, click streams, bookmarks) for robust parameter estimation. In this sense, LM's and P2P networks are a "marriage made in heaven".

Unfortunately, however, an LM-based approach to P2P IR also entails major computational costs, and it is unclear how to make this approach practically viable in a large-scale environment.[1] More specifically, we face the following efficiency problems:

1. Computing the exact KL divergence between two peers' term-frequency distributions incurs non-negligible overhead as it ranges over high-dimensional feature spaces. It would be desirable to disregard a large part of "insignificant" features, but it is not obvious how to efficiently approximate the KL divergence this way and control the approximation error.

2. The computations involve shipping term-frequency vectors over the network. With high-dimensional feature spaces, these messages have non-negligible size and may incur significant consumption of network bandwidth.

3. Most severely, as the KL divergence is not a metric (i.e., the triangle inequality does not hold), there is no obvious way of transitively inferring, from knowing the distances for some pairs of peers, transitive distances so as to eliminate peers that are "too far away" from a given peer. Thus, we would need to compute the KL divergence for *all pairs of peers*, and this quadratic cost is out of the question for a scalable approach. Being able to prune the search space of all peer pairs is fundamentally important and poses a great challenge.

In this paper we provide an efficient solution to overcoming these problems and making LM-based P2P IR truly scalable and practically viable.

1.2 Contribution

We address the above problems by combining insights and methods from different areas and integrating them into a novel kind of SON maintenance strategy. First, we utilize recent results from information theory [13], which show that the square root of the *Jensen-Shannon (JS) divergence* is a metric. The JS divergence is a symmetrized and smoothed relative of the KL divergence. We adopt the JS divergence instead of the KL divergence as a metric distance measure because it allows us to effectively prune the search space of peer pairs. Second, we build on existing methods for metric similarity search in centralized settings and adapt them for our P2P setting. Peers meet randomly and exchange information about their nearest neighbors in the metric space. Peers remember the distances to interesting peers, and with the JS-divergence-based metric, they can effectively use the triangle inequality for early elimination of uninteresting peers. This technique is key for scalability. Third, we compress the term-frequency vectors and speed up the computation of two peers' JS divergence by using appropriately designed compact synopses based on Bloom filters [6]. Our technique comes with guarantees about the approximation error.

To the best of our knowledge, this is the first method for LM-based P2P query routing that can claim to be truly scalable to millions of peers and resilient to high dynamics (i.e., frequent peer failures and high churn). We overcome the efficiency problems outlined in Subsection 1.1 and provide a scalable solution with the following salient properties:

1. Fast comparisons of per-peer LM's by synopsis-based approximations with error bounds,

2. Low communication cost by LM compression and search-space pruning using the JS-based metric,

3. Judicious message routing for SON construction and maintenance.

The rest of the paper is organized as follows. Section 2 presents our architectural model and introduces notation. Section 3 discusses our techniques for approximating LM's using compact synopses. Section 4 presents our algorithms for SON construction and maintenance. Section 5 analyzes the networking costs of our method. We conclude with an outlook on ongoing and future work.

[1] This is probably the reason why prior work has considered LM-based approaches only for superpeer-based P2P systems, but these approaches do not scale up to completely decentralized networks with millions of equally standing peers and no hierarchical structure at all.

2. SYSTEM ARCHITECTURE

2.1 Semantic Overlay Network

Our system consists of a physical layer (an example is the TCP/IP infrastructure) connecting a variable number of autonomous peers, which cooperate in order to build and maintain a virtual network of nodes that are directly linked only if their local data collections are "semantically similar". Ideally, this similarity should be (reciprocal to) a *metric distance*, computed by a function $d : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}_0^+$ which satisfies the triangular inequality and where $\mathbb{X}$ is the set of peers. Thus we would wish to view the P2P network as a *metric space* and construct the SON from the metric distances between peers. The advantage of such a model would be that we can exploit the triangle inequality to prune the search space of all peer-pairs when we construct the edges of the SON. Once we know that for peers x, y, z the distances are upper-bounded by $d(x, y) \leq a$ and $d(y, z) \leq b$, we can infer that $d(x, z) \leq a + b$. With sufficiently small values a and b we could then consider peer z as a candidate to be linked to x in the SON. Conversely, if we do not have any indication of this kind and rather suspect that z is semantically far away from x based on other observed distances, we could prune z from our search for good neighbors of peer x even without computing $d(x, z)$.

With the above consideration, an intriguing strategy is to run a process of random (but possibly biased) meetings of peer-pairs. When two peers "meet", we compute or approximate their semantic distance and keep this pair as a candidate for a SON edge if we estimate that the distance is small enough. We iterate these meetings and use the triangle inequality and other techniques for distance estimation. In the end, the peer-pairs with the lowest (estimated) distances form the edges of the SON. As peers may leave the P2P network anytime, new peers join, and the document contents of a peer evolves over time, we actually run peer meetings continuously in the background in order to maintain the SON in the presence of this dynamics. We present more details of this SON construction later in Section 4.

2.2 Metric Distance for Language Models

A typical "distance" which measures the dissimilarity between any two documents is the Kullback-Leibler divergence [9] between their respective Language Models denoted as Θ and Ψ:

$$D_{KL}(\Theta \parallel \Psi) = \sum_{i=1}^{|\mathcal{T}|} \theta_i log \frac{\theta_i}{\psi_i} \quad (1)$$

where θ_i and ψ_i are the frequencies associated to term t_i in the two Language Models. Unfortunately, D_{KL} is not symmetric and does not satisfy the triangular inequality. A symmetrized version is the Jensen-Shannon divergence [9],

$$D_{JS}(\Theta, \Psi) = \frac{1}{2} \cdot [D_{KL}(\Theta \,\|\, \Phi) + D_{KL}(\Psi \,\|\, \Phi)] \quad (2)$$

where $\Phi = \frac{1}{2}(\Theta + \Psi)$. Adopting an information-theoretic approach, if we have a random variable X whose observations can be drawn with equal probability from any of two distributions Θ and Ψ, then the minimum codelength needed to represent X can be found computing the "inefficiency distance" of both distributions from Φ and by averaging over the result. Recent advances in the field of information theory [13] have shown that the measure $\sqrt{D_{JS}}$ is a metric.

Table 1: Summary of relevant notation

Symbol	Description
$\mathcal{T}$	Global term space
$\mathcal{T}_\Theta \subset \mathcal{T}$	Term space of peer P_Θ
Θ, Ψ	LM's associated with peers P_Θ,P_Ψ
$\widetilde{\Theta}$, $\widetilde{\Psi}$	Approximate LM's of peers P_Θ,P_Ψ
θ_i, ψ_i	Frequency of the i-th term in P_Θ, P_Ψ
$\widetilde{\theta}_i$, $\widetilde{\psi}_i$	Approx. frequency of the i-th term
$\theta_H^{(i)}$	Frequency associated with the l-th BF
r_Θ, r_Ψ	Search radii for peers P_Θ, P_Ψ
$d = d(\Theta, \Psi)$	Distance between P_Θ and P_Ψ
$\widetilde{d} = d(\widetilde{\Theta}, \widetilde{\Psi})$	Approx. distance between P_Θ and P_Ψ

Thus, the metric distance function $d : \Theta \times \Psi \rightarrow \mathbb{R}_0^+$ between any two Language Models Θ and Ψ is defined as:

$$d(\Theta, \Psi) = \sqrt{\sum_{t_i \in \mathcal{T}_\Theta \cup \mathcal{T}_\Psi} d_{JS}(\theta_i, \psi_i)} \quad (3)$$

and

$$d_{JS}(\theta_i, \psi_i) = \frac{1}{2} \left[\theta_i \cdot \log \frac{2 \cdot \theta_i}{\theta_i + \psi_i} + \psi_i \cdot \log \frac{2 \cdot \psi_i}{\theta_i + \psi_i} \right]. \quad (4)$$

In Table 2.2 we summarize the notation that will be used in the rest of the paper. In particular, Θ and Ψ denote the Language Models of peers P_Θ and P_Ψ; $\widetilde{\Theta}$ and $\widetilde{\Psi}$ are their approximate versions (introduced in Section 3) and r_Θ and r_Ψ denote the search radii for the two peers (used by the meeting algorithms in Section 4).

3. SYNOPSIS-BASED APPROXIMATIONS OF LANGUAGE MODELS

When two peers meet they first exchange their LM's and then compute their similarity by applying Equation 3: unfortunately, this operation leads to high communication and computational costs because it performs complex mathematical operations over a huge set of frequencies. In this section, we show how a LM can be compressed by using a set of I Bloom-filters and how this representation can be exploited to efficiently compute the distance between the peers, allowing only a small and controllable error on the final value.

At the end of the section, we will give practical guidelines to the design of the system, in terms of correctly choosing the number and the size of the Bloom-filters used to represent the Language Model.

3.1 A Solution based on Bloom-Filters

Bloom filters have received a lot of attention since their introduction in [6], because of their ability of giving probabilistic error guarantees when representing a set of elements drawn from any given domain. A Bloom-filter is a vector of m bits initially set to 0, which is used to represent a set of n elements and to subsequently test their membership to the set. Each element is drawn from a finite universe U and is hashed by h hash functions which independently set one of the m bits. To test if an element is in the Bloom-filter, the h hash functions are applied to it: if each bit addressed by the hash functions is 1, then the searched element "should"

be present, being the false positive probability given by

$$fpp \approx \left[1 - e^{-\frac{hn}{m}}\right]^h. \quad (5)$$

To represent a LM in a compact manner, first note that when comparing two LM's by the square root of the JS divergence, the most important contributors to a large distance are the terms for which the two LM's have *widely differing frequencies*. So these terms should be captured more precisely than the terms for which the two LM's show low differences in the term frequencies. Our approach conceptually compresses an LM in two steps. First, we construct a histogram of possible term frequencies with a small number of equi-width histogram cells. Conceptually, each histogram cell is associated with the subset of terms whose frequencies fall into the cell's boundaries. As a second step, we then compress these subsets by mapping them onto a Bloom-filter (BF). Thus, we arrive at one BF per histogram cell of each LM.

A histogram for P_Θ which describes the distribution of terms in Θ is built over the domain of frequencies by partitioning it in I disjoint intervals of equal-size, such that the i-th interval corresponds to the range $\left[\theta_H^{(i)} - \frac{\delta}{2}, \theta_H^{(i)} + \frac{\delta}{2}\right]$. Each interval is represented by a Bloom-filter which contains all terms whose frequency belongs to it: the approximate Language Model $\widetilde{\Theta}$ is represented by the set $\{BF(\theta_H^{(i)})\}_{i=1,\ldots,I}$, where $BF(\theta_H^{(i)})$ is the i-th Bloom-filter. To extract from a given set of BF's the frequencies which characterize the corresponding $\widetilde{\Theta}$, the following rules apply:

1. For each term $t_i \in \widetilde{\Theta}$, its frequency $\widetilde{\theta_i}$ is $\theta_H^{(i)}$ if $t \in BF_\Theta(\theta_H^{(i)})$.

2. When a term belongs to two or more BF's at the same time, we suspect a false positive. This case should happen with very low probability, but it may happen because of the lossy compression characteristics of Bloom-filters. We could then make a heuristic choice such as choosing the average value of the term frequencies, or we may directly ask the original peer for the correct frequency. In any case, the metric properties of the data space are preserved if, for each peer, only one of the these policies is consistently applied.

Finally, we can define the approximate distance $\widetilde{d}$ between two peers P_Θ and P_Ψ as

$$\tilde{d}(\Theta, \Psi) = d(\widetilde{\Theta}, \widetilde{\Psi}). \quad (6)$$

by observing that, if there exists only one possible approximation for each LM, then $\widetilde{d}$ will still be a metric distance over the space of all peers.

In the following, we study the behavior of Equation 3, given the approximation introduced by the BF-based histograms. Note that representing a frequency distribution with a histogram is a well-known technique in the database community and has already been extensively studied, however the problem we are facing now is quite different because we do not need to estimate the performance of this technique *per se*, rather we want to characterize the approximation introduced in the computation of the distance function.

We start our analysis by introducing $\varepsilon_{\Theta,max}$, defined as the maximum distance between a LM and its approximation, i.e., as $\max\{d(\Theta, \widetilde{\Theta})\}$ for any Θ. By applying the triangular inequality to any two peers P_Θ and P_Ψ whose exact distance is d, we can write that:

$$\max\{0, \widetilde{d} - 2 \cdot \varepsilon_{\Theta,max}\} \leq d \leq \widetilde{d} + 2 \cdot \varepsilon_{\Theta,max} \quad (7)$$

where $\widetilde{d}$ is the the approximate distance $\widetilde{d}(P_\Theta, P_\Psi)$. From Equation 7, then, we compute an upper-bound on the absolute error ε_{abs}, defined as the the difference between the approximate and exact distance between any pair of peers:

$$\varepsilon_{abs} = |d(\Theta, \Psi) - d(\widetilde{\Theta}, \widetilde{\Psi})| \leq 2 \cdot \varepsilon_{\Theta,max}. \quad (8)$$

Before fully characterizing $\varepsilon_{\Theta,max}$, we consider the distance $d(\Theta, \widetilde{\Theta})$ as it is computed by Equation 3 and move the square-root inside the sum; then, we find that:

$$d(\Theta, \widetilde{\Theta}) \leq \sum_{t_i \in \mathcal{T}} \sqrt{d_{JS}(\theta_i, \widetilde{\theta_i})}. \quad (9)$$

Note that each term in Equation 9 is a single-term LM: for any pair of frequencies θ_i and $\widetilde{\theta_i}$, then, if the difference $|\theta_i - \widetilde{\theta_i}|$ is maximized, their JS-divergence is maximized as well. Since each approximate frequency $\widetilde{\theta_i}$ is drawn from the set $\{\theta_H^{(1)}, \ldots, \theta_H^{(I)}\}$, we conclude that the natural choice for θ_i is among $\theta_H^{(i)} - \frac{\delta}{2}$ and $\theta_H^{(i)} + \frac{\delta}{2}$. If we define ε_{lat} as the maximum distance between any θ_i and its approximation $\widetilde{\theta_i}$, i.e.,

$$\varepsilon_{lat} = \max_d \left\{ d = \sqrt{d_{JS}\left(\theta_H^{(i)}, \theta_H^{(i)} \pm \frac{\delta}{2}\right)},\ \forall i \right\}, \quad (10)$$

we obtain that $\varepsilon_{\Theta,max} = N \cdot \varepsilon_{lat}$, where $N = |\mathcal{T}|$ is the number of terms in Θ. We can further substitute this result in Equation 8 and eventually find the expression of the absolute error

$$\varepsilon_{abs} \leq 2 \cdot N \cdot \varepsilon_{lat} \quad (11)$$

which, as we would have expected, is proportional to the approximation introduced by the BF-based histogram.

3.2 An Efficient Technique for Distance Computation

A distance function based on the JS-divergence is computationally very expensive, because it consists of a sum of several terms whose mathematical expression is not trivial.

Consider two Language Models $\widetilde{\Theta}$ and $\widetilde{\Psi}$, represented by a set of I Bloom-filters as described in Section 3.1, and suppose you want to compute the distance $d(\widetilde{\Theta}, \widetilde{\Psi})$ between them. The computation can be highly optimized if we have a data structure M, that we represent as a matrix of dimension $[I \times I]$, which stores, at location (i, j), the result of the computation of the JS-divergence (Equation 2) between any two frequencies drawn from the set $\{\theta_H^{(1)}, \ldots, \theta_H^{(I)}\}$. The algorithm works as follows:

1. For each term $t_i \in \mathcal{T}$ search the two sets of Bloom-filters in parallel to obtain the two approximate frequencies $\widetilde{\theta_i}$ and $\widetilde{\psi_i}$.

2. When there is indication of a false positive (i.e., a term appearing in two BF's, see above), we heuristically choose a value or we directly ask the peer for the exact value.

3. In repeated computations (i.e., in a series of peer-pair meetings), we use the precomputed value at location $M[\widetilde{\theta}_i, \widetilde{\psi}_i]$ or, if not present, we compute it explicitly and insert it into M.

4. We maintain an incremental sum over all terms and return the square-root of the final value.

This technique has the major advantage that M is common to all meetings and that its size is proportional to $O(I^2)$ which, considering the small number of BF's needed to represent a Language Model, leads to a negligible space consumption. The approximate distance is computed very efficiently and has the desirable property to be incremental (this will be fully exploited in Section 4 to allow peers to perform an early pruning during their meetings). Suppose, for example, that P_Θ has received from P_Ψ only $I' < I$ BF's: the lower-bound is then obtained by applying Equation 2 only to the terms contained in the available BF's. By separately applying the square-root to the result, the computation does not need to start again when a new BF is received.

3.3 Choosing Bloom-filter Sizes

A key element for our architecture is to choose the correct number I and the size m of the Bloom-filters used to represent a peer. From Equation 11, we observe that each term contributes to the absolute error with a value proportional to ε_{lat}: given Equation 10, we need to compute the single-term JS-divergence between the generic θ_H and the lower- and upper-limits of its interval (we should consider the farthest one). If we denote with a and b the two frequencies such that $a < b$ holds, i.e. we indicate that $(a,b) = \left(\theta_H - \frac{\delta}{2}, \theta_H\right)$ or $(a,b) = \left(\theta_H, \theta_H + \frac{\delta}{2}\right)$, we characterize their JS-divergence in terms of Equation 4 as:

$$d_{JS}(a,b) = a \cdot \log \frac{2a}{a+b} + b \cdot \log \frac{2b}{a+b} \leq (b-a) \log \frac{b}{a} \quad (12)$$

The proof of this inequality is straightforward, by observing that the first term is always negative; the difference $b - a$ equals to $\frac{\delta}{2}$ in both cases and the logarithm is always positive and increases with the two values getting closer to 0. The expression of ε_{lat} is, thus, the following:

$$\varepsilon_{lat} \leq \sqrt{\frac{\delta}{2} \cdot \log \frac{\theta_H^{(1)}}{\theta_H^{(1)} - \frac{\delta}{2}}}, \quad \theta_H^{(1)} \leq \theta_H^{(i)} \ \ \forall i \quad (13)$$

with $\theta_H^{(1)}$ being the smallest frequency used to build a BF with. This result has a major impact on the tuning of the system. In particular, it suggests that variable-width histograms would be preferable as different frequency ranges contribute differently to the overall approximation. We make the following observations that may lead to further optimizations: the first BF might contain only terms with a frequency close to 0. Moreover, it produces the largest error. BF's representing higher frequencies should have wider histogram intervals, since they cause smaller absolute errors.

Consequently, we should use variable-width histogram boundaries for the BF approximation, with carefully tuned cell widths. Here we can apply results from selectivity estimation and data synopses in database systems [15, 19]. The analysis of the LM approximation error can be extended along these lines, but this is beyond the scope of the current paper.

The last step of our tuning procedure consists in choosing the size of the Bloom-filters: consider, for a moment, the false positive probability given by Equation 5. If we use an optimal number of hash functions, given by $h = m/n \cdot \ln 2$ [7], Equation 5 can be approximated by $fpp = \left(\frac{1}{2}\right)^{\frac{m}{n}}$, which gives us the possibility to tune the size of our BF's in order to achieve the desired value of fpp.

Unfortunately, this is not always the case: in particular, when the number n of elements inserted in the BF grows too much, the compression technique based on BF's might become not convenient anymore. Suppose, for example, that n gets close to $|\mathcal{T}|$ (which is the number of elements in the universe): to have a false positive probability $fpp \approx 1/2$ (which is indeed a very high value), we need at least $m \approx |\mathcal{T}|$ bits in the BF. On the other hand, if we use a bit vector of size $|\mathcal{T}|$, we can represent the whole universe without any error, as long as we assume to have a function which univocally maps the i-th object on the i-th bit of the vector.

4. ALGORITHMS FOR SON CONSTRUCTION AND MAINTENANCE

In this section we present the algorithms needed to build the semantic overlay as outlined in Section 2. Each peer aims to find its k *nearest neighbors* in the network, in terms of the introduced metric distance. The nearest neighbor problem in a metric space has been widely studied for centralized environments [17, 34], where the space regions relevant for the query are distinguished from the non relevant ones by exploiting information of an index data structure built on the data collection; since in our scenario we cannot rely on such an index structure, each peer needs to perform a series of random meetings with other peers in order to find its neighbors and build its own view of the network.

Whenever a meeting (see Algorithm 4.1) occurs, the two peers exchange information in order to compute a lower-bound on their similarity distance, so that they can decide whether they should become SON neighbors or not. The *search radius* of a peer is defined as the distance from the farthest peer in the current list of nearest neighbors, and is used to prune the search space of those nodes whose distance (or lower-bound) is too high. To increase the probability of success of future meetings, after a peer P_Θ has completed a meeting, it computes a lower-bound between the met peer P_Ψ and all peers currently in its neighbors list and "suggests" these values to P_Ψ (see Algorithm 4.2). Algorithm 4.3 shows the strategy by which peers are chosen to initiate a meeting with. Note that, in order to minimize the chance that two peers meet too frequently, each peer maintains a list of peers already met, along with their distances or lower-bounds.

The random choice of meeting partners can be facilitated by the underlying network, using, for example, a distributed hash table or epidemic message spreading. The random choice may be biased, based on different criteria such as network distance (e.g., measured in round-trip latency). In our actual algorithm, we use a priority queue which stores peers (and their lower-bounds) suggested by others: the highest priority is given to the peer with the lowest lower-bound, in order to shrink the initiating peer's search radius as quickly as possible [17].

For additional flexibility and robustness to network dynamics (i.e., peer failures, churn, and evolving contents of

peers), each peer periodically initiates new meetings, even if it has reached its "optimal state".

4.1 The Meeting Algorithm

Algorithm 4.1 *meeting*(P_Θ,P_Ψ)

Require: P_Θ P_Ψ, r_Θ and r_Ψ

1: $i \leftarrow 0$ and $lb_\Theta \leftarrow 0$
2: **while** $(lb_\Theta < r_\Theta \wedge i < I)$ **do**
3: P_Θ: $receive(P_\Psi, BF_\Psi^{(i)})$ and $lb_\Theta.update(\cdot)$
4: $i \leftarrow i + 1$
5: **if** $(i = I)$ **then** {All BF's were received}
6: P_Θ: $\widetilde{d} \leftarrow lb_\Theta$
7: **if** $(\widetilde{d} < r_\Theta)$ $neighbors(P_\Theta).Insert([P_\Psi, \widetilde{d}])$
8: P_Θ: $send(P_\Psi, \widetilde{d})$
9: **if** $(\widetilde{d} < r_\Psi)$ **then**
10: P_Ψ: receive P_Θ's Bloom-filters
11: P_Ψ: $neighbors(P_\Psi).Insert([P_\Theta, \widetilde{d}])$
12: **else** {P_Ψ is pruned on lower-bound}
13: P_Θ: $send(P_\Psi, lb_\Theta)$
14: **if** $(lb_\Theta \geq r_\Psi)$ **then** Stop
15: $i \leftarrow 0$ and $lb_\Psi \leftarrow 0$
16: **while** $(lb_\Psi < r_\Psi \wedge i < I)$ **do**
17: P_Ψ: $receive(P_\Theta, BF_\Theta^{(i)})$ and $lb_\Psi.update(\cdot)$
18: $i \leftarrow i + 1$
19: **if** $(lb_\Psi \geq r_\Psi)$ **then** Stop
20: **else** P_Ψ: $\widetilde{d} \leftarrow lb_\Psi$
21: P_Ψ: $neighbors(P_\Psi).Insert([P_\Theta, \widetilde{d}])$

Algorithm 4.1 shows the procedure by which two peers P_Θ and P_Ψ exchange information about their LM's, compute a lower-bound and, if necessary, the distance $\widetilde{d}$. The search radii of P_Θ and P_Ψ are denoted as r_Θ and r_Ψ, respectively, and lb_Θ and lb_Ψ are the lower-bounds computed at each peer. As explained in Section 3.2, the lower-bound in lines 1-4 (and 15-18) is computed by using a partial representation of the two Language Models, i.e., only the subset of the I BF's that is currently available. Thus, as a new BF arrives from the remote peer, the value of the lower-bound increases (line 3 for P_Θ and 17 for P_Ψ), and when no BF's are left, eventually turns into the distance $\widetilde{d}$ (i.e., the second condition at lines 2 and 16 fails). This technique minimizes the number of BF's sent when an unsuccessful meeting occurs. Unfortunately, if P_Θ is not interested in the meeting anymore, P_Ψ needs to restart the computation (line 15), since, in general, the i-th BF in different peers contains different terms. Note that if P_Θ has already computed the distance $\widetilde{d}$, P_Ψ needs P_Θ's BF's only to make it one of its SON neighbors (lines 9-11).

Algorithm 4.2 shows the procedure by which any two peers P_Θ and P_Ψ exchange information about their knowledge of the network topology. We present the algorithm from P_Θ's viewpoint denoting with lb_Θ the lower-bound it computed on $\widetilde{d}(\Theta, \Psi)$ and with $\widetilde{d}_1, \ldots, \widetilde{d}_k$ the distances between P_Θ and its k nearest neighbors $P_1, \ldots, P_k$. Another input to the algorithm is the list PQ_{met}, containing those peers that P_Θ has met during previous meetings. This list, filled at the end of Algorithm 4.1, was not shown to simplify the notation.

By applying the triangular inequality to P_Θ, P_i and P_Ψ,

Algorithm 4.2 *gossip*(P_Θ,P_Ψ)

Require: lb_Θ, $\widetilde{d}_1, \ldots, \widetilde{d}_k$, PQ_{met}

1: V_Θ: vector containing up to k elements
2: **for** $i = 1, \ldots, k$ **do**
3: P_i: i-th neighbor of P_Θ
4: $V_\Theta[i] = \langle P_i, |lb_\Theta - \widetilde{d}_i| \rangle$
5: P_Θ: $send(P_\Psi, V_\Theta)$
6: $V_\Psi \leftarrow receive(P_\Psi, V_\Psi)$
7: **for all** $(\langle P, lb \rangle \in V_\Psi)$ **do**
8: **if**$(P \notin PQ_{met})$ **then** $PQ.Enque(P)$ with priority lb

it is possible to compute a lower-bound on the distance $\widetilde{d}(P_\Psi, P_i)$ without the need of any additional computation (line 4). P_Θ, then, builds the vector V_Θ with the pairs $\langle P_i, lb \rangle$ and sends it to P_Ψ (line 5). When receiving P_Ψ's vector, P_Θ extracts all P_Ψ's neighbors and inserts them in its candidate list PQ, with priority given by the lower-bound computed by P_Ψ (lines 6-8).

Algorithm 4.3 *choosePeer*()

Require: PQ, α
1: $\Pr_{rm} \leftarrow \Pr\{random\ meeting\ occurs\}$
2: **if** $(PQ \neq \oslash$ **and** $\Pr_{rm} > \alpha)$ **then** $P \leftarrow PQ.Deque()$
3: **else** $P \leftarrow chooseRandomPeer()$
4: $meeting(P_\Theta, P)$
5: $gossip(P_\Theta, P)$
6: **if** $r_\Theta > \widetilde{d}_k$ **then** $r_\Theta \leftarrow \widetilde{d}_k$
7: **if** $(PQ.First().lb > r_\Theta)$ **then** $PQ.Clear()$

4.2 Algorithm for Network Maintenance

Algorithm 4.3 shows the procedure adopted by peer P_Θ to choose the next peer to meet. When a peer joins the network, its search radius is set to the maximum distance, so that the first k meetings are always successful. Note that, in general, the peer to meet is chosen as the top one in the priority queue PQ, i.e., the one with the smallest lower-bound; however, it might happen that PQ is empty and a peer is chosen at random.

To add flexibility to the meeting strategy, and to increase the chances that even peers far from each other will eventually meet, with a certain probability α a random meeting will occur even with a non-empty priority queue (lines 1-2). When a peer P has been chosen, the *meeting*() and *gossip*() procedures are invoked (lines 4-5) and, if P has become a neighbor of P_Θ, the search radius is updated (line 6). Note that if r_Θ becomes smaller than any lower-bound in PQ (line 7), the priority queue may be safely emptied because it does not contain any peer whose distance is lower than r_Θ.

5. COST ANALYSIS

In this section, we derive a cost model for Algorithm 4.1, which describes a meeting between two peers P_Θ and P_Ψ. In the first phase of the algorithm, peer P_Θ asks P_Ψ for one BF at a time and, at each step i of the algorithm, checks whether the condition $lb^{(i)} < r_\Theta$ is satisfied, before continuing the meeting. The second phase is analogous to the first one, with P_Ψ asking P_Θ for its BF's. To characterize the cost of the meeting, we introduce the probability distribution $G_P^{(i)}(r)$,

which denotes the probability that, upon the reception of i BF's, peer P is still interested in the meeting:

$$G_P^{(i)}(r) = \Pr\left\{lb^{(i)} \leq r,\ lb^{(i)} = lb(BF^{(1)}, \ldots, BF^{(i)})\right\}. \quad (14)$$

The cost of the i-th step of the first phase of the algorithm, then, is characterized by the cost $\mathcal{C}_{BF}$ of a Bloom-filter (proportional to its size m), multiplied by the probability that the BF is needed by P_Θ. The cost $\mathcal{C}_\Theta$ for the first phase, then, is given by

$$\mathcal{C}_\Theta = \mathcal{C}_{BF} \cdot \left[1 + \sum_{i=1}^{I-1} G_P^{(i)}(r_\Theta)\right]. \quad (15)$$

Note that the sum ranges between 0 and $I-1$, with $G_P^{(0)}(r) = 1$, assuming that no prior information is given about peers that have not been met.

The cost $\mathcal{C}_\Psi$, which characterizes the second phase of Algorithm 4.1, can be expressed as:

$$\mathcal{C}_\Psi = G_P^{(I-1)}(r_\Theta) \cdot \mathcal{C}_{\Psi,1} + \left[1 - G_P^{(I-1)}(r_\Theta)\right] \cdot \mathcal{C}_{\Psi,2} \quad (16)$$

where P_Θ, with probability $G_P^{(I-1)}(r_\Theta)$, has computed the distance $\widetilde{d}$ (i.e., it has asked for I BF's) and, with probability $1-G_P^{(I-1)}(r_\Theta)$, has computed a lower-bound using $i^* < I$ BF's.

$$\begin{cases} \mathcal{C}_{\Psi,1} = I \cdot \mathcal{C}_{BF} \cdot G_P^{(I)}(r_\Theta) \\ \mathcal{C}_{\Psi,2} = \mathcal{C}_{BF} \cdot \left[1 + \sum_{i=1}^{I-1} G_P^{(i)}(r_\Psi)\right] \cdot G_P^{(i^*)}(r_\Psi) \end{cases} \quad (17)$$

In $\mathcal{C}_{\Psi,1}$'s expression, P_Ψ will not ask for any BF, unless it wants P_Θ to become its neighbor (a lower-bound computed with I BF's is equal to the distance $\widetilde{d}$). On the other hand, to compute $\mathcal{C}_{\Psi,2}$, we take into consideration the probability that P_Ψ stops the meeting by means of P_Θ's lower-bound (we approximate the probability $\Pr\{lb^{(i^*)} \leq r_\Psi\}$ with $G_P^{(i^*)}(r_\Psi)$); if this is not the case, and P_Ψ has to compute its own lower-bound, the same considerations valid for $\mathcal{C}_\Theta$ apply.

The overall cost is proportional to the number I of BF's that are sent across the network, which should be kept as low as possible (note that this can be done at the price of a higher absolute error ε_{abs}); on the other hand, by increasing I, the average number n of elements inserted in each BF will automatically decrease, thus reducing the cost $\mathcal{C}_{BF}$ as well. This suggests that there exists an optimal I which minimizes transmission costs: computing this optimal value, however, is not easy because the data distribution among peers, which is represented by the $G_P^{(i)}(r)$ functions, should also be taken into account, since they play a key role in the cost model. If we consider, for example, $\mathcal{C}_\Theta$ and $\mathcal{C}_{\Psi,2}$ in Equations 15 and 17, we observe that whenever we have to deal with a set of $G_P^{(i)}(r)$ functions that are fast decreasing on i, a higher number I of BF's could still be acceptable because, on average, only a subset of the whole LM will be asked by a peer during a meeting.

6. CONCLUSION AND FUTURE WORK

In this work we have presented an efficient technique for building a Semantic Overlay Network which takes advantage of a metric distance based on the JS-divergence to compute the similarity between peers. We have described a complete architecture whose key idea is that of associating with each peer a local view of the network that will be exploited during query routing.

We are planning to extensively test our system in the near future, in particular in terms of systems performances. We are particularly interested in understanding how to adapt query processing techniques so as to fully exploit the metric properties of the network. This may involve combining our approach with distributed indexes for metric spaces (e.g., work along the lines of [3, 14, 4]).

We believe that our approach has applicability beyond information retrieval. In many modern information systems, not only distributions in data collections, but also user preferences, recommendations, profiles, and other elements of user behavior can be described in terms of LM-style distributions with information-theoretic comparisons being natural measures of similarity. Thus, embedding measures like KL divergence or JS divergence in a metric search space is an appealing direction for much broader classes of social networks.

7. REFERENCES

[1] K. Aberer and P. Cudré-Mauroux. Semantic overlay networks. In *VLDB Tutorial*, page 1367, Aug. 2005.

[2] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden. DL meets P2P - distributed document retrieval based on classification and content. In *ECDL*, pages 379–390, Sep. 2005.

[3] M. Batko, C. Gennaro, and P. Zezula. A scalable nearest neighbor search in p2p systems. In *DBISP2P*, pages 79–92, Aug. 2004.

[4] M. Batko, D. Novk, F. Falchi, and P. Zezula. On scalability of the similarity search in the world of peers. In *INFOSCALE*, May 2006.

[5] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Improving collection selection with overlap awareness in P2P search engines. In *SIGIR*, pages 67–74, Aug. 2005.

[6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[7] A. Broder and M. Mitzenmacher. Network applications of bloom filters: a survey. *Internet Mathematics*, 1(4):485–509, 2003.

[8] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR*, pages 21–28, July 1995.

[9] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.

[10] A. Crespo and H. Garcia-Molina. Semantic overlay networks for P2P systems. In *AP2PC*, pages 1–13, July 2004.

[11] C. Doulkeridis, K. Noervaag, and M. Vazirgiannis. Scalable semantic overlay generation for P2P-based digital libraries. In *ECDL*, Sep. 2006.

[12] C. Doulkeridis, K. Noervaag, and M. Vazirgiannis. The SOWES approach to P2P web search using semantic overlays. In *WWW*, pages 1027–1028, May 2006.

[13] D. M. Endres and J. E. Schindelin. A new metric for probability distributions. *IEEE Trans. Inf. Theory*, 49(7):1858–1860, July 2003.

[14] F. Falchi, C. Gennaro, and P. Zezula. A content-addressable network for similarity search in metric spaces. In *DBISP2P*, pages 126–137, Aug. 2005.

[15] P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, Sep. 2002.

[16] L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, June 1999.

[17] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, Dec. 2003.

[18] P. Kalnis, W. S. Ng, B. C. Ooi, and K.-L. Tan. Answering similarity queries in peer-to-peer networks. *Inf. Syst.*, 31(1):57–72, Mar. 2006.

[19] A. C. König and G. Weikum. Automatic tuning of data synopses. *Inf. Syst.*, 28(1-2):85–109, Mar. 2003.

[20] X. Liu and W. B. Croft. Statistical language modeling for information retrieval. *Annual Review of Information Science and Technology*, 39:3–31, 2005.

[21] A. Löser, C. Tempich, B. Quilitz, W.-T. Balke, S. Staab, and W. Nejdl. Searching dynamic communities with personal indexes. In *ISWC*, pages 491–505, Nov. 2005.

[22] J. Lu and J. Callan. Federated search of text-based digital libraries in hierarchical peer-to-peer networks. In *ECIR*, pages 52–66, Mar. 2005.

[23] J. Lu and J. Callan. User modeling for full-text federated search in peer-to-peer networks. In *SIGIR*, aug 2006.

[24] J. Lu and J. P. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, Nov. 2003.

[25] P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *SPAA*, pages 155–164, July 2005.

[26] P. Mahlmann and C. Schindelhauer. Distributed random digraph transformations for peer-to-peer networks. In *SPAA*, Aug. 2006.

[27] W. Meng, C. T. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Comput. Surv.*, 34(1):48–89, Mar. 2002.

[28] S. Michel, M. Bender, P. Triantafillou, and G. Weikum. Iqn routing: Integrating quality and novelty in P2P querying and ranking. In *EDBT*, pages 149–166, Mar. 2006.

[29] H. Nottelmann and N. Fuhr. Combining CORI and the decision-theoretic approach for advanced resource selection. In *ECIR*, pages 138–153, Apr. 2004.

[30] H. Nottelmann and N. Fuhr. Comparing different architectures for query routing in peer-to-peer networks. In *ECIR*, pages 253–264, Apr. 2006.

[31] J. X. Parreira, S. Michel, and G. Weikum. p2pDating: Real life inspired semantic overlay networks for web search. In *SIGIR workshop on Heterogeneous and Distributed Information Retrieval*, aug 2005.

[32] I. Podnar, M. Rajman, T. Luu, F. Klemm, and K. Aberer. Beyond term indexing: A P2P framework for web information retrieval. *Informatica*, 30(2):153–161, June 2006.

[33] R. Steinmetz and K. Wehrle. *Peer-to-Peer Systems and Applications*. Springer, 2005.

[34] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer-Verlag New York, Inc., 2005.

[35] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, Apr. 2004.