Improved Search for Socially Annotated Data

Nikos Sarkas
University of Toronto
nsarkas@cs.toronto.edu

Gautam Das
University of Texas at Arlington
gdas@cse.uta.edu

Nick Koudas
University of Toronto
koudas@cs.toronto.edu

ABSTRACT

Social annotation is an intuitive, on-line, collaborative process through which each element of a collection of resources (e.g., URLs, pictures, videos, etc.) is associated with a group of descriptive keywords, widely known as tags. Each such group is a concise and accurate summary of the relevant resource's content and is obtained via aggregating the opinion of individual users, as expressed in the form of short tag sequences. The availability of this information gives rise to a new searching paradigm where resources are retrieved and ranked based on the similarity of a keyword query to their accompanying tags.

In this paper, we present a principled and efficient search and resource ranking methodology that utilizes exclusively the user-assigned tag sequences. Ranking is based on solid probabilistic foundations and our growing understanding of the dynamics and structure of the social annotation process, which we capture by employing powerful interpolated n-gram models on the tag sequences. The efficiency and applicability of the proposed solution to large data sets is guaranteed through the introduction of a novel and highly scalable constrained optimization framework, employed both for training and incrementally maintaining the n-gram models.

We experimentally validate the efficiency and effectiveness of our solutions compared to other applicable approaches. Our evaluation is based on a large crawl of del.icio.us, numbering hundreds of thousands of users and millions of resources, thus demonstrating the applicability of our solutions to real-life, large scale systems. In particular, we demonstrate that the use of interpolated n-grams for modeling tag sequences results in superior ranking effectiveness, while the proposed optimization framework is superior in terms of performance both for obtaining ranking parameters and incrementally maintaining them.

1. INTRODUCTION

Social annotation, also referred to as *collaborative tagging*, has been constantly building momentum since its recent inception and has now reached the critical mass required for driving exciting new applications. On September 2006 del.icio.us reported 1 million registered users, while YouTube claimed 500 thousand registered users and stored 6 million videos; a figure that increased to 83 million by April 2008. Flickr had 7 million registered users and stored 2 billion images by November 2007. Our own Spring 2007 crawl of del.icio.us revealed 570 thousand registered users and 24 million URLs. Given that the user base and content of such sites has been observed to double every few months, these numbers only loosely approximate the immense popularity and size of systems that employ social annotation.

Users of an on-line, collaborative tagging system add to their personal collection a number of resources (e.g., URLs, pictures, videos, etc.) and associate with each of them a short sequence of keywords, widely known as *tags*. Each *tag sequence*, referred to as an *assignment*, is a concise and accurate summary of the relevant resource's content according to the user's opinion. The premise of annotating resources in that manner is the subsequent use of tags in order to facilitate the searching and navigation of one's personal collection.

As an example, del.icio.us users add to their collection the URLs of interesting Web pages and annotate them with tags so that they can subsequently search for them easily. Users can discover and add URLs to their collection by browsing the web, searching in del.icio.us or browsing the collections of other users. Given the considerable overlap among the individual collections, resources accumulate a large number of assignments, each one of them posted by a different individual.

This information is publicly available and gives rise to a new searching paradigm where resources are retrieved based on the similarity of a query to their accompanying tags. The advantages of such an approach are immense. When annotating a resource, users distil its complex content into an accurate and concentrated textual summary. Subsequent aggregation of the individual opinions into a collective wisdom serves to eliminate noise and increase our confidence, thus offering an accurate textual description of a resource's content. Consequently, social annotation (a) enables the extension of the keyword search model to *non-textual* objects of arbitrarily high complexity, like videos, images and music, and (b) enhances our ability to identify and retrieve relevant textual objects in response to a keyword query, since we no longer need to infer by means of heuristics which of the words and phrases present in the text are truly representative of its content.

In this spirit, we propose *RadING* (**R**anking **a**nnotated **d**ata using **I**nterpolated **N**-**G**rams), a principled and efficient search and ranking methodology that exclusively utilizes the user-assigned tag sequences. The solution employs powerful *interpolated n-grams* to model the tag sequences associated with each resource, motivated by our growing understanding of the dynamics and structure of the social annotation process. The interpolated n-grams employed are a more robust variation of vanilla n-gram models – commonly used

VLDB '09, August 24-28, 2009, Lyon, France

to model keyword (and more generally event) sequences – that linearly combine information from all lower order n-grams, i.e, n-grams, $(n-1)$-grams and so on. In our application, the use of interpolated n-gram models exposes significant and highly informative tag co-occurrence patterns (correlations) present in the user assignments. Our ranking strategy leverages this information in order to identify the resources most relevant to a query and rank them accurately.

In order to guarantee the scalability of our approach to millions of resources, annotated with hundreds to thousands of assignments, we also introduce a novel optimization framework, employed both in the training and the incremental maintenance of the interpolated n-gram models. The optimization framework is able to rapidly identify the optimal weighting that must be assigned to each lower order n-gram, as well as efficiently update them as resources accumulate new assignments.

More specifically, we make the following contributions:

- We present RadING, a principled search and resource ranking methodology that utilizes interpolated n-grams to model the tag sequences associated with every resource. The approach is based on solid probabilistic foundations and our insight of the collaborative tagging process.

- The training and the incremental maintenance of the interpolated n-gram models is performed by means of a novel constrained optimization framework that employs powerful numerical optimization techniques and exploits the unique properties of both the function to be optimized and the parameter domain. We demonstrate that our framework outperforms at both tasks and by a large margin other applicable techniques.

- We experimentally validate the effectiveness of the proposed ranking methodology and the efficiency of the n-gram training and maintenance framework using data from a large crawl of del.icio.us, numbering hundreds of thousands of users and millions of resources, thus demonstrating the applicability of our approach to real-life, large scale systems.

The rest of the paper is organized as follows. In Section 2 we review related work. In Section 3 we present and motivate the n-gram based ranking methodology, while in Section 4 we present our novel n-gram training solution. In Section 5 we briefly discuss how a real-life system can implement the proposed searching solution. Section 6 presents our experimental evaluation and, lastly, Section 7 offers our conclusions.

2. RELATED WORK

Research on collaborative tagging has so far followed two distinct directions. One direction focuses on utilizing this newly-found wealth of information in the form of tags to enhance existing applications (like searching) and develop new ones. The second attempts to understand, analyze and model the various aspects of the social annotation process. However, to the best of our knowledge, our growing insight and understanding of the tagging process has not been so far utilized in a principled manner.

With respect to searching for and ranking items from a tagged collection, Hotho et al. [12] propose a static, query-independent ranking of the resources (as well as of users and tags) based on an adaptation of the PageRank algorithm [5]. Users, resources and tags are first organized in a tripartite graph, whose hyper-edges are links of the form (user,resource,tag). This graph is then collapsed into a normal undirected graph whose nodes represent indiscriminately users, resources and tags, while edge weights count co-occurrences of the entities in the hyper-edges of the original graph. The PageRank algorithm is then applied, producing a total ordering involving all three types of entities. The semantics of the resulting ranking (given the transformation of the tri-partite graph to an undirected graph) are questionable. In an attempt to rectify, the authors suggest a per "keyword" static ranking of the resources, once more based on their PageRank adaptation, which is clearly not scalable. Incremental maintenance issues are not addressed at all. Overall this is a heuristic adaptation of the PageRank algorithm with unclear semantics.

Bao et al. [3] adapt a machine learning approach to ranking, when the resources are Web pages. A support vector machine is used to "learn" the ranking function [14] which utilizes five different features of the pages: the tf/idf similarity of the page and the query, two different similarity measures between the query and the tags, its PageRank and the PageRank adaptation that was discussed before. Their technique however, is only limited to Web pages and is based on ad-hoc heuristics. Scalability issues and incremental maintenance issues under dynamic updates are left unspecified.

Amer-Yahia et al. [2] propose a solution for ranking resources efficiently, under the constraint that only the assignments posted by users in social network neighborhoods are to be used, thus personalizing query results. Their framework is complementary to ours as it can be used in combination with monotonic ranking functions, like the one that we propose in this work.

Search using tags can be viewed as a generalization of keyword-based search of non-textual content (e.g., images) using textual hints found in HTML code. While the corresponding *alt* HTML attribute or image name provides a single "assignment" (the description provided by a single person), which we have no option but to blindly trust, social annotation systems aggregate the opinion of many users, thus eliminating noise and increasing our confidence that the assignments are relevant to a resource.

Besides ranking, researchers have also looked into other interesting problems related to collaborative tagging. Hotho et al. [13] use the PageRank adaptation presented in [12] in order to detect a variety of trends in a collaborative tagging system. Li et al. [17] organize the tags in a loose hierarchical structure in order to facilitate browsing and exploration of tags and resources. Chirita et al. [8] present a system for automatically suggesting personalized tags for a Web page.

Another body of work is concerned with the analysis and modeling of collaborative tagging systems [9, 10, 6, 11]. [9, 10] observed that the distribution of tags assigned to a resource converges rapidly to a remarkably stable heavy-tailed distribution. [9] concentrates on identifying the user behavior that leads to this phenomenon, while [10] attempts to mathematically model it. [6] on the other hand, explores and models the co-occurrence patterns of tags across resources. Finally, Heyman et al [11] investigate whether the additional information provided by the social annotations has the potential to improve Web Search and offer positive conclusions.

Language models for information retrieval have been previously introduced [25, 21]. Our approach is essentially a language model; however it is based on different principles (pertinent to our specific application) and the resulting ranking strategy is obtained through a different theoretical derivation. Furthermore, unlike previous approaches (e.g., [25, 21]), we are also concerned with the efficiency and incremental maintenance issues of the proposed solution.

Lastly, the optimization of general objective functions, in the presence or absence of domain constraints, is a well-studied subject [4, 23, 18]. In this work, we do not introduce a new generic opti-

mization technique, but rather an optimization framework which exploits the unique properties of our problem and enables the use of otherwise inapplicable, unconstrained optimization techniques.

3. PRINCIPLED RANKING OF ANNOTATED RESOURCES

In this section we derive and motivate the RadING searching and ranking strategy. We begin by presenting its solid foundations in probabilistic information retrieval [25, 21] and our understanding of the social annotation process [9, 10], which is subsequently modeled by employing progressively more sophisticated n-gram models [16, 20, 15].

3.1 Probabilistic Foundations

The basis of probabilistic information retrieval is the ranking of the resources according to the probability of each resource being relevant to the query [25, 21], i.e., given a keyword query Q and a collection of tagged resources $\{R\}$, it is desirable to rank them in descending order of $p(R \textit{ is relevant } |Q)$. By applying Bayes' rule we have that

$$p(R \textit{ is relevant } |Q) = \frac{p(Q|R \textit{ is relevant})p(R \textit{ is relevant})}{p(Q)}$$

The term $p(R \textit{ is relevant})$ is the a-priori probability that resource R is relevant, independently of the query being posed. This term can potentially be used to bias the ranking towards certain categories of resources in a domain-, application- or even user-specific manner. In what follows, we assume that this prior probability is constant throughout our resource collection, without affecting the analysis and results that will be subsequently presented. We revisit this issue and offer our suggestions for non-uniform priors that could be employed in Section 6.

Term $p(Q)$ is the a-priori probability of the query being issued, which, since the query is given, is constant for all resources and therefore does not affect their relative ranking.

Based on the aforementioned observations, ranking the resources according to $p(R \textit{ is relevant } |Q)$ is equivalent to a ranking based on $p(Q|R \textit{ is relevant})$. This term captures the intuition that a resource can be retrieved by a number of different queries, however not all queries are equally likely to be used for this purpose.

EXAMPLE 1. *Consider the web page of the Firefox browser. Perhaps the most reasonable, and therefore probable, keyword query that one would use in order to retrieve the web page is "firefox". Nevertheless, it is not the unique query that can potentially be used to identify the web page: "mozilla browser" or "open source browser" are other perfectly valid query candidates that we can expect, albeit with a lower probability.*

More formally, we established that:

$$p(R \textit{ is relevant } |Q) \propto p(Q|R \textit{ is relevant})$$

This simple transformation implies that resources need to be modeled so that we can estimate the probability of the query being "generated" by each resource. While the problem of ranking query results has been viewed from this perspective before [25, 21], the appropriate modeling of the resources is decisive in producing an intuitive ordering and is sensitive to the characteristic of the application domain. In what follows, we will discuss how this probability can be modeled in a meaningful and principled manner by studying the social annotation process and motivating the use of language models.

3.2 Dynamics and Properties of the Social Annotation Process

Users annotate resources in order to facilitate their future retrieval. We assign to a resource the tags that we would instinctively use in the future in order to retrieve it from our personal collection of resources. Therefore, although we tag resources in a personal and idiosyncratic manner, the underlying goal of the tagging process is to describe the resource's content in a concise and accurate manner, so that we can easily locate it when the need arises.

Even though a resource is annotated by hundreds or thousands of individuals, its content can only be viewed from a limited number of perspectives, so that even after witnessing a small number of assignments, we should be able to identify the annotation trends associated with these perspectives. The annotation of a resource with additional assignments will increase our confidence in the trends already identified, but is unlikely to unveil a fresh prevalent perspective.

This intuitive observation has also been validated by previous work on the dynamics of collaborative tagging. [9, 10] demonstrated that the distribution of tags for a specific resource converges rapidly to a remarkably stable, heavy tailed distribution that is lightly affected by additional assignments. The heavy tailed distribution ascertains the dominance of a handful of influential trends in describing a resource's content. The rapid convergence and the stability of the distribution points to its predictability: namely, after witnessing a small number of assignments, we should be able to predict with a high degree of confidence subsequent tag assignments.

Given the fast crystallization of users' opinion about the content of a resource, we can make a natural assumption that will serve as a bridge between our ability to predict the future tagging activity of a resource and our need to compute $p(Q|R \textit{ is relevant})$.

Users will use keyword sequences derived from the same distribution to both tag and search for a resource.

This logical link allows us to equate the probability $p(Q|R \textit{ is relevant})$ to the probability of an assignment containing the same keywords as Q being used to tag the resource, i.e.,

$$p(Q|R \textit{ is relevant}) = p(Q \textit{ is used to tag } R)$$

The stability of the tag distribution allows us to accurately estimate the probability of a tag being used in the future, based on the resource's tagging history. However, assignments are rarely comprised by a single tag. In our study (Section 6) we observed that the average length of an assignment is 2.77 tags. It is reasonable to expect that neither the order in which tags are placed in an assignment, nor the co-occurrence patterns of tags in assignments are random.

In fact, [9] observed that tags are not used in random positions within an assignment, but rather progress (from left to right) from more general to more specific and idiosyncratic. Therefore, assignments are not orderless sets of tags, but sequences of tags, whose ordering tends to be consistent across the assignments attached to a resource, and consequently the queries used to search for it.

Additionally, tags representing different perspectives about a resource's content, although popular in their own right, are less likely to co-occur in the same assignment.

EXAMPLE 2. *In our del.icio.us crawl, the Mozilla project main page is heavily annotated with tags "opensource", "mozilla" and "firefox". We observed that tags "opensource" and "firefox" appear together much less frequently than expected given their popularity, demonstrating two different perspectives for viewing the*

web site: as the home of the Firefox browser or as an open source project. Such statistical deviations, more or less severe, were observed throughout the del.icio.us collection.

Therefore, the assignments comprising the tagging history of a resource are sequences of tags exhibiting strong tag co-occurrence patterns. In order to accurately estimate the probability of a tag sequence Q being assigned to a resource R, we need to capture this elaborate structure. Simply taking into account the frequencies (in R's history) of the tags comprising Q can lead to gross miscalculations. To this end, we propose the use of sequential n-gram models [16, 20, 15], that can effectively model such co-occurrence patterns present in the assignments (tag sequences) comprising a resource's tagging history.

3.3 N-gram Models

Consider an assignment comprised of a particular sequence s of l tags, $t_1, \ldots, t_l$, ordered from left to right. We are interested in calculating the probability of this sequence of tags being assigned to a resource. More formally,we are interested in computing the probability $p(t_1, \ldots, t_n)$. By employing the chain rule of probability, we can express it as:

$$p(t_1, \ldots, t_l) = p(t_1)p(t_2|t_1) \cdots p(t_l|t_1, \ldots, t_{l-1})$$

$$= \prod_{k=1}^{l} p(t_k|t_1, \ldots, t_{k-1})$$

This formula links the probability of a tag t_k appearing in sequence s to its preceding tags $t_1, \ldots, t_{k-1}$. In other words, the probability of a tag appearing in the sequence depends on all of the preceding tags. The intuition behind n-gram models is to compute this probability by approximating the preceding subsequence with only the last $n-1$ tags:

$$p(t_k|t_1, \ldots, t_{k-1}) \simeq p(t_k|t_{k-n+1}, \ldots, t_{k-1})$$

The most commonly used n-gram models are the 1-gram or *unigram* model, so that $p(t_k|t_1, \ldots, t_{k-1}) = p(t_k)$, the 2-gram or *bigram* model, with $p(t_k|t_1, \ldots, t_{k-1}) = p(t_k|t_{k-1})$, and the 3-gram or *trigram* model that approximates $p(t_k|\ t_1, \ldots, t_{k-1})$= $p(t_k|t_{k-2}, t_{k-1})$. It is clear that the use of n-gram models is associated with an inherent trade-off. Higher order models utilize more information and are able to approximate $p(t_k|t_1, \ldots, t_{k-1})$ more accurately, at the expense of an increased storage and computation overhead.

In order to ease notation we use the bigram model in our examples and mathematical formulations, since the concepts can be easily generalized for higher order n-gram models. Under the bigram model, the probability of a tag appearing in the sequence depends only on the preceding tag so that:

$$p(t_k|t_1, \ldots, t_{k-1}) = p(t_k|t_{k-1})$$

$$p(t_1, \ldots, t_l) = \prod_{k=1}^{l} p(t_k|t_{k-1})$$

Each adjacent pair of tags (words) in a sequence (assignment) is also known as a bigram, but it will be clear from the context whether we refer to the model or to a pair of adjacent tags. Similarly, a single tag will be referred to as a unigram. The bigram probabilities $p(t_k|t_{k-1})$ for a resource can be computed from its tagging history, by using its previously posted assignments as training data. The most natural way to estimate $p(t_k|t_{k-1})$ is by using Maximum Likelihood Estimation (MLE). Then, the probability of a bigram t_1, t_2 is computed as:

$$p(t_2|t_1) = \frac{c(t_1, t_2)}{\sum_t c(t_1, t)}$$

where $c(t_1, t_2)$ are the number of occurrences of the corresponding bigram t_1, t_2 in the training data, that is the assignments associated with the resource, and $\sum_t c(t_1, t)$ is the sum of the occurrences of all different bigrams involving t_1 as the first tag.

Summarizing our approach, in order to compute the probability that a given tag sequence $Q = t_1, \ldots, t_l$ is used to annotate a resource R, which as we discussed in Section 3.2 will enable us to rank the resources according to their relevance to Q, we use the past tagging activity of the users in order to train a bigram model for each resource in our collection. The bigram models can then be used to evaluate the probability $p(Q \textit{ is used to tag } R) = p(t_1, \ldots, t_l|R)$ for each resource R.

3.4 Interpolation

A limitation of the plain bigram model presented previously is the problem of sparse data [25, 16, 20, 15]. Because the size of the data used to train the model is typically limited, the probability of any bigram t_1, t_2 not appearing at least once in the training data will be zero, since $c(t_1, t_2) = 0$. This is undesirable as any sequence of tags that contains a bigram never seen before, will evaluate to zero probability. As an example, consider a resource heavily tagged with the words "Toronto" and "snow". If for some reason both tags fail to appear in adjacent positions in any assignment, the resource should intuitively be less relevant to the query "Toronto snow", but not completely irrelevant.

To compensate for this limitation, a wealth of *smoothing* techniques can be employed [16, 20, 15]. The idea motivating these methods is that the bigram count distribution should be made smoother by subtracting a bit of probability mass from higher counts and distributing it amidst the zero counts, so that no bigram evaluates to zero probability.

For our purposes we employ the widely-used, intuitive and powerful Jelinek-Mercer linear interpolation technique. Let us consider a bigram t_1, t_2 and let $\hat{p}(t_2|t_1)$ and $\hat{p}(t_2)$ be the MLE bigram and unigram estimates respectively. The unigram MLE estimate is simply the number of times that a tag appears in the training data over the total number of tags. Then the bigram probability is provided by linearly interpolating both MLE estimates:

$$p(t_2|t_1) = \lambda_2\hat{p}(t_2|t_1) + \lambda_1\hat{p}(t_2), \quad \lambda_1 + \lambda_2 = 1$$

The motivation behind this solution is that when there is insufficient data to estimate a probability in the higher-order model (bigram), the lower-order model (unigram) can provide useful information.

Motivated similarly, it is common practise to also interpolate a bigram t_1, t_2 using the probability $p_{bg}(t_2)$ of t_2 appearing in random text. In our case, we interpolate with the background probability of the tag being used by a user, which we estimate as the total number of times this tag was used in the context of any resource in the collection, over the total number of tags assigned to the resources of the collection. By using the background probability of a tag as an interpolation factor, it is possible to assign non-zero (but small) probability to sequences (queries) that contain tags not appearing a resource's history. Intuitively, a resource tagged with "Toronto", but not "snow", should be somewhat relevant to query

"Toronto snow" and not completely irrelevant. Finally, the Jelinek-Mercer estimate of a bigram is:

$$p(t_2|t_1) = \lambda_2\hat{p}(t_2|t_1) + \lambda_1\hat{p}(t_2) + \lambda_0 p_{bg}(t_2)$$
$$0 \leq \lambda_0, \lambda_1, \lambda_2 \leq 1, \quad \lambda_0 + \lambda_1 + \lambda_2 = 1$$

or

$$p(t_2|t_1) = \lambda_2\hat{p}(t_2|t_1) + \lambda_1\hat{p}(t_2) + (1 - \lambda_1 - \lambda_2)p_{bg}(t_2)$$
$$0 \leq \lambda_1, \lambda_2 \leq 1, \quad \lambda_1 + \lambda_2 \leq 1$$

3.5 Advantages of Linear Interpolation

Although, as was mentioned, there exists a wealth of n-gram smoothing methods [16, 20, 15], the use of the Jelinek-Mercer linear interpolation technique offers two unique advantages, besides its great smoothing performance.

The first is our ability to devise a novel and efficient method for initially setting and subsequently maintaining, as new assignments are attached to a resource, the parameters of the corresponding per-resource smoothed n-gram models. The technique, which we present in Section 4, guarantees the applicability of the proposed n-gram based ranking approach to real-life systems of immense size, containing hundred of millions or even billions of resources.

Secondly, the linearly interpolated bigram models can be associated with the social annotation process in a very natural and intuitive manner. The probability of a bigram t_1, t_2 is computed as the weighted average of its MLE bigram probability $\hat{p}(t_2|t_1)$, its MLE unigram probability $\hat{p}(t_2)$ and its background probability $p_{bg}(t_2)$. The values of the interpolation parameters λ_2, λ_1 and λ_0, signify our confidence into each of these three sources of information.

Consider a resource that has been annotated with almost random tags, so that all assignments are in disagreement. In that case, little information can be derived from the assignments' content and the relevant bigram and unigram probabilities that have been extracted from them. This should be reflected in the parameters by setting λ_2 and λ_1 to low values. If the assignments of a resource are in agreement, but exhibit no correlation in the co-occurrence patterns of tags, then we should place high confidence in the unigram probabilities (λ_1) computed, but lower in the respective bigram probabilities (λ_2). Lastly, if assignments are in compliance and exhibit strong tag co-occurrence patterns, we should place our trust in the bigram probabilities computed, thus setting parameter λ_2 to a high value.

4. PARAMETER OPTIMIZATION

Setting the interpolation parameters to meaningful and appropriate values is a challenge that needs to be addressed. In this section we discuss the algorithm used currently for setting the parameters, as well as its limitations, and introduce a novel adaptation of powerful optimization algorithms for handling the problem much more efficiently. In our exposition we use the bigram model and generalize our results to n-gram models at the end of the section.

4.1 Likelihood Function

The intuitive parameter setting procedure that we described in Section 3.5 can be performed by dividing the training data (tagging history) into two sets. The first is used to compute the MLE estimates, while the second, known as held-out set, is used for "learning" the parameters λ_i. The interpolation parameters are set to the values that maximize the likelihood of the held-out set being generated by the interpolated bigram model. In our case we can divide the assignments into two groups, constituting the training and held-out data.

Let us compute the (log)likelihood function that needs to be maximized. Suppose that the held-out data set contains m assignments $a_1, \ldots, a_i, \ldots, a_m$ each one of them containing $k(i)$ tags, $t_{i1}, \ldots, t_{ik(i)}$. The likelihood of an assignment is:

$$\log p(a_i) = \log \prod_{j=1}^{k(i)} p(t_{ij}|t_{i(j-1)}) = \sum_{j=1}^{k(i)} \log p(t_{ij}|t_{i(j-1)})$$

Since assignments are generated independently, by different users, the likelihood of all assignments in the held-out data is

$$\log \prod_{i=1}^{m} p(a_i) = \sum_{i=1}^{m} \log p(a_i) = \sum_{i=1}^{m} \sum_{j=1}^{k(i)} \log p(t_{ij}|t_{i(j-1)})$$

Notice that this is the sum of the log-probabilities of all bigrams in the held-out set. To ease notation, we will consider that the training set is comprised of $l = \sum_{j=1}^{m} k(j)$ bigrams $t_{i1}t_{i2}$, $i = 1 \ldots l$. Then, the likelihood can be written as $\sum_{i=1}^{l} \log p(t_{i2}|t_{i1})$.

Since we are using a bigram model, $p(t_{i2}|t_{i1}) = \lambda_2\hat{p}(t_{i2}|t_{i1}) + \lambda_1\hat{p}(t_{i2}) + (1 - \lambda_1 - \lambda_2)p_{bg}(t_{i2})$. In order to further ease notation we write

$$p(t_{i2}|t_{i1}) = \lambda_2 p_{i2} + \lambda_1 p_{i1} + p_{i0}$$

where $p_{i2} = \hat{p}(t_{i2}|t_{i1}) - p_{bg}(t_{i2})$, $p_{i1} = \hat{p}(t_{i2}) - p_{bg}(t_{i2})$ and $p_{i0} = p_{bg}(t_{i2})$.

Then, the likelihood function that needs to be maximized is:

$$L(\lambda_1, \lambda_2) = \sum_{i=1}^{l} \log(\lambda_2 p_{i2} + \lambda_1 p_{i1} + p_{i0})$$

An important observation that simplifies the maximization problem is that the function $L(\lambda_1, \lambda_2)$ is *concave* [4].

DEFINITION 1. *A function $f : D \to \Re$ is concave if $\forall \mathbf{x}, \mathbf{y} \in D$ and $0 \leq \theta \leq 1$, we have that $f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \geq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})$.*

Concavity is essentially the symmetric property of convexity. A function f is concave iff $-f$ is convex. An important property of concave functions is the following [4].

THEOREM 1. *If $f : D \to \Re$ is concave, any point that is a local maximum is also a global maximum.*

Therefore, any optimization procedure that converges to a local maximum will identify the global maximum of the function. The concavity of $L(\lambda_1, \lambda_2)$ can be easily demonstrated using the properties of concave functions [4].

Although the concavity of $L(\lambda_1, \lambda_2)$ simplifies the optimization problem due to the absence of local optima, a complication that needs to be considered is the constrained domain of λ_1, λ_2: remember that $0 \leq \lambda_1, \lambda_2 \leq 1, \lambda_1 + \lambda_2 \leq 1$. We will denote this constrained domain as D^*. The original domain $D \supseteq D^*$ of $L(\lambda_1, \lambda_2)$ depends on the specific values of p_{i2}, p_{i1}, p_{i0}. Figure 1 illustrates the constrained domain D^*.

Let us denote with $\lambda^* = (\lambda_1^*, \lambda_2^*)$ the global maximum (if it exists) of $L(\lambda_1, \lambda_2)$, and let λ^c be the point where $L(\lambda_1, \lambda_2)$ evaluates to its maximum value within D^*. If $\lambda^* \in D^*$, then $\lambda^* = \lambda^c$. However, it is possible that $\lambda^* \notin D^*$ or that $L(\lambda_1, \lambda_2)$ is unbounded, i.e., $\lim_{\lambda_1 \to \infty} L(\lambda_1, \lambda_2) = \infty$ or $\lim_{\lambda_2 \to \infty} L(\lambda_1, \lambda_2) = \infty$. In these cases λ^c must be identified. Our goal in optimizing $L(\lambda_1, \lambda_2)$ is locating λ^c, regardless whether $\lambda^* = \lambda^c$ or not.

4.2 EM Algorithm

The standard method [16, 7] for optimizing the likelihood function and setting the parameters is by using the Expectation-Maximization (EM) algorithm. The EM algorithm is an iterative optimization procedure commonly used for optimizing the objective functions of probabilistic models in the presence of latent variables. Each iteration of the EM algorithm, comprised of the so called Expectation and Maximization steps, is guaranteed increase the value of the objective function, eventually converging to a local optimum.

In our case, the probability of a bigram t_1, t_2 is a weighted combination of the bigram probability $p(t_2|t_1)$, the unigram probability $p(t_2)$ and the background probability $p_{bg}(t_2)$. In other words, we have modeled the bigram probability as a *mixture* of three models and the latent variable in this case determines which of these models will be used to compute the final bigram probability. This observation is the basis for deriving the EM algorithm iterations for our application.

For each of the n bigrams in the held-out data set, we introduce two auxiliary variables q_{i1} and q_{i2}. Then:

$$\text{E-step:} \quad q_{i1}^{k+1} = \frac{\lambda_1^k(p_{i1}+p_{i0})}{\lambda_2^k p_{i2} + \lambda_1^k p_{i1} + p_{i0}}$$
$$q_{i2}^{k+1} = \frac{\lambda_2^k(p_{i2}+p_{i0})}{\lambda_2^k p_{i2} + \lambda_1^k p_{i1} + p_{i0}}$$

$$\text{M-step:} \quad \lambda_1^{k+1} = \frac{\sum_{i=1}^n q_{i1}^{k+1}}{n}$$
$$\lambda_2^{k+1} = \frac{\sum_{i=1}^n q_{i2}^{k+1}}{n}$$

Another important property of the EM algorithm in our case is that if the starting point is in D^*, then the algorithm during the search for λ^c will remain within D^*. Therefore, the EM algorithm (a) increases with every iteration the value of $L(\lambda_1, \lambda_2)$ and (b) remains within D^*. Due to these two properties, the algorithm converges to λ^c, even if $\lambda^c \neq \lambda^*$.

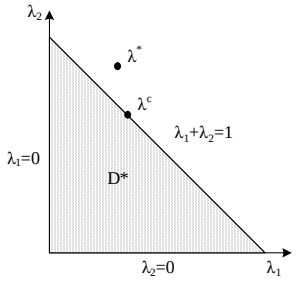


Figure 1: The constrained search space D^*.

4.3 Adapting Unconstrained Optimization Methods for Constrained Optimization

The EM algorithm is an attractive solution for optimizing $L(\lambda_1, \lambda_2)$. It is extremely simple to implement and converges to the optimal value of the parameters within the constrained domain. However, as it has been observed in practice [26] and we will also experimentally validate in Section 6, its convergence can be slow. Given that we require the technique for optimizing the interpolation parameters to be scalable to hundreds of millions of resources, annotated with hundreds to thousands of assignments, its speed is crucial for the applicability of the proposed solution.

An extremely promising alternative would be the use of efficient numerical optimization techniques. Researchers have developed algorithms for both constrained and unconstrained numerical optimization problems [4, 23, 18]. However, general constrained optimization techniques are too heavyweight and unconstrained numerical optimization methods are not directly applicable to our problem, since our goal is to maximize $L(\lambda_1, \lambda_2)$ within its constrained domain D^*.

Additionally, in our problem setting we cannot simply use Lagrange Multipliers [4] in order to incorporate the *equality* constraint ($\lambda_0 + \lambda_1 + \lambda_2 = 1$) into the objective function and thus enable the use of unconstrained optimization methods. The reason is that Lagrange multipliers cannot be used to remove the *inequality* constraints of our problem, namely $\lambda_0, \lambda_1, \lambda_2 \geq 0$.

In order to compensate, we introduce the RadING optimization framework which leverages efficient numerical optimization techniques as a primitive in order to maximize $L(\lambda_1, \lambda_2)$ within its constrained domain D^*.

In what follows, we demonstrate how this can be accomplished, depending on whether $L(\lambda_1, \lambda_2)$ is bounded (Section 4.3.1) or unbounded (Section 4.3.2). Our results are unified into a simple but powerful optimization framework (Section 4.3.3). Finally, we identify a particular numerical optimization technique with appealing properties and argue that it is an ideal candidate for use within the proposed framework (Section 4.3.4).

4.3.1 Bounded likelihood function

As was discussed in Section 4.1, the global maximum of $L(\lambda_1, \lambda_2)$ can either lie inside or outside D^*. The following two theorems demonstrate that if $\lambda^* \neq \lambda^c$, then λ^c must lie on the boundary of D^* (Figure 1).

THEOREM 2. *Let $f : D \rightarrow \Re$ be a concave function and $\mathbf{x}^*$ be its global maximum. Let also $\mathbf{x} \in D$ be a random point. Then every point $\mathbf{v} = k\mathbf{x}^* + (1-k)\mathbf{x}$, $0 \leq k \leq 1$, that is located on the segment connecting $\mathbf{x}$ and $\mathbf{x}^*$ will satisfy $f(\mathbf{x}) \leq f(\mathbf{v}) \leq f(\mathbf{x}^*)$.*

PROOF. From the definition of concavity, $f(\mathbf{v}) = f(k\mathbf{x}^* + (1-k)\mathbf{x}) \geq kf(\mathbf{x}^*) + (1-k)f(\mathbf{x})$. Since $\mathbf{x}^*$ is the global maximum, $f(\mathbf{x}^*) \geq f(\mathbf{x})$. Then, $f(\mathbf{v}) \geq kf(\mathbf{x}^*) + (1-k)f(\mathbf{x}) \geq kf(\mathbf{x}) + (1-k)f(\mathbf{x}) = f(\mathbf{x})$. Therefore, $f(\mathbf{x}) \leq f(\mathbf{v}) \leq f(\mathbf{x}^*)$. □

THEOREM 3. *Let $f : D \rightarrow \Re$ be a concave function and $D^* \subset D$ be a convex subset of the function's domain. Let $\mathbf{x}^c$ be the value that maximizes f within D^* and $\mathbf{x}^*$ the value that globally maximizes f. If $\mathbf{x}^* \in D - D^*$, then $\mathbf{x}^c$ lies on the boundary of D^*.*

PROOF. Let $\mathbf{x}^c$ lie in the interior of D^*. Then, according to Theorem 2, all the points that lie on the line segment connecting $\mathbf{x}^c$ and $\mathbf{x}^*$ will have higher function values than $\mathbf{x}^c$. Since $\mathbf{x}^c$ lies inside D^* and $\mathbf{x}^*$ lies outside D^*, this line segment intersects the boundary of the convex set D^* in a single point v. Therefore, $f(\mathbf{v}) \geq f(\mathbf{x}^c)$ and $\mathbf{x}^c$ cannot be the optimum within D^*. □

The previous theorems give rise to the following strategy. We can use a two-dimensional numerical optimization algorithm to maximize $L(\lambda_1, \lambda_2)$. If the optimum lies inside D^*, we have located λ^c. Otherwise, if the procedure converges to an optimum outside D^*, we can search along the boundary of D^* in order to locate λ^c.

The search along the boundary can be decomposed to three searches along the three sides of D^* (Figure 1). Each side can be embedded in a one-dimensional space, therefore we can maximize along each side using a one-dimensional optimization procedure.

Furthermore, depending on the location of λ^*, as was identified by the two-dimensional optimization algorithm, we only need to

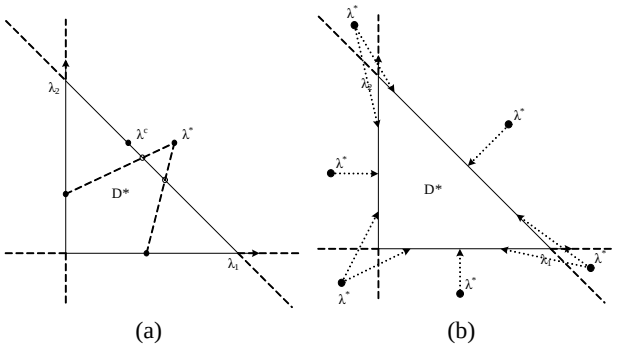


Figure 2: The constrained search space D^*.

search one or two at most sides of D^*. This is demonstrated in Figure 2(a) by means of an example. Due to Theorem 2, for any point on the perpendicular sides of D^*, there is a point in the hypotenuse that evaluates $L(\lambda_1, \lambda_2)$ to a higher value. Based on this observation, we can partition the plane around D^* into six areas and depending on the area where λ^* is located, only search the relevant sides of D^* (Figure 2(b)).

4.3.2 Unbounded likelihood function

Due to the nature of our objective function $L(\lambda_1, \lambda_2)$, we can identify whether the function is unbounded or not only by inspecting the values of p_{i2}, p_{i1} and p_{i0} .

Consider a single term in the sum of logarithms, $\log(\lambda_2 p_{i2} + \lambda_1 p_{i1} + p_{i0}) = \log(a_i)$, where $a_i = \lambda_2 p_{i2} + \lambda_1 p_{i1} + p_{i0}$. If $p_{i2} > 0$ then we can increase the value of λ_2 as much as we want without worrying about a_i becoming negative. Also notice that as $\lambda_2 \to +\infty$, $log(a_i) \to +\infty$, therefore the term becomes unbounded. Similarly, if $p_{i2} < 0$, then the term becomes unbounded as $\lambda_2 \to -\infty$. The same observations hold for the value of p_{i1} and parameter λ_1.

However, $L(\lambda_1, \lambda_2)$ is the sum of many such terms. If there exist for example i, j such that $p_{i2} > 0$ and $p_{j2} < 0$, we can neither increase nor decrease the value of λ_2 towards $+\infty$ or $-\infty$ respectively. As a consequence, neither the i-th, nor the j-th term can become unbounded. But if $\forall i, p_{i2} > 0$, then the objective function increases arbitrarily as λ_2 increases. The following theorem formalizes and proves this intuition.

THEOREM 4. *Let $L(\lambda_1, \lambda_2) = \sum_{i=1}^{l} \log(\lambda_2 p_{i2} + \lambda_1 p_{i1} + p_{i0})$ be the objective function to be optimized and λ_1^c, λ_2^c be the optimal parameter values within D^*. Then,*

- *If $\forall i$, $p_{2i} > 0$ and $p_{1i} > 0$, then $\lambda_1^c + \lambda_2^c = 1$.*
- *If $\forall i$, $p_{2i} > 0$ and $p_{1i} < 0$, then $(\lambda_1^c, \lambda_2^c) = (0, 1)$.*
- *If $\forall i$, $p_{2i} > 0$ and $p_{1i} \lesseqgtr 0$, then $\lambda_1^c + \lambda_2^c = 1$.*
- *If $\forall i$, $p_{2i} < 0$ and $p_{1i} > 0$, then $(\lambda_1^c, \lambda_2^c) = (1, 0)$.*
- *If $\forall i$, $p_{2i} < 0$ and $p_{1i} < 0$, then $(\lambda_1^c, \lambda_2^c) = (0, 0)$.*
- *If $\forall i$, $p_{2i} < 0$ and $p_{1i} \lesseqgtr 0$, then $\lambda_2^c = 0$.*
- *If $\forall i$, $p_{2i} \lesseqgtr 0$ and $p_{1i} > 0$, then $\lambda_1^c + \lambda_2^c = 1$.*
- *If $\forall i$, $p_{2i} \lesseqgtr 0$ and $p_{1i} < 0$, then $\lambda_1^c = 0$.*

PROOF. The proof of the theorem is by simple case analysis and utilizes a slightly different form of Theorem 2. □

The additional constraint $\lambda_1^c + \lambda_2^c = 1$ resulting from utilizing the Theorem, instructs us to search for λ^c along the hypotenuse of D^*, while $\lambda_1^c = 0$ and $\lambda_2^c = 0$, to search along one of the perpendicular sides (Figure 1). This can be performed by means of a one-dimensional optimization technique.

4.3.3 RadING optimization framework

The results derived from the application of Theorems 2, 3 and 4, can be unified into a simple optimization protocol that utilizes 1D and 2D unconstrained optimization techniques as its primitives.

1. Use Theorem 4 to check if $L(\lambda_1, \lambda_2)$ is unbounded and if so perform 1D optimization to locate λ^c along the boundary of D^*.
2. If the likelihood function is bounded, apply a 2D optimization algorithm to identify the global maximum λ^*.
3. If $\lambda^* \notin D^*$, use Theorem 3 to locate λ^c along the boundary of D^*.

As we will experimentally verify in Section 6, the extra cost of optimizing twice when $\lambda^* \notin D^*$, does not offset the benefit of using efficient numerical optimization algorithms.

4.3.4 Newton's method

Although the RadING optimization framework is independent of the specific unconstrained optimization technique that is employed as a primitive, we argue that Newton's method and its variants are ideal candidates for the task.

In brief, the method assumes that the optimization function is quadratic and fits the parameters using derivative information at the current point. It then moves to the point that maximizes the quadratic being fitted. This process converges quadratically fast near the optimum [4, 23, 18].

Newton's method is considered one of the fastest converging optimization methods, yet it can suffer from two limitations [4, 23, 18]. The first is the need to compute the Hessian matrix of the function at each iteration and then invert it.

DEFINITION 2. *The Hessian $H_{n\times n}(f)$ of a twice differentiable function $f(x_1, \ldots, x_n)$, is the matrix of all second order partial derivatives, i.e., $H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$.*

The second limitation is the requirement that the Hessian be a negative semi-definite matrix at the current point, in order for the next Newton's method iteration to increase the value of the objective function.

DEFINITION 3. *A matrix $X_{n\times n}$ is negative semi-definite iff $\forall$ $v_{n\times 1}$, $v^T X v \leq 0$.*

Thus, if the Hessian is not negative semi-definite, it needs to be modified by means of a variety of available time-consuming techniques [23].

To summarize, the use of additional information about the objective function (in the form of its derivatives) to guide the search for the optimum leads to faster convergence but at a potentially high cost per iteration. This trade-off has led to the development and use of the so-called direct search algorithms (e.g., Powell's search, Nelder-Mead) that utilize minimal information about the objective function but demonstrate slower convergence rates [22].

However, none of the two limitations that we discussed pose a problem for our objective function. The Hessian that needs to be computed and inverted is only a 2×2 matrix and is guaranteed to be negative semi-definite due to the concavity of $L(\lambda_1, \lambda_2)$ [4]. Therefore, in our context the cost of a Newton's method iteration is minimal, justifying its use over alternatives that converge at a slower pace.

4.4 Incremental Maintenance

When users continuously annotate resources with new assignments, the efficient *maintenance* of the interpolation parameters is critical. After computing the interpolation parameters from scratch, their values should be updated when the number of new assignments attached to the resource exceeds a threshold. The renewed values will reflect the updated information provided by the new assignments.

The maintenance of the interpolation parameters is in principle the same procedure as optimizing them from scratch. The difference is that we can utilize the parameter values learned previously and use them as the starting values of the optimization algorithms, instead of a random starting point as in the case of optimizing from scratch. Given the stability of the user tagging activity (Section 3), the updated optimal parameter values are not expected to deviate much from their previous values. Hence, initiating the optimization from a point close to the optimum will accelerate convergence. As we will experimentally demonstrate in Section 6, the proposed optimization framework is extremely efficient in this critical task, since it can leverage the Newton's method which converges quadratically fast near the optimum [4, 23, 18].

4.5 Generalization for N-gram Models

Even though we demonstrated the entire process using bigrams for simplicity of exposition, the form and properties of the likelihood function and its domain carry over to higher order n-gram models. Namely, in the case of interpolated n-gram models, the likelihood function that needs to be maximized is

$$L(\lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{l} \log(\lambda_n p_{in} + \cdots + \lambda_1 p_{i1} + p_{i0})$$

where $p_{ik} = \hat{p}(t_{in}|t_{i(n-k+1)}, \ldots, t_{i(n-1)}) - p_{bg}(t_{in})$, i.e., p_{ik} is the modified maximum likelihood estimate for the k-gram of tag t_{in} (Section 4.1). The constrained optimization domain D_n^* is defined by inequalities $\lambda_1 \geq 0, \ldots, \lambda_n \geq 0, \lambda_1 + \cdots + \lambda_n \leq 1$. In other words, rather than being a triangle, domain D_n^* is an n-dimensional polytope. The polytope is depicted for $n = 3$ (interpolated trigram model) in Figure 3.

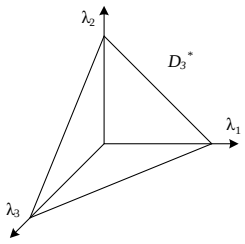


Figure 3: Constrained search space D_3^*.

In the general case, the EM algorithm is extended by simply using one auxiliary variable per interpolation parameter and appropriately modifying its two steps.

The generalization of the RadING optimization framework is also straightforward and intuitive. Domain D_n^* is a n-dimensional polytope with $(n-1)$-dimensional facets. The domain defined by each of the $(n-1)$-dimensional faces is essentially equal to domain D_{n-1}^* of an equivalent $(n-1)$-dimensional problem. By utilizing Theorems 2, 3 and the relevant extension of Theorem 4, the *RadING* optimization framework now employs n-dimensional and $(n-1)$-dimensional versions of the optimization algorithm. Since the $(n-1)$-dimensional optimization is constrained along a facet with domain D_{n-1}^*, it is recursively defined.

The following two theorems demonstrate how to handle the cases of an unbounded likelihood function (Theorem 5) and a likelihood function whose optimum λ_* lies outside domain D_n^* (Theorem 6).

THEOREM 5. *Let $L(\lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{l} \log(\lambda_n p_{in} + \cdots + \lambda_1 p_{i1} + p_{i0})$ be the objective function to be optimized and $\lambda_1^c, \ldots, \lambda_n^c$ be the optimal parameter values within D^*. Then,*

- *For all k, such that $\forall i$, $p_{ki} < 0$, we have $\lambda_k^c = 0$.*
- *If there exists k, such that $\forall i$, $p_{ki} > 0$, we can have that $\lambda_1^c + \cdots + \lambda_n^c = 1$.*

For example, if by inspecting constants p_{ji} we identify that $\lambda_1^c = 0$ and $\lambda_1^c + \cdots + \lambda_n^c = 1$, we can set $\lambda_1 = 0$ and $\lambda_2 = 1 - \lambda_3 - \cdots - \lambda_n$. Then, we need to optimize for variables $\lambda_3, \cdots, \lambda_n$, with constraints, $\lambda_3, \ldots, \lambda_n \geq 0$ and $\lambda_3 + \cdots + \lambda_n \leq 1$, i.e., we need to address an $(n-2)$-dimensional optimization problem in domain D_{n-2}^*.

THEOREM 6. *Let $L(\lambda_1, \ldots, \lambda_n) = \sum_{i=1}^{l} \log(\lambda_n p_{in} + \cdots + \lambda_1 p_{i1} + p_{i0})$ be the objective function to be optimized, $\lambda_1^c, \ldots, \lambda_n^c$ be the optimal parameter values within D_n^* and $\lambda_1^*, \ldots, \lambda_n^*$ the optimal parameter values in $\mathcal{R}^n$. Then,*

- *If $\lambda_i^* < 0$, then facet (boundary) $\lambda_i = 0$ should be checked in order to locate λ^c.*
- *If $\lambda_1^* + \cdots + \lambda_n^* > 1$, then facet (boundary) $\lambda_1 + \cdots + \lambda_n = 1$ should be checked in order to locate λ^c.*

For instance, if we identify that $\lambda_1^* < 0$ and $\lambda_2^* < 0$, then the likelihood function's optimum within D_n^* must lie on either $(n-1)$-dimensional facet $\lambda_1 = 0$ or $(n-1)$-dimensional facet $\lambda_2 = 0$. It is easy to verify that optimizing on these two facets is equivalent to solving a $(n-1)$-dimensional version of the optimization problem on a domain equivalent to D_{n-1}^*.

An extremely desirable property of Newton's method, which can be employed by the RadING framework, is that the number of iterations till convergence remains almost constant, independently of the dimensionality of the problem [4]. The only considerable additional overhead is the cost of inverting a larger Hessian. In practice, n-gram models with $n > 5$ offer no additional performance advantages [15] and given the short length of assignments in our application, the use of a model more complicated than the trigram is hard to justify. Therefore, each iteration should involve the inversion of an at most 5×5 matrix instead of a 2×2 matrix, which is highly scalable.

5. SEARCHING

We now have all the machinery in place for ranking a collection of tagged resources. The first step required is to train a bigram model for each resource, which involves the bigram and unigram probability computation and the optimization of the interpolation parameters. At query time we can compute the probability of the query keyword sequence being "generated" by each resource's bigram model. More precisely, the score of each resource R, given a query $Q = q_1, \ldots, q_k$, is

$$p_R(q_1, \ldots, q_k) = \prod_{j=1}^{k} p(q_j|q_{j-1})$$

where $p(q_j|q_{j-1}) = \lambda_2\hat{p}(q_j|q_{j-1}) + \lambda_1\hat{p}(q_j) + \lambda_0 p_{bg}(q_j)$ is the interpolated bigram probability. Since the scoring function used is monotone (a simple multiplication of terms), the Threshold Algorithm (TA) [24] can be employed for computing the top-k ranking resources.

The whole process is better illustrated with a simple example (Figure 4). The example uses *non*-interpolated bigram models to ease exposition, but its extension to the interpolated variant is simple. Consider a collection of four resources $R_1, \ldots, R_4$ that is only annotated with two tags, t_1 and t_2. In order to stress the importance of a tag appearing in the first position of an assignment, we introduce a "start of assignment" tag, denoted by $\langle s \rangle$[1]. Then the only bigrams that can potentially appear in the assignments of this collection are $(t_1|\langle s \rangle)$, $(t_2|\langle s \rangle)$, $(t_1|t_2)$ and $(t_2|t_1)$.

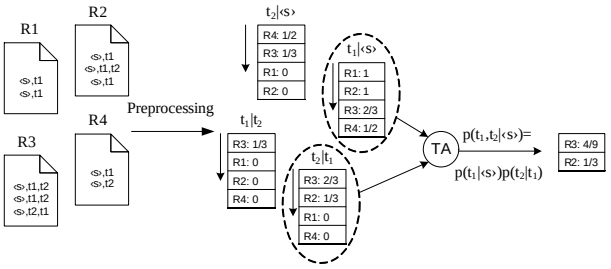


Figure 4: Query evaluation.

The preprocessing step involves going through the collection and calculating the corresponding bigram probabilities for each resource. (If interpolated models are used, the unigram probabilities and interpolation parameters are also computed for each resource.) These probabilities are subsequently stored in four sorted lists, one for each bigram. Then, suppose a query t_1, t_2 arrives at the system. The score of each resource R_i is $p_{R_i}(t_1|\langle s \rangle)p_{R_i}(t_2|t_1)$. The top-$k$ scoring resources can be computed by invoking the TA algorithm and using the lists for bigrams $t_1|\langle s \rangle$ and $t_2|t_1$ (Figure 4).

6. EXPERIMENTAL EVALUATION

In order to evaluate the solutions proposed within the context of the RadING methodology, we used data from our crawl of del.icio.us, one of the most popular social annotation systems, whose shared resources are URLs. The data are comprised of 70,658,851 assignments, posted by 567,539 users and attached to 24,245,248 unique URLs. The average length of each assignment is 2.77, while their standard deviation and median are 2.70 and 2 respectively. An interesting observation is that out of all the URLs in our sample, approximately 19M of them have only been tagged once. This is not an issue with our sample, but rather a characteristic of del.icio.us [11].

6.1 Optimization Efficiency

We compared the performance of the RadING optimization framework (Section 4.3) against the EM algorithm (Section 4.2) in two tasks: optimizing bigram models from scratch and incrementally updating their interpolation parameters. The RadING optimization framework employed Newton's method. The convergence of the Newton and EM algorithms was declared when an iteration failed to improve the likelihood function by more than 10^{-9}.

In our first experiment, we used both algorithms to optimize from scratch the interpolation parameters of every URL in our data set, associated with 10 or more assignments. One fifth of all assignments were placed in the held-out data set and used in the optimization procedure. We discuss techniques that are potentially more appropriate for setting the interpolation parameters of lightly-tagged resources, or resources that have been tagged only once, in Section 6.3.

Figure 5 depicts the total time required by both optimization techniques, to train interpolated n-gram models for $n = 2$ (bigram model) to $n = 5$ (fivegram model). As it is evident, the proposed RadING optimization framework is approximately 4-5 times faster than the EM algorithm. The speed-up can be attributed both to the unique ability of the RadING framework to utilize the efficient Newton's method, as well as its ability to rapidly identify unboundedness and reduce the dimensionality of the optimization procedure.

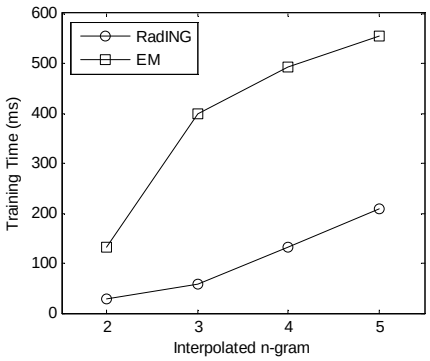
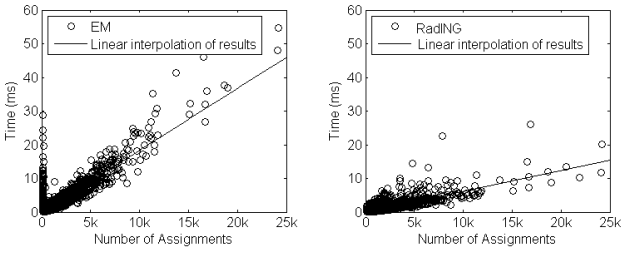


Figure 5: Total training time.

Figures 6(a) and 6(b) depict the time required by both techniques to optimize a single resource for $n = 2$ (interpolated bigram model), with respect to the number of its assignments (NR denotes the RadING optimization framework which employs the Newton-Raphson algorithm). Both methods scale linearly, but as it is evident, the introduced optimization framework is about 4 times faster than the EM algorithm. This guarantees the applicability of the proposed solution in efficiently optimizing resources with any number of assignments.



(a) (b)

Figure 6: Time vs Number of Assignments.

In order to simulate a setting where assignments are continuously added to the resources and evaluate the incremental maintenance efficiency of the algorithms (Section 4.4), we designed the following experiment. In this experiment only resources with more than 200 assignments were used. The assignments of each resource were sorted in chronological order and the first 100 assignments

[1] An "end of assignment" tag can also be introduced.

were used for initially setting the parameters. Then we triggered a re-optimization every 50 new assignments until all the assignments of the resource were exhausted. We measured the total optimization and subsequent re-optimization time for each resource.

Figure 7 presents the total time that was required by each method to incrementally maintain the resources described above, and for interpolated n-gram models of varying sophistication. The RadING optimization framework offers a large performance benefit in the case of bigram and trigram models. As was discussed in Section 4.4, this can be attributed to the quadratic convergence of Newton's method near the optimum. This benefit diminishes for higher order n-gram models. For higher order n-grams, it is much more common for the optimum to lie on a lower dimensional facet of the constrained optimization region D_n^*. Hence, multiple lower dimensional facets must be checked for the optimum in order to guarantee correctness, even though we initiate our search close to it.

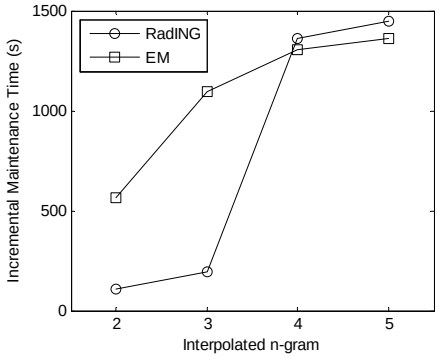


Figure 7: Total incremental maintenance time.

6.2 Ranking Effectiveness

We performed a large scale experiment to evaluate the ranking effectiveness of the RadING ranking strategy. We evaluated the performance of interpolated n-grams – employed by the RadING methodology – of varying complexity, and compared against a significant number of competing alternatives. More specifically, we also considered plain (non-interpolated) n-gram models, in order to validate the benefits offered by interpolation, and two adaptations of the widely used tf/idf ranking [19]. As we subsequently demonstrate, while tf/idf based approaches are extremely powerful and constitute the most appropriate alternative in many contexts, they fail to capture the significant and elaborate organization of tags into assignments, that the proposed solution was explicitly designed to model.

The first method, denoted by Tf/Idf, concatenates the assignments of a resource into a "document" and performs the ranking based on the tf/idf similarity of the query and these "documents". In the second variation, denoted by Tf/Idf+, we compute the tf/idf similarity of the query to each individual assignment and rank the resources based on the *average* similarity of the query to their corresponding assignments.

The lack of a publicly available and widely-accepted test collection, comprised of both annotated resources and relevant queries (in the spirit of the TREC [1] collections), renders the comparison of the different approaches particularly challenging. Hence, ranking effectiveness was evaluated by asking impartial judges to decide upon the relevance of the top results returned by the various methods in response to keyword queries and measuring the precision achieved by each method.

The judges were contacted through the Amazon Mechanical Turk service[2]. The Mechanical Turk service is an on-line "marketplace for work", bringing together users that need information gathering/processing tasks performed and users willing to perform such tasks in exchange for monetary compensation. The workers remain anonymous but have every incentive to perform to the best of their ability since the employer retains the right to review and potentially reject poor work, thus penalizing the worker's reputation.

More specifically, we designed and executed the following experiment. For each keyword query Q that we tested, we retrieved the top-10 results produced by the various alternatives. The results consisted of URLs pointing to web pages. Let $\mathcal{R}_A(Q)$ be the set of the top-10 results, with respect to a single query Q, produced by ranking method A. The URLs comprising the union $\mathcal{R}(Q) = \bigcup_A \mathcal{R}_A(Q)$ of the top-10 URLs from all methods considered were shuffled and presented to 10 impartial, anonymous and unknown to us judges (Mechanical Turk workers).

Every one of the judges (workers) was asked to evaluate whether the content of each web page $p \in \mathcal{R}(Q)$ was relevant or not (as a response to the corresponding keyword query Q). No additional information was given with respect to which method was used to retrieve p or its ranking position in the corresponding top-10 ranking. This was done to avoid biasing the judges in favor of a certain technique or high-ranking URLs. Furthermore, no judge's decision was rejected by us. We compensated for any potential judge misconduct by employing a large number of independent judges.

Given that the relevance of each result p was evaluated by 10 judges, we computed the relevance $r(p)$ of page p as the fraction of the 10 judges that found the page to be relevant, i.e., if 9 out of 10 judges found p to be relevant, its relevance score was $r(p) = 0.9$. Using the decisions of the judges, we computed the Precision at 10 (Precision@10) [19] performance of the various methods for a number of queries. More formally, for the top-10 result $R_A(Q)$ of method A we computed Precision@10 $= \sum_{p \in R_A(Q)} r(p)$, namely the aggregate relevance of the top-10 results in $R_A(Q)$.

EXAMPLE 3. *For instance, a Precision@10 value of 8.3 for a particular method A and in response to a query Q, implies that approximately 8 of the top-10 results retrieved by method A were found to be relevant. Alternatively, 83 of the 100 relevance judgements (10 judges $\times$ 10 binary decisions each) associated with the top-10 results computed by A were positive.*

We submitted to Mechanical Turk for evaluation a large number of diverse queries and present results for a sample consisting of 18 representative ones. Similar results were obtained for the remainder of the submitted queries. Additionally, for our experiment we used a fraction of our original data set comprised of all URLs that were tagged at least 5 times (660k URLs), in order to avoid presenting URLs for which a consensus among the users of del.icio.us had not yet emerged and therefore introducing URLs that were irrelevant exclusively due to noise.

Table 1 presents the Precision@10 performance of 4 different techniques. $I2g$ stands for interpolated bigram, $I3g$ for interpolated trigram, while $2g$ and $3g$ stand for plain bigram and trigram respectively. Two conclusions can drawn from the data.

First, the use of more sophisticated interpolated n-gram than a simple bigram model does not seem to improve ranking effectiveness. The performance of the $I2g$ and $I3g$ methods (both part of the RadING ranking strategy) and the corresponding rankings produced is almost identical. This can be attributed to the relatively

[2]www.mturk.com

Query	$I2g$	$I3g$	$2g$	$3g$
birthday gift ideas	7.4	7.4	3.6 (4)	1.0 (1)
college blog	7.6	7.6	6.6 (10)	7.6 (10)
trigonometric formulas	7.9	7.9	0.9 (1)	0.9 (1)
stock market bubble	8.4	8.4	1.0 (1)	0.0 (1)
sea pictures	6.7	6.3	2.1 (3)	0.9 (1)
free music recording software	7.6	7.6	7.6 (10)	2.6 (3)
insomnia cure	8.4	8.6	1.9 (2)	0.9 (1)
red wine benefits	8.2	8.2	0.0 (0)	0.0 (0)
software draw bubbles	6.0	6.2	0.0 (0)	0.0 (0)
duck recipe	6.3	6.3	0.7 (1)	0.7 (1)
chinese food	8.7	8.7	8.2 (10)	8.5 (10)
recycling tips	9.3	9.3	8.8 (10)	5.3 (7)
water filter	8.5	8.5	9.1 (10)	9.1 (10)
economics tutorial	8.9	8.9	8.0 (10)	8.4 (10)
linear algebra book	9.1	9.1	9.0 (10)	5.5 (6)
f1 technology	7.9	7.9	0.7(1)	0.0 (0)
coffee machine	7.4	7.4	7.7 (10)	6.7 (9)
history books	8.4	8.4	7.9 (10)	7.2 (10)
Average Precision@10	7.93	7.93	4.65	3.63

Table 1: Precision@10 for a representative query sample.

Query	$I2g$ (RadING)	Tf/Idf	Tf/Idf+
birthday gift ideas	7.4	2.8	5.8
college blog	7.6	5.9	4.9
trigonometric formulas	7.9	6.3	5.2
stock market bubble	8.4	5.1	6.1
sea pictures	6.7	4.5	2.1
free music recording software	7.6	6.0	5.5
insomnia cure	8.4	6.5	7.0
red wine benefits	8.2	6.1	6.5
software draw bubbles	6.0	2.1	2.9
duck recipe	6.3	3.4	3.8
chinese food	8.7	7.7	2.2
recycling tips	9.3	8.0	8.2
water filter	8.5	7.1	5.1
economics tutorial	8.9	7.8	6.5
linear algebra book	9.1	7.0	6.4
f1 technology	7.9	7.5	7.6
coffee machine	7.4	7.1	5.9
history books	8.4	8.4	7.3
Average Precision@10	7.93	6.07	5.50
Average Improvement	n/a	+31%	+44%

Table 2: Precision@10 for a representative query sample.

short length of tag sequences assigned by users. The median length of a sequence was only 2.7. Hence, an interpolated bigram model is sufficient for capturing tag co-occurrence patterns in such short tag sequences.

Second, the use of interpolated n-grams ($I2g$, $I3g$) offers substantial benefit over the use of plain n-gram models ($2g$, $3g$). The reason is that most of the URLs in the collection are associated with relatively few assignments, i.e., the training data associated with most URLs is sparse (Section 3.4). This is not an artifact of our data set, but rather a characteristic of social annotation systems in general [11]. We present in parentheses, next to the Precision@10 value of the $2g$ and $3g$ methods, the overall number of URLs that evaluated to non-zero relevance probability and were, hence, retrieved. In most cases, the number is considerably less than 10.

Table 2 contrasts the ranking effectiveness of the RadING ranking strategy with that of the tf/idf based alternatives. RadING utilized interpolated bigram models. As it is evident, the proposed ranking solution is superior to both adaptations of the tf/idf similarity metric. On average, the bigram-based approach achieves a 31% and 44% better Precision@10 score than the Tf/Idf and Tf/Idf+ methods respectively.

The superior ranking effectiveness of RadING can be attributed to the ability of the interpolated bigrams to model and utilize the tag co-occurrence patterns that are observed in the URLs' tagging history. This crucial piece of information cannot be exploited by the tf/idf based approaches. Both Tf/Idf and Tf/Idf+ fail to capture the fact that less-frequent tags can co-occur with high frequency and instead only rewards high tag frequencies, therefore underestimating the relevance of many URLs.

Additionally, the improved ranking effectiveness demonstrated by the RadING solution over the tf/idf based alternatives is statistically significant and is not an artifact of limited experimentation. Consider a query Q and the top-10 results returned by RadING and Tf/Idf solutions. Each set of results is associated with $n = 100$ binary relevance judgements (URL relevant/not relevant). The fraction r of positive judgements is intrinsically linked to the corresponding Precision@10 value: we simply have that Precision@10 $= 10 * r$. Notice that r is (the maximum likelihood estimate of) the probability of a randomly selected judgement being positive. In order to demonstrate that the improvement offered by RadING for query Q is significant, we need to demonstrate that $r_{RadING} > r_{Tf/Idf}$ with high confidence. This would imply that the corresponding Precision@10 scores follow the same relation. Using a binomial statistical test the null hypothesis $r_{RadING} < r_{Tf/Idf}$ can be rejected at the 1% confidence level (p-value) for the first 15 queries presented in Table 2. Hence, we have $r_{RadING} > r_{Tf/Idf}$ with high confidence.

Lastly, Figure 8 presents the average Precision@k scores achieved by the three ranking solutions, on the same query sample and for different values of k. The average Precision@k values have been normalized using the value of k, i.e., we present the corresponding Precision@k/k values. As it is evident, the RadING ranking strategy consistently outperforms the tf/idf based approaches for all values of k. Notice also the extremely high Precision@1 score achieved by RadING: the top-1 query result returned by the technique was found to be highly relevant on all occasions. Additionally, although the Precision@k performance of RadING is consistently high, it is better for smaller values of k. This trend implies that RadING produces an intuitive and desirable ranking, with more relevant URLs appearing higher in the top-10 result. This trend is not observed for the tf/idf based approaches.

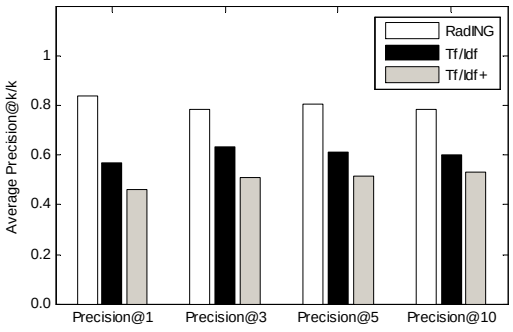


Figure 8: Average Precision@k.

6.3 Discussion

The introduced RadING methodology offers an additional and substantial benefit, besides its superior ranking effectiveness and scalability, namely the ability to incorporate a number of features affecting ranking in a *principled* manner. This can be performed by

employing non-uniform resource prior probabilities (Section 3.1) and modifying, if needed, the interpolation parameters for certain resources (Section 3.5). In what follows we do not offer concrete solutions, but rather hint at possibilities for providing a richer searching experience in a collaborative tagging context.

As an example, the resource priors can be used to bias the final ranking towards resources that are popular. Another option would be biasing the ranking in favor of more recent resources. This would offer the possibility for such resources to be more easily discovered and therefore allow them to build momentum, provided that users find them interesting enough to tag them. Furthermore, the non-uniform resource priors can also be used to offer a personalized searching experience. This much-desired property could be provided in a practical and scalable manner by clustering the users and employing different priors for different user clusters.

Lastly, a significant number of resources in a collection can be expected to be tagged only once or a handful of times at most. Whether the assignments associated with those lightly-tagged resources can be trusted is debatable. Regardless, the linearly interpolated models that we utilize offer a principled solution for addressing this issue. For such resources, *if desired*, we can skip the interpolation parameter optimization process and instead set those parameters manually. As we discussed in Section 3.5, we can use parameters λ_2, λ_1 and λ_0 to express our confidence towards the bigram, unigram and background tag probabilities of a resource's history respectively. For example, if we lack confidence on the assignments of lightly-tagged resources, we can set parameters λ_1 and λ_2 to low values. This type of reasoning should also be applied to resources that have been tagged only once. The ability to handle lightly-tagged resources in such a flexible and principled manner is unique to our approach.

Another parameter estimation approach that is appropriate for lightly-tagged resources (but can also be applied to all resources) is *cross-validation*. The assignments associated with a resource are separated into b disjoint "buckets". Then, b rounds of parameter estimation are performed: each bucket is used as the held-out data set, while the remaining $b-1$ buckets comprise the training data set. The final interpolation parameter values are calculated by averaging the values computed in the b optimization rounds.

7. CONCLUSIONS

In this paper we presented a principled, efficient and effective ranking methodology that utilizes statistical language modeling tools, as motivated by our understanding of the collaborative tagging process. Training of the language models was performed by means of a novel optimization framework that outperforms by a large margin the optimization techniques employed currently. The validity of our claims was verified using a massive data set extracted from a popular social annotation system.

Acknowledgements: The work of Gautam Das was supported in part by the US National Science Foundation under grants 0845644 and 0812601, unrestricted gifts from Microsoft Research and Nokia Research, and start-up funds from the University of Texas at Arlington.

8. REFERENCES

[1] Text retrieval conference (trec). http://trec.nist.gov/.

[2] S. Amer-Yahia, M. Benedikt, L. V. Lakshmanan, and J. Stoyanovich. Efficient network-aware search in collaborative tagging sites. In *VLDB*, 2008.

[3] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW*, pages 501–510, 2007.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[6] C. Cattuto, V. Loreto, and L. Pietronero. From the Cover: Semiotic dynamics and collaborative tagging. *PNAS*, 104(5):1461–1464, 2007.

[7] S. F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.

[8] P. A. Chirita, S. Costache, W. Nejdl, and S. Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW*, pages 845–854, 2007.

[9] S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[10] H. Halpin, V. Robu, and H. Shepherd. The complex dynamics of collaborative tagging. In *WWW*, pages 211–220, 2007.

[11] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM*, pages 195–206, 2008.

[12] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC*, pages 411–426, 2006.

[13] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Trend detection in folksonomies. In *SAMT*, pages 56–70, 2006.

[14] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[15] T. Joshua. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, 2001.

[16] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. MIT Press.

[17] R. Li, S. Bao, Y. Yu, B. Fei, and Z. Su. Towards effective browsing of large scale social annotations. In *WWW*, pages 943–952, 2007.

[18] D. Luenberger. *Linear and Nonlinear Programming*. Kluwer Academic Publishers, 2003.

[19] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[21] D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *SIGIR*, pages 214–221, 1999.

[22] O. Nelles. *Nonlinear System Identification*. Springer, 2000.

[23] J. Nocedal and S. Wright. *Numerical Optimization, 2nd Edition*. Springer, 2006.

[24] R. Fagin and A. Lotem and M. Naor. Optimal Aggregation Algorithms For Middleware. *PODS*, June 2001.

[25] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM*, pages 316–321, 1999.

[26] L. Xu and M. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.