Retrieving Similar Discussion Forum Threads: A Structure Based Approach

Amit Singh Deepak P Dinesh Raghu
IBM Research - India, Bangalore, India
{amitkumarsingh,deepak.s.p,dinraghu}@in.ibm.com

ABSTRACT

Online forums are becoming a popular way of finding useful information on the web. Search over forums for existing discussion threads so far is limited to keyword-based search due to the minimal effort required on part of the users. However, it is often not possible to capture all the relevant context in a complex query using a small number of keywords. Example-based search that retrieves similar discussion threads given one exemplary thread is an alternate approach that can help the user provide richer context and vastly improve forum search results. In this paper, we address the problem of finding similar threads to a given thread. Towards this, we propose a novel methodology to estimate similarity between discussion threads. Our method exploits the thread structure to decompose threads in to set of weighted overlapping components. It then estimates pairwise thread similarities by quantifying how well the information in the threads are mutually contained within each other using lexical similarities between their underlying components. We compare our proposed methods on real datasets against state-of-the-art thread retrieval mechanisms wherein we illustrate that our techniques outperform others by large margins on popular retrieval evaluation measures such as NDCG, MAP, Precision@k and MRR. In particular, consistent improvements of up to 10% are observed on all evaluation measures.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval Models*

Keywords

Forums, Threads, Retrieval, Similarity

1. INTRODUCTION

Online forums are fast becoming a very popular data source on the web. They are an easy way to publish knowledge since composing a post in a forum is much easier than composing a web page. Further, forums are interactive spaces where many people contribute and hence are *live* (as against web pages that are often static); making them more attractive to contribute to. Forums are monitored by many people making spurious information easy to detect and point out; hence, it may be expected that contributors would exercise some caution to ensure that the information that they post is correct and useful. It may also be observed that forums mostly contain subjective knowledge (e.g., opinions) and hence are often complimentary to other knowledge sources such as Wikipedia[1]. Despite such uniqueness, there have been only a few forum search engines[2][3] and searching forum threads is still only an emerging topic [28, 10, 12].

In this paper, we consider the problem of *finding similar threads to a given thread*. Threads in online discussion forms are collections of posts that are posted in response to a common starting post. Finding similar data objects in other popular data sources such as text collections [13], web documents [8], images [34], websites[4], social network profiles [15], linked entities [19] and relational data [9] have been a subject of much research. *However, finding similar threads in online forums, to the best of our knowledge, has not received enough attention.*

Applications: Much like finding similar objects in other knowledge sources, finding similar threads has many potential applications. For example, a user browsing a thread in a discussion forum could be provided links to *similar threads* for a more detailed reading on the same or related topic. Likewise, links to *related threads* (much like grouping of news articles in systems such as *Google News*[5]) could be of use in thread retrieval systems [12]. A notion of similarity among threads would also help developing a clustered interface for search results (e.g., Yippy[6]) in thread retrieval. A *knowledge author* who seeks to read through discussions forums (e.g., developer forums) and document problems and solutions from them would most likely prefer to read through similar threads together since that would help minimize context switch, by enabling her to cover one type of problems in entirety before moving to the next. Though finding similar threads has many potential applications as outlined above, we focus on the scenario of providing a functionality of *finding similar threads* to a given thread (e.g., the thread the

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

[1]http://en.wikipedia.org
[2]http://www.boardreader.com
[3]http://www.boardtracker.com
[4]http://www.similarsites.com/
[5]http://news.google.com
[6]http://search.yippy.com

user is reading currently) and evaluate the effectiveness of the similarity measure that we propose, using classical Information Retrieval evaluation measures.

User:0 My Beetel 220 modem has suddenly stopped working. The data led has stopped blinking. Can somebody help?	
Thread T1	**Thread T2**
User:1 You could restart the modem and see whether it works. Else, it may be due to some problem in the splitter. Try connecting the line directly to modem	**User:2a** It would most likely be solved by a modem restart.
	User:2b If the above solution doesnt work, try connecting the line directly without splitter.

Table 1: Example Thread Patterns.

Challenges vis-a-vis Document Similarity: Threads in discussion forums, despite being largely composed of textual content, have various properties that distinguish them from regular text documents such as newswire articles. The latter are mostly authored by a single author, whereas discussion forum threads involve many contributors. This makes them less coherent, and more vulnerable to abrupt jumps in topics. Further, discussion forum threads have an inherent structure; this could be visualized as a tree structure where posts form nodes and edges linking posts to their replies. The intuitive method of estimating similarity that considers *each thread as a large document comprising of all the component posts* followed by usage of traditional similarity metrics such as tf.idf cosine would obviously lead to large posts being able to influence similarity more than smaller ones; this is undesirable since the length of a post is more often related to other factors such as author's style than the utility of the post. At the other end of the fragmentation spectrum, is that of *considering each thread as a bag of posts*, with similarity between them being quantified as an aggregate of the pairwise similarities between the posts. Consider two hypothetical threads T1 and T2 spawned out of the same first question in Table 1. Despite being very similar due to the nature of content posted, no pair of posts (one from each thread) have a high lexical similarity. This is due to the content in the single post in T1 getting split across two posts in T2. Such fragmentation is common in discussion forums due to varying expertise levels of content authors and dents the effectiveness of the bag of posts approach. The third solution is that of *selecting highly informative words to form summaries of each thread* and using them as a keyword query in keyword search systems that operate on forums [28, 10, 12]. However, such selection of words is often hard making it difficult to capture the entire context of the query thread. We address the above challenges in developing a methodology for finding similar threads.

Our specific contributions are as follows:

- We propose a novel methodology for estimating pairwise similarities for discussion forum threads that leverages structural information of threads by decomposing threads into weighted overlapping components. For our choice of component types, we show that the similarity computation can be performed efficiently.
- An extensive empirical evaluation that illustrates that our methodology outperforms state-of-the-art baselines.

2. PROBLEM DEFINITION

Consider a discussion forum thread X and a set of threads $\mathcal{Y}$. We would like to develop a method that heuristically estimates the similarity between X and each thread in $\mathcal{Y}$. Given such a method $S(.,.)$, one can rank the threads in $\mathcal{Y}$ in the non-increasing order of similarity with X, such that the following condition is satisfied:

$$\forall y \in \mathcal{Y} - \{Y_1, Y_2, \ldots, Y_{i-1}\}, S(X, Y_i) \geq S(X, y)$$

where Y_i denotes the i^{th} thread when threads in $\mathcal{Y}$ are arranged in the non-increasing order of similarity wrt X. The above condition simply suggests that Y_i has a score at least as much as any of threads not ahead of it in the ranking.

2.1 Thread and Posts

Since we will make many references to threads and posts, we hereby informally outline what we mean by them. We define a post as the smallest unit of communication in online forums that consists of content posted by a user. Each post has associated meta-data such as user ID of the user posting the post, time of posting the post etc. A thread starts off with a post (that we call as the *first post*, whose title is the thread title too), and comprises all posts in reply to the initial posts. There is a recursiveness in the definition in that it includes all posts in reply to posts already in the thread. A thread tree is formed by the posts in the thread as nodes, and with a directed edge from a post to all direct replies to it. In such a thread tree represented in Figure 1(b), B2 is a direct reply to B1 whereas B4 was posted in reply to B2. In most online discussion forums, the thread structure is apparent due to the restrictions imposed (a reply having to be attached a specific post); if not, the thread structures could be identified heuristically [25].

2.2 Thread Components

Threads being composed of posts, an intuitive way to model thread similarity would be to consider a thread as a bag of posts, wherein the scoring function could be posed as an aggregate of the pairwise similarity between posts. However, such a formulation would not be robust to fragmentation of information across posts, as seen in Section 1. In short, we would do better by allowing for collections of posts in a one thread to be mapped to collections of posts in another, while performing similarity calculations. Thread substructures have been exploited for online community search in [28] where four substructures were considered viz., the *entire thread*, *single posts*, *post-reply pairs* and *the dialogue* (a chain of posts from root to a leaf). The results therein suggest that the dialog substructure was not very useful. Based on such conclusions, we also opt to consider *single posts* and *post-reply pairs* as the substructures for consideration. For this choice of substructure types, we will show in a later section, that our similarity computation is very efficient. Though we only consider the above two types of

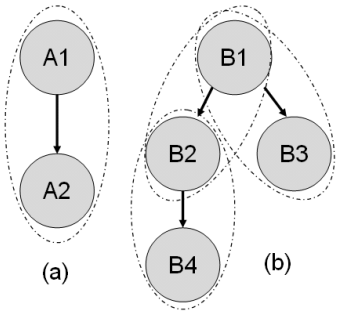

Figure 1: Threads with Post-Reply Substructures.

substructures, the framework of our methodology is generic and can accommodate any kinds of substructures.

For a thread represented by nodes as posts and reply-to relations as edges, in addition to the individual posts, the set of substructures would include the pairs indicated by the ovals as shown in Figure 1. Consider the thread in Example 1(b); the component substructures would then be {$B1$, $B2$, $B3$, $B4$, $[B1, B2]$, $[B1, B3]$, $[B2, B4]$}. It may be noted that such a modeling does not fully solve the information fragmentation issue since it does not allow a set of three posts to match to a single post, even if that happens to be the real nature of fragmentation in a specific case. However, it goes one step forward than matching posts to posts, and could allow for matching the two posts in T2 in Table 1 to map to the single post in T1. We will use post and post-reply substructures (which we refer to, as components, hereon) in estimating similarity between threads in Section 3.

3. OUR APPROACH

We now describe the techniques that we propose for estimating thread similarity; we will first describe a framework and then go on to describe instantiations of the framework that we use in our empirical evaluation. The basic problem is that of estimating the similarity between two threads X and Y. Once we have a means of estimating such thread pairwise similarities, it is easy to build a retrieval engine that finds the top-k similar threads to a given query thread, as seen in Section 2.

3.1 Modeling Thread Similarity

We now outline our concerns in modeling the similarity between threads, $S(X, Y)$. *Firstly*, threads in discussion fora, due to intervention by many people, could steer away from the main topic in the course of time and digress to tangential topics; all posts are typically not equally important to the central topic of discourse. Thus, a methodology for estimating thread similarity would have to weigh posts differentially, based on heuristics such as relevance to the central topic(s) or author reputation. *Secondly*, a thread is a collection of posts, and the number of posts in a thread could vary very widely. For example, a thread in a technical discussion forum could be composed of just 2 posts (e.g., the statement of a problem, followed by a reply with the solution), or could be a long drawn discussion (involving scores of posts) between participants that culminates in finding the solution to the problem posed in the first post. In finding similar threads to a given query thread, one may have to compare threads of widely varying sizes. Due to these properties of *thread posts having to be weighted differentially*, and *threads being composed of widely varying number of posts*, we find it very intuitive to decompose the notion of thread pairwise similarity into two distinct components; one that measures how well the information in X is subsumed in Y (where Y could contain other extra information), and another that measures how well Y is subsumed in X. Towards this, as detailed in Section 2.2, we decompose threads into a set of potentially overlapping components with an associated weight for each, weights indicating the importance of the component to the thread in consideration.

For a pair of threads, X and Y, our estimate of similarity between them (that we denote by $S(X, Y)$) would consider three factors:

- $P_S(X, Y)$, a score that denotes how well the thread X is subsumed by (i.e., contained in) Y.
- $P_S(Y, X)$, an analogous score that estimates how well Y is subsumed by X.
- $Sim(X_{headpost}, Y_{headpost})$, the similarity between the titles and the first posts of the threads.

$X_{headpost}$ denotes the document formed by collating the title and the first post of the thread X. We use several popular text similarity metrics in estimating $Sim(.,.)$.

3.1.1 Estimating Subsumption

In estimating how well X *is being subsumed by* Y, *we roughly try to find the best matching component of* Y *for* **each** *component in* X. In particular, many of Y's components may be left unmatched; this is because while estimating how well X is contained in Y, we would intuitively not want to penalize Y for the extra information that it may contain. To recollect, the components of the thread in Example 1(b) (which we refer to as X) are {$B1$, $B2$, $B3$, $B4$, $[B1, B2]$, $[B1, B3]$, $[B2, B4]$}. We associate a score with each component X_i w.r.t Y:

$$Score_Y(X_i) = max\{Sim(X_i, Y_j) | Y_j \in Y\} \quad (1)$$

where $Sim(.,.)$, the similarity between components, denotes the text similarity and X_i denotes the text document formed by putting together the text from posts within X_i. Essentially, for each component of X, we estimate the maximal similarity with any component in Y.

As can be seen, $B2$ has its representation in 3 of these 7 components of X; it is easy to see that a post would have representation in as many pairs as it has immediate children (if not a root, it is also represented in a pair with its parent). However, we would not like to allow certain posts to influence the $P_S(X, Y)$ score more than others by design. *In particular, we want to choose a set of components of* X *such that each post is only represented once across the set.*

Figure 2 represents a graph with the components of the thread X as nodes; we call it the *component graph*. We induce an edge between components that share a node. The scores (computed as described above) are shown against the nodes. Now, our problem reduces to finding the *maximum weight independent set*[7] (MWIS)[27] of such a graph. The non-adjacency property of the MWIS set ensures that no post is counted twice, due to the nature of inducing edges.

To be fair in comparison between components that contain single posts and those that contain two, we use an intuitive weighting scheme where the score of each component is

[7]http://en.wikipedia.org/wiki/Maximal_independent_set

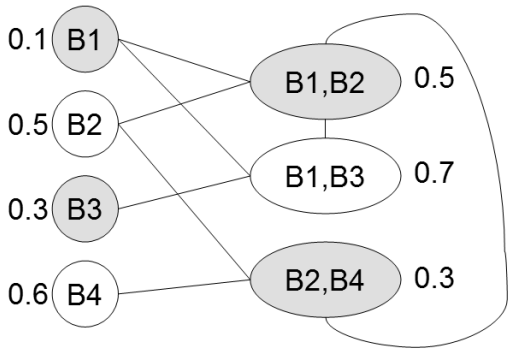

Figure 2: Component Graph.

weighed as much as the number of posts it contains. Thus, the MIS shown in white in Figure 2 would have a score of $0.5 + 0.6 + 0.7 * 2 = 2.5$; for the graph in our example, the white nodes denote the MWIS as well.

THEOREM 3.1. *For the component graph of a thread, the MWIS can be computed in polynomial time and always covers each post exactly once.*

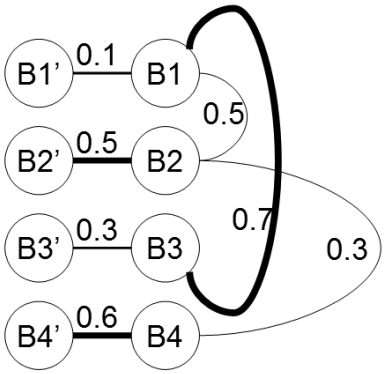

Figure 3: Transformed Graph.

Proof: In Figure 2, we used nodes as components for the sake of simplicity. Now, we illustrate a new graph where the nodes in the component graph (i.e., components of the thread) are translated to edges; this is given in Figure 3. In this graph, we use two nodes to represent each post p, one node to stand for a post (that we denote as p itself, at the risk of overloading notation), and another one that we use as a pseudo-node, p' to avoid self-edges (since we are about to talk about matching and most literature on matching is silent about graphs that contain self-edges). A pair component $\{p_1, p_2\}$ is represented by the edge $p_1 - p_2$ whereas the edge $p_1 - p_1'$ represents the singleton component $\{p_1\}$. Now, the MWIS problem for component graphs corresponds to the problem of finding the Maximum Weighted Matching (MWM) in the transformed graphs like in Figure 3. This transformation is possible since we have modeled the component graph as one where each component can *at most* contain two posts, thanks to our choice of components that can at most span two posts. The MWM problem is known to be solvable in polynomial time [14]; this proves the first part of the theorem.

We now informally clarify as to why the maximum matching of the new graph would cover every post exactly once. A post is deemed to be covered in a matching if the matching contains an edge that is incident of a node *or* the pseudo-node of the post. Matching, by definition, would not allow for covering a node (either of p or p') multiple times. Our informal proof as to why every post is covered works by contradiction. Let us assume that an MWM does not cover post p. This implies that neither the node p, nor p' (the pseudo-node), is included in it. Hence, we can simply add the edge connecting p and p' to the matching without violating the property of it being a matching (no node incident on multiple edges). Since we assume (like in most cases in literature) that edge weights are non-negative, this can only improve the weight of the matching, and also includes more nodes. This negates the initial premise that the matching that excluded p was the MWM and completes the proof. ■

Our formulation of $P_S(X, Y)$ is now the following:

$$P_S(X,Y) = \frac{\sum_{X_i \in MWIS(X,Y)} w_X(X_i) \times Score_Y(X_i)}{\#\ posts\ in\ X}$$

where $MWIS(X, Y)$ denotes the MWIS for the component graph constructed from thread X w.r.t thread Y and $Score_Y(X_i)$ is computed as in Equation 1. Normalizing the sum by the number of posts in X is kosher since each post is represented exactly once in one of the terms in the numerator (the one corresponding to the component in $MWIS(X, Y)$ that contains the post). The only term that has not yet been introduced yet is $w_X(X_i)$; we delve deeper into this weighting parameter in the following subsection.

Complexity of finding MWIS(X,Y): As we have already illustrated, the problem of finding MWIS(X,Y) reduces to finding the Maximum Weighted Matching in our case. MWM is solvable in $\mathcal{O}(m\sqrt{n})$ where m denotes the number of edges, and n denotes the number of vertices in the transformed graph. However, very fast approximations have been proposed recently [11], thus making the computational expense not of much concern.

3.1.2 *Differentially Weighting Components*

Consider a thread about a technical issue (e.g., *bluetooth not connecting in symbian*) and two posts from it, one of which talks about the core issue (e.g., *bluetooth fix for symbian*), whereas the other talks about a tangential issue (e.g., *comparsion of iphone with android and symbian*); we would intuitively want to weigh the former higher since it is *more central* to a topic. This is all the more important for generic posts (e.g., *please help me solve the problem*) that would have to be weighed lower to limit their influence. To enable this sort of weighting, we represent post as a mixture of topics.

$$w_X(X_i) = \sum_{topic \in X} P(X_i|topic) \times P(topic|X)$$

Now, $P(X_i|topic)$ can be interpreted as the likelihood of the topic model generating component X_i under the "bag-of-words" assumption. Similarly, $P(topic|X)$ measures the importance of the topic to the thread. We estimate these weights over a specified number of topics (that are maximally represented in the thread) with the number of topics taken as a system parameter.

3.1.3 Putting it all together

Having described our formulation of $P_S(X,Y)$ (and thus, the analogous $P_S(Y,X)$), we now describe how we combine the three factors outlined in Section 3.1. Since we are interested in finding similar threads, similarity being quantified by means of containing similar content, we would intuitively prefer finding such Ys (in response to a query thread X) which have to high values of $P_S(X,Y)$ and $P_S(Y,X)$. Either of these being low is a strong indication of the asymmetry, and thus, we choose to give the lower of the two values more preference by using the lowest of the three means (harmonic, geometric and arithmetic) to aggregate these. Our measure of similarity then takes the following form:

$$S(X,Y) = \lambda\ harmonic\_mean(P_S(X,Y), P_S(Y,X)) + (1-\lambda)\ Sim(X_{headpost}, Y_{headpost}) \quad (2)$$

λ balances the weight given to the subsumption term against that given to the *headpost* similarity. It may be noted that though we model similarities using feature sets like in [33], we stick to a symmetric notion of similarity as is popular for similarities between textual entities.

3.2 Approach Instantiations

In modeling thread similarities, we have outlined how we exploit substructures and differentially weigh them using topics. To show the incremental utility of substructures and topics, we consider four different techniques:

- **SUB:** This denotes the subsumption based thread similarity approach where we do not consider post-pairs as components (thus, the only components would be individual posts) and weigh them all equally.
- **SUB-TOP:** While considering only individual posts as components, we now harness the topic clusters to weigh each component (i.e., post) differentially.
- **SUB-STR:** This improves upon the SUB approach by considering post-pairs as components (in addition to individual posts); however, we weigh all components equally (do not use topics).
- **SUB-STR-TOP:** This denotes the full-fledged approach where the set of components is made up of both individual posts and post-pairs and components are weighed using the topics.

4. BASELINE METHODS

Since the problem that we are addressing in this paper has not been subject of prior research (Ref. Section 6), we outline a series of intuitive methods to estimate thread similarities. We outline the techniques under two heads; one in which the query thread summaries or extracts are used as queries to retrieve threads harnessing state-of-the-art thread retrieval models, and another in which the threads are directly compared against each other.

4.1 Using Thread Retrieval Models

4.1.1 Pseudo Cluster Selection (PCS)

The title and the first post of a thread are important since they are very influential in setting the agenda of the thread; they have been found useful for thread retrieval [2]. We put together the title and the first post of the query thread as a query and employ the Pseudo Cluster Selection (PCS) model for thread retrieval (proposed in [28]) with the system parameter k set to 5.

4.1.2 Summarization Techniques

Summarization techniques [22] process a document and produce a concise version encompassing the most important concepts in the document. We use those to summarize the query thread to a text snippet; the snippet is then passed to the PCS thread retrieval model to retrieve similar threads.

We use two popular summarization techniques. MMR [6] focusses on increased diversity (and thus, reduced redundancy) while choosing sentences from the document to add to the summary. Submodular [21] is a recently proposed technique that models the summarization problem as that of maximizing a submodular function under a budget constraint. We denote these approaches as $Q.MMR$ or $Q.SM$ depending on what summarization technique is used (Q indicates that the query thread alone is summarized).

4.2 Comparing Threads

Here, we outline intuitive techniques (based on current trends in text processing) to find similar threads.

4.2.1 Bag Of Words (BOW)

The most widely used model for representing documents in Information Retrieval is that of considering a document as a collection (i.e., a bag) of the component words [1]. In this baseline, we consider a thread as a large document (under the large document model [12]) formed by putting together the text from all the component posts. Now, the similarities between threads can be estimated as the similarities between such documents as assessed using one of the popular text similarity measures that use the bag of words (BOW) model. We will experiment with measures such as cosine similarity of term frequency vectors, cosine similarity of tf.idf vectors [16] and Jaccard similarity co-efficient[17].

$$BOW(X,Y) = Sim(X,Y) \quad (3)$$

where $Sim(.,.)$ stands for the text similarity between the document representations of the threads.

4.2.2 Bag of Posts (BOP)

In this method, we consider each thread as a bag of the component posts [12]. Now, to compute thread pairwise similarity, we aggregate the postwise BOW similarities (between a pair of posts, one from each thread) using the arithmetic mean to arrive at a single measure of thread pair similarity.

$$BOP(X,Y) = mean\{BOW(t_x,t_y)|t_x \in X, t_y \in Y\} \quad (4)$$

Such aggregation of pair-wise similarities using the arithmetic mean is used in methods such as UPGMA and (Average-Link) Hierarchical Agglomerative Clustering [18].

4.2.3 Title and First Post (HeadPost)

Since the first post of the thread initiates the thread, it is intuitive for the thread initiator to provide a representative title for it to attract relevant attention from the forum. In many threads like those in support forums, the first post conveys the problem and subsequent posts typically contain

solutions or (requests for) more elaborate descriptions of the problem. The contents of the first post and its title can be thought of *setting the agenda* of the thread, and thus being more representative of the thread than other posts [2]. Here, we estimate the similarity between threads as a function of the similarity between their titles and the first posts.

$$HeadPost(X,Y) = Sim(X_{headpost}, Y_{headpost}) \quad (5)$$

where $X_{headpost}$ and $Y_{headpost}$ denote the text documents formed by putting together the title and the first post of the thread X and Y respectively.

4.2.4 BOP+HeadPost

This denotes a linear combination of BOP and HeadPost.

$$BOP + HeadPost(X,Y) = \alpha BOP(X,Y) + (1-\alpha) HeadPost(X,Y) \quad (6)$$

We set $\alpha = 0.45$ since that was empirically found to be the choice yielding the best performance.

4.2.5 Maximum Similarity between Posts (MAX)

The BOP method (Section 4.2.2) may not be very robust to noise. Consider comparing a long thread with a thread containing just two posts, one of which is a noisy post; the noisy post would then affect half of the values on which the arithmetic mean is computed, and thus reduce the thread pairwise similarity considerably. Such considerations lead us to the MAX method that uses the maximum BOW similarity between a pair of posts to estimate similarity.

$$MAX(X,Y) = max\{BOW(t_x, t_y) | t_x \in X, t_y \in Y\} \quad (7)$$

This method is analogous to the similarity computation used in the popular Single-Link Hierarchical Agglomerative Clustering method [18].

4.2.6 Top-k Similarities (Top-k)

Instead of considering just the max similarity (like we do in MAX above), we now consider the mean of the top-k pairwise similarities.

$$Top\text{-}k(X,Y) = mean(top\text{-}k\{BOW(t_x, t_y) | t_x \in X, t_y \in Y\}) \quad (8)$$

where the top-k(.) method, when applied to a set of scores, returns the subset of top-k values.

4.2.7 Central Post Similarity (Central)

In this method, we use a UPGMC [23] style aggregation of pairwise similarities. Unlike UPGMC that computes the similarity between centroids of collections (i.e., threads), we choose to use the similarity between the *central* posts of the two threads (similar to using the most central entity to represent the collection in K-medoids [26]), centrality estimated by mechanisms such as those outlined in [12]. This leads to:

$$Central(X,Y) = BOW(X_{central}, Y_{central}) \quad (9)$$

where $X_{central}$ denotes the most central post in X.

4.2.8 Summarization Techniques

Unlike Section 4.1.2 where the query thread alone was summarized, we now summarize each of the threads in the corpus to separate documents. We then retrieve similar documents using traditional IR techniques using the query thread summaries as queries. Based on the summarization technique used, we denote the technique as either $All.MMR$ or $All.SM$ (All denotes that the query as well as the corpora are summarized). For each of the summarization technique, we summarize each corpus thread to between 25 and 50 words and pick the summary size that yields best performance; it turns out that the best performance was consistently achieved for summaries between 30 and 40 words.

HeadPost	threadID
Safari Crashing : ...safari is crashing almost every 5 minutes while surfing the web. I get immediately returned to the homepage and have to start over.....	1033829
Light leakage: Just upgraded from an original iPhone on Sunday to the iPhone 3g....I noticed that all down the left side between the iPhone casing ad the glass screen.. there is light leaking through..,	1968289
iPhone 4 Speaker Volume: ....earpiece volume on each has been astoundingly bad...speakerphone function is basically worthless and seems no louder whether it is on or off...	2459984
No sound in one earbud: .. when I play my music, no sound is emitted to one of the earbuds. ...	2537835
iPhone 4 in endless reboot: ..If I plug it into a USB power adapter .. connect to iTunes.. unplug the phone from the USB power adapter it shuts off ...get the endless reboot if I plug it into my MBP	2475944

Table 2: Sample queries. "threadID" is used to identify the thread in online apple discussion forums

5. EXPERIMENTAL EVALUATION

We now describe the empirical evaluation where we compare our techniques (listed in Section 3.2) against the baseline techniques detailed in Section 4. We use several standard measures (NDCG, MAP, MRR, Precision) in evaluation. We measure NDCG and Precision at various rank cut-offs of the search results (5, 10 and 15 results). We first describe the dataset used followed by describing an exhaustive set of results across techniques and performance measures, with all techniques using the tf.idf cosine similarity measure to model the $Sim(.,.)$ function. We then analyze the techniques across other popular similarity measures and varying values of algorithm parameters.

	Total	Query	Annotated
Total number of Threads	147,000	38	3412
Avg #posts per Thread	6	11	12
Avg #words per posts	76	81	91
Avg #words in title	7	6	8
Avg #words in first post	79	78	84

Table 3: Summary statistics of dataset used

5.1 Dataset

We crawled a dataset consisting of ∼147K threads from Apple Discussion Forums (*https://discussions.apple.com/*),

Apple's official discussion forum for sharing tips and solutions with other users. All the crawled web-pages were preprocessed and posts present in different webpages but belonging to the same thread were identified. For each thread, the crawl contained information such as title, firstpost, other posts, author information, reply-to link and time stamp.

The queries (threads) for this dataset were generated by identifying threads related to popular issues[8] in iPhone. In total, we generated 38 queries[9]. Some of the sample query threads are listed in Table 2. For each query thread, we query the collection using the keywords in the query thread to identify candidate threads. For a given query thread, we sought the help of human labelers to assign ternary relevance judgments to each candidate thread; 2 for *high relevance*, 1 for *partial relevance* and 0 for *irrelevant*. We had $\sim$ 85 candidate threads per query for the final evaluation and the labeling effort was roughly 60-80 minutes per query thread, leading to a total of 45 hours of human effort spent in labeling. It was often difficult to distinguish between *high relevance* and *partial relevance* threads; hence we combined these two relevance levels to give a binary labelling (*relevant*/*irrelevant*), used to evaluate our system. Inter-annotator agreement of 89% and 61% with Kappa coefficient of 0.68 & 0.42 was observed for the binary and ternary relevance judgements respectively.

5.2 Evaluation on Cosine tf.idf Similarity

Table 4 presents the performance of the various techniques considered, when the tf.idf cosine similarity is used for estimating text similarity within each of them. Under each measure, we highlight the best performing technique. All the approaches that use thread retrieval techniques are seen to be competitive to each other, with minor variations among performance trends. The better approaches among the ones that compare threads (i.e. BOP, HeadPost and BOP+HeadPost) significantly outperform the thread retrieval baselines. This is expected since thread retrieval techniques work with thread summaries (or excerpts such as first posts) whereas the thread comparison techniques such as BOP, HeadPost and BOP+HeadPost have the entire query thread at their disposal. MAX, Top-K and Central, techniques that use only a few pairwise post similarities are seen to not perform as well as BOP; this is likely to be because of certain generic posts (e.g., *help me please*) in the query thread that could spuriously boost the thread similarity estimates with those threads that also contain such posts. More significantly, the subsumption based approaches that we propose significantly outperform all the other techniques. Among them, the SUB-STR-TOP that uses both substructures and topics is found to perform the best in seven out of the eight evaluation measures considered.

5.3 Evaluation on other Similarity Measures

We considered two other similarity measures in our evaluation; the cosine similarity between normalized term-frequency vectors, and the Jaccard similarity co-efficient[17]. The performance trends on these similarity measures are largely similar to those observed in Table 4; hence, we plot only the NDCG@10, MAP and MRR values. The performance on the Cosine Similarity and Jaccard measures are given in

[8]http://www.iphonefaq.org/faq

[9]Please contact the first or second author to obtain a copy of the dataset with queries and relevance judgements.

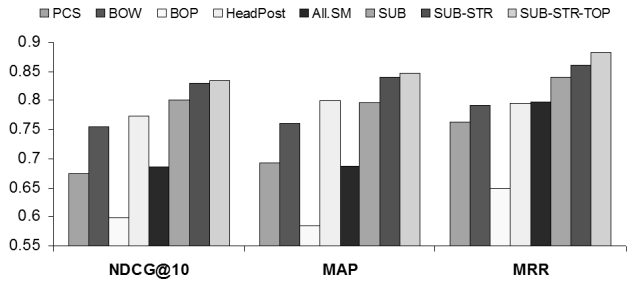

Figure 4: Results on Cosine Similarity

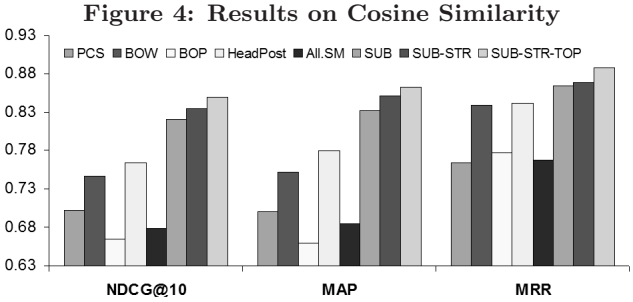

Figure 5: Results on Jaccard Similarity

Figure 4 and 5 respectively. BOW is seen to consistently beat BOP in these charts unlike in Table 4; this is expected since the lack of idf while comparing small text documents (BOP compares posts) could spuriously enhance similarities of dissimilar posts that contain common words. Our techniques are seen to be outperforming the others on all the evaluation measures with SUB-STR-TOP being consistently the best among on all the three evaluation measures.

5.4 Value of Usage of Components

Our technique incorporates the HeadPost-style similarity and the subsumption based assessment to arrive at an overall measure of similarity between threads. Usage of weighted post and post-reply pairs as components in subsumption-based similarity assessment is at the core of the technique that we propose. In this section, we isolate and analyze the benefit of using components in subsumption based similarity assessment independent of other factors such as HeadPost-style similarity and topic-based weighting.

Table 5 compares ranking of some selected candidate threads for the HeadPost and SUB-STR$^-$ (which denotes SUB-STR without the HeadPost i.e., the setting $\lambda = 1$) systems; the starred ones were judged as relevant by human labelers. For these chosen threads, title is indicative of their first post content; hence we have only shown title of the candidate threads. As evident for threads 1 and 2, highly similar title (and first posts) could mean that the threads are very similar; thus, the top few threads based on HeadPost similarity end up being relevant to the query thread. It is, hence, natural to expect that HeadPost would perform very well when the top few results alone are considered, such as indicated by NDCG@5 and MAP@5 measures. The effectiveness of HeadPost, however, deteriorates when threads with very similar titles and first posts are exhausted. This can be seen for candidates 5, 6 and 7 (that are all relevant) that have minimum title overlap with the query thread, thus falling behind in the ranking. Thus, HeadPost is less effective when more results are considered. This is indicative of the complementary nature of the thread content based similarity assessment and the HeadPost based similarities.

Figure 6 compares various techniques: BOP, SUB-STR$^-$,

Method	NDCG@5	NDCG@10	NDCG@15	P@5	P@10	P@15	MAP	MRR
PCS	0.741	0.737	0.703	0.775	0.738	0.695	0.771	0.779
Q.MMR	0.712	0.729	0.714	0.734	0.727	0.691	0.737	0.754
Q.SM	0.735	0.692	0.699	0.754	0.710	0.675	0.758	0.770
BOW	0.727	0.693	0.678	0.749	0.679	0.644	0.752	0.851
BOP	0.808	0.791	0.765	0.841	0.801	0.770	0.809	0.857
HeadPost	0.820	0.773	0.749	0.851	0.786	0.748	0.801	0.864
BOP+HeadPost	0.831	0.812	0.789	0.855	0.824	0.783	0.825	0.871
MAX	0.695	0.684	0.688	0.729	0.717	0.696	0.731	0.780
Top-K	0.706	0.707	0.721	0.759	0.749	0.724	0.765	0.819
Central	0.710	0.683	0.655	0.737	0.699	0.636	0.742	0.770
All.MMR	0.730	0.720	0.717	0.758	0.727	0.685	0.751	0.767
All.SM	0.763	0.727	0.719	0.774	0.746	0.694	0.771	0.808
SUB	0.851	0.836	0.811	0.866	0.831	0.799	0.851	0.909
SUB-TOP	0.872	0.846	0.830	0.871	0.847	0.809	0.870	0.914
SUB-STR	0.874	0.851	0.833	0.870	**0.850**	0.811	0.875	0.910
SUB-STR-TOP	**0.901**	**0.895**	**0.855**	**0.885**	0.844	**0.827**	**0.907**	**0.932**

Table 4: Experiment Results with tf.idf Cosine Similarity Measure.

Id	Title	HeadPost	SUB-STR$^-$
1*	SIM card not found message	1	4
2*	No SIM card installed msg	5	7
3	Sim card not supported	6	9
4	swap out the micro sim into a another phone?	47	11
5*	My phone had been LOCKED! Help plssssss....	59	8
6*	iPhone disabled.	69	6
7*	problems with iphone 3G update to OS 4.0.1	77	67
8	Import Contacts isnt working	82	23

Table 5: Ranking of candidate threads for a sample query thread with title "sim card not installed". * indicates that the candidate was adjudged relevant by the annotators.

HeadPost (i.e., SUB-STR with $\lambda = 0$), BOP+HeadPost and SUB-STR. BOP+HeadPost and SUB-STR are able to leverage the complementarity of the HeadPost and content based similarity aspects to beat approaches that consider only either of them. While SUB-STR outperforms BOP+HeadPost consistently, SUB-STR$^-$ is also competitive to BOP+HeadPost thereby indicating effectiveness of component based approach. SUB-STR$^-$ system has its own pitfalls due to not considering the HeadPost similarity, for e.g., it fails for cases when there is high lexical similarity between the candidate thread and the query thread but different aspects are discussed. This is evident from candidate 4 and 8 that are ranked high by SUB-STR$^-$, where a different issue related to iphone sim is discussed. This confirms the complementarity of HeadPost and content based similarities, and underlines that our notion of components is able to achieve significant gains over just considering threads as Bag Of Posts.

5.5 Significance Tests

Table 6 presents results of randomization tests [29], conducted for the most competitive baselines (i.e., BOP, BOP + HeadPost) and the various configurations of our techniques. While our techniques provide results that are statistically significant at a p-value < 0.05 over the baselines, it is interesting to note that SUB-STR-TOP's performance is statistically significant on each of MRR, NDCG@10, MAP wrt all the other techniques.

To further verify the utility of the post-reply pair substructure, we conducted another statistical significance test where in, apart from single posts, random post pairs were fed to SUB-STR as post-reply pairs; we call it SUB-STR$_{\text{Random}}$. The null-hypothesis that SUB-STR and SUB-STR$_{\text{Random}}$ are equivalent was rejected easily with a two-sided p-value < 0.015, thus empirically establishing the utility of real post-reply pairs over random post-pairs.

5.6 Effect of System Parameters

Our approaches have two parameters, λ, the relative weighting between subsumption based similarity and HeadPost similarity and then, the number of topics being considered. In this section, we analyze the sensitivity of our approaches to variations in these parameters and illustrate that our techniques are insensitive to variations on these parameters over a large range of their values.

5.6.1 Sensitivity to λ

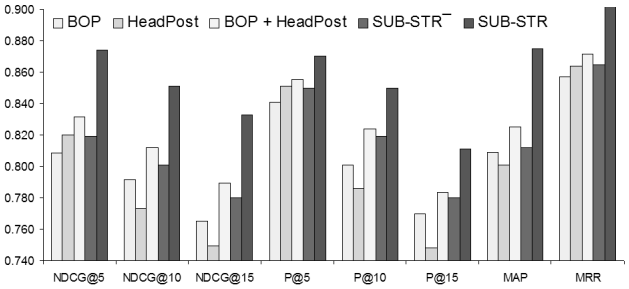

Figure 6: Effect of Usage of Components

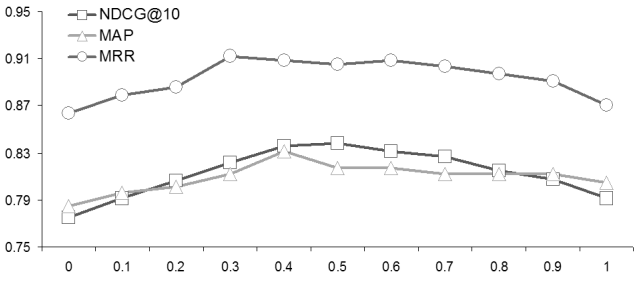

Figure 7: Sensitivity to λ

To enable analyze the effect of λ independent of the number of topics and usage of post-pairs as components, we plot the performance of SUB across varying values of λ in Figure 8. When $\lambda = 0$, the approach degenerates to the HeadPost technique, whereas at the other extreme, it discards the title and first post based similarity altogether. More interestingly, it fairs better than HeadPost (the left-most point of the curves) for any non-zero value of λ and is rather insensitive to λ values between 0.3 and 0.8.

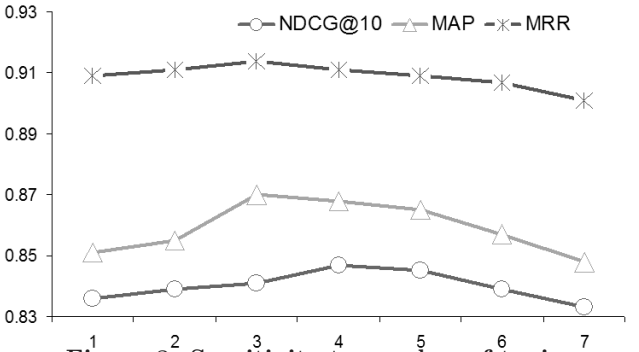

Figure 8: Sensitivity to number of topics

5.6.2 Sensitivity to number of topics

Selecting the right number of topics is an important problem in topic modeling [30]. Nonparametric models like the Chinese Restaurant Process [3] are not scalable to large datasets as can be expected in thread retrieval scenarios. Due to the popularity of generative approaches in topic modeling, we used LDA [4] for representing a component (and hence the entire thread) as mixture of topics. One could potentially use any other method for doing the same. To estimate the topic distribution for each component, we built our LDA model over a collection of 20000 threads with $\#topics = 100$. This is an offline process. Since we use topics to weigh the components, the number of topics in a component is another important parameter for our system. We plot the performance of the SUB-TOP on varying number of topics (restricting the component to belong to at most k topics). Similar to the observations for λ, the approach is largely insensitive to variations between 2 and 5.

6. RELATED WORK

The problem of finding threads similar to a given thread, to the best of our knowledge, has not been considered before, in literature. In this section, we provide a brief review of related literature. Firstly, we describe literature dealing with processing of discussion threads. Secondly, we review prior works focussing on finding similarities between semi-structured content such as XML (we will see shortly that discussion threads may be considered as semi-structured data).

Work on Discussion Threads: A forum thread, unlike text documents, are authored by multiple people and contain discussions on a particular topic. Upto 75% of the links from forums are found to link to noise pages like user profiles and login pages [35]; this makes traditional web page ranking techniques like PageRank [5] and HITS [20] inapplicable since such links are not indicative of recommendations. Thread/forum retrieval, the task of identifying relevant threads or posts within them in response to a user query (e.g., a phrase or 3-4 words), has been of much interest in recent years. One of the early works on forum retrieval attempted to adapt the random surfer model to induce links with content information (to content-wise similar pages) to rank forum pages better [35]. An extension to this model that utilizes information about common posters for link induction was presented in [7]. Subsequent works on thread retrieval moved away from the considering threads as collections of pages to considering them as collections of posts. [12] shows that thread retrieval improves by considering only some relevant posts, instead of all the posts in a thread. [28] illustrates that considering the post-reply pair (a post along with a reply to it) substructure in threads improves accuracy in thread retrieval. Though the problem that we address is significantly different in that we attempt to rank threads based on similarity to a *given thread* (which, unlike a user query, is typically larger and has a well-formed structure), we have evaluated our technique against adaptations of thread retrieval methods (in Section 5).

Work on XML Similarity: A forum thread may be represented as a tree with posts as nodes where a parent-child relationship exists between a post and it's reply. Additionally, siblings (multiple replies to the same post) may be ordered based on the temporal ordering. Each of these nodes (posts) could have attributes like time and poster id. It is intuitive to think of an XML schema to represent such threads due to the hierarchical nature (i.e., the tree). This leads to the question whether we can use XML similarity measures to estimate similarities between threads. However, traditional XML similarity measures make heavy use of the structural similarity [24] whereas we may not want to rely too much on structural similarity in estimating thread similarities. This is so since the content is likely to be more important than structure; the example in Table 1 shows two threads that are very similar despite the structural dissimilarity. Though macro-level structural similarity clearly cannot be used off-the-shelf to estimate thread similarities, certain types of sub-structures (e.g., post-reply pair) have been found to be useful in thread processing [28]. In this context, it is useful to note that research on XML similarity has gone beyond plain structural similarity [24] to hybrid methods that make use of semantic similarity of content [31]. A survey of XML similarity methods appears in [32]. In short, we argue that estimating thread similarities is different from estimating similarities between XML data since the notion of thread similarity relies more on content and certain sub-structures (unlike XML similarity where the similarities between macro-level structures is seen to be useful).

7. CONCLUSIONS AND FUTURE WORK

In this paper, we considered the problem of finding similar discussion forum threads to a query thread. Similarity Search on threads has numerous applications such as an enabler to providing links to similar threads in discus-

Method	BOP	BOP+Headpost	SUB	SUB-STR	SUB-TOP	SUB-STR-TOP
SUB	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	–			
SUB-STR	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	*map*	–		
SUB-TOP	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	*map*		–	
SUB-STR-TOP	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	*mrr*,*ndcg*,*map*	–

Table 6: Randomization Test Results. The measures on which the left column method is significant over the top row method at a p-value < 0.05 are indicated in the appropriate cell. We consider the *mrr*, *mdcg* (NDCG @ 10) and *map* evaluation measures.

sion forums and providing a clustered interface to present results in thread retrieval systems. Our similarity measure revolves around the notion of how well the threads being compared are mutually contained within each other. To estimate pairwise thread similarities, we model threads as a set of weighted overlapping components, whereby containment is quantified by the lexical similarity between components from the threads under consideration. Specifically, we use the post-reply component that has been found to be useful in discussion forum search, in addition to using individual posts as components; we proved that our similarity computation can run in polynomial time for this choice of component types. We outlined several intuitive methods that use thread retrieval techniques and direct thread comparison and evaluated our approaches against them. Through an extensive series of experiments on real world data, we established the effectiveness of our technique on popular similarity measures such as NDCG, MAP, Precision and MRR. Specifically, our techniques are seen to outperform the baselines by approximately 10% on an average on each of the measures.

We have considered two kinds of sub-structures, individual posts and post-pairs, in our technique. Incorporating more sophisticated structural components may enable better quality retrieval possible at the expense of increased similarity computation costs. Devising indexing techniques that could enable looking at just a subset of threads in response to a query thread could enhance performance. We are currently exploring ways to build a parameterized thread similarity measure that could be continuously tuned in response to user feedback. In email discussion groups, a single post may involve replies to multiple previous posts. Identifying such relationships and utilizing them in similarity modeling could potentially improve the retrieval quality.

Acknowledgements

We thank Srujana Merugu for discussions that helped us improve the final presentation.

8. REFERENCES

[1] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[2] S. Bhatia and P. Mitra. Adopting inference networks for online thread retrieval. In *AAAI*, 2010.

[3] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*. MIT Press, 2004.

[4] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*

[6] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. SIGIR '98, pages 335–336, 1998.

[7] Z. Chen, L. Zhang, and W. Wang. postingrank: bringing order to web forum postings. AIRS'08, pages 377–384, 2008.

[8] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *Computer Networks*, 31(11-16), 1999.

[9] P. M. Deshpande, D. P, and K. Kummamuru. Efficient online top-k retrieval with arbitrary similarity measures. EDBT '08, pages 356–367, 2008.

[10] H. Duan and C. Zhai. Exploiting thread structures to improve smoothing of language models for forum post retrieval. In *ECIR*, pages 350–361, 2011.

[11] R. Duan and S. Pettie. Approximating maximum weight matching in near-linear time. FOCS '10, pages 673–682, 2010.

[12] J. L. Elsas and J. G. Carbonell. It pays to be picky: an evaluation of thread retrieval in online forums. In *SIGIR*, pages 714–715, 2009.

[13] G. Forman, K. Eshghi, and S. Chiocchetti. Finding similar files in large document repositories. In *KDD*, pages 394–400, 2005.

[14] H. N. Gabow. *Implementation of algorithms for maximum matching on nonbipartite graphs.* PhD thesis, Stanford, CA, USA, 1974. AAI7413628.

[15] J. Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Trans. Web*, 3:12:1–12:33, September 2009.

[16] A. Huang. Similarity measures for text document clustering. pages 49–56, 2008.

[17] P. Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, 1912.

[18] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[19] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[20] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, September 1999.

[21] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. HLT '10, pages 912–920, 2010.

[22] I. Mani and M. T. Maybury. *Advances in Automatic Text Summarization*. 1998.

[23] F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *Computer Journal*, 26(4):354–359, 1983.

[24] A. Nierman and H. V. Jagadish. Evaluating structural similarity in xml documents. In *WebDB*, pages 61–66, 2002.

[25] D. P, D. Garg, and V. Varshney. Analysis of enron email threads and quantification of employee responsiveness. In *TextLink Workshop at IJCAI*, 2007.

[26] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36, March 2009.

[27] S. Sanghavi, D. Shah, and A. Willsky. Message passing for max-weight independent set. In *In NIPS*, 2007.

[28] J. Seo, W. B. Croft, and D. A. Smith. Online community search using thread structure. In *CIKM*, pages 1907–1910, 2009.

[29] M. D. Smucker, J. Allan, and B. Carterette. In *CIKM*, Lisboa, Portugal, 2007.

[30] M. Steyvers and T. Griffiths. *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007.

[31] J. Tekli, R. Chbeir, and K. Yétongnon. A hybrid approach for xml similarity. In *SOFSEM (1)*, pages 783–795, 2007.

[32] J. Tekli, R. Chbeir, and K. Yétongnon. An overview on xml similarity: Background, current trends and future directions. In *Computer Science Review 3(3)*, 2009.

[33] A. Tversky. Features of Similarity. In *Psychological Review*, volume 84, pages 327–352, 1977.

[34] R. C. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical report, 2000.

[35] G. Xu and W.-Y. Ma. Building implicit links from content for forum search. In *SIGIR*, pages 300–307, 2006.