On the Selection of Tags for Tag Clouds

Petros Venetis
Computer Science Dept.
Stanford University
venetis@cs.stanford.edu

Georgia Koutrika
IBM Almaden Research Center
gkoutri@us.ibm.com

Hector Garcia-Molina
Computer Science Dept.
Stanford University
hector@cs.stanford.edu

ABSTRACT

We examine the creation of a tag cloud for exploring and understanding a set of objects (e.g., web pages, documents). In the first part of our work, we present a formal system model for reasoning about tag clouds. We then present metrics that capture the structural properties of a tag cloud, and we briefly present a set of tag selection algorithms that are used in current sites (e.g., `del.icio.us`, Flickr, Technorati) or that have been described in recent work. In order to evaluate the results of these algorithms, we devise a novel synthetic user model. This user model is specifically tailored for tag cloud evaluation and assumes an "ideal" user. We evaluate the algorithms under this user model, as well as the model itself, using two datasets: CourseRank (a Stanford social tool containing information about courses) and `del.icio.us` (a social bookmarking site). The results yield insights as to when and why certain selection schemes work best.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: Miscellaneous; H.3.3 [**Information Search and Retrieval**]: Search process; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Design, Experimentation, Measurement

Keywords

tags, tag clouds, tag selection

1. INTRODUCTION

A tag cloud is a visual depiction of text content. Tags are words found in or associated with the underlying content (e.g., annotations) and are typically listed alphabetically and in different color or font size based on their importance [16]. Tags in the cloud are hyperlinks. The user can click on a tag and see related content.

Tag clouds have been popularized by social sites, such as Flickr, Technorati and `del.icio.us`, where users annotate shared resources using short textual labels, namely tags. In these sites, a tag cloud typically summarizes a collection of resources by showing the most popular tags, i.e., the tags that are most often used for describing resources. For example, in Flickr, users share photos. A user may upload a photograph that was taken at the Golden Gate bridge at San Francisco and annotate it with the tags '*ca*', '*california*', '*sanfrancisco*' and '*goldengate*'. A search for photos tagged with '*california*' would return a list of photos like the previous one and a tag cloud summarizing these photos, which could contain tags such as: '*sanfrancisco*', '*ocean*', '*beach*', '*water*', '*sunset*', '*bridge*', '*pacific*', '*usa*', '*nature*', '*waves*'.

Tag clouds have been used in other settings beyond social information sharing sites, including personal and commercial web pages, blogs, and search engines. They are used for summarizing web search results [2], results over biomedical databases [9], and results of structured queries [8]. In these scenarios, tags are extracted from the textual content of the results and are weighted based on their frequencies. For instance, PubCloud generates a tag cloud from words extracted from the abstracts returned by a query over the biomedical database PubMed [9].

Despite their popularity, little is known about what makes a tag cloud "good" for summarizing and navigating through a particular collection. What properties should a tag cloud have? For instance, in social sites, tags are typically selected based on their popularity and a predefined number of them (usually $\sim$35) is shown to the user [16]. Are these tags a good choice and why? For example, for the earlier query '*california*', Flickr may return 1,000 photos. If we select the 35 most popular tags for the cloud, there might be some photos in the query results that have not been assigned any of the selected 35 tags. These would not be "covered" by the tag cloud and hence would be unreachable by the user clicking on tags. On the other hand, if the query returns just 20 photos, then using (any) 35 tags to represent them may be too much, because tags may overlap, such as '*usa*' and '*us*', or be assigned to just one photo.

Furthermore, how can we evaluate the output of an algorithm that selects tags to summarize a corpus? In what sense, is a tag cloud "better" than another for the same corpus? How do we compare the output of different algorithms (e.g., [2, 8, 9])? If a tag selection approach works for one setting, will it be appropriate for different settings? For example, an earlier work has shown that when summarizing search results over a database, showing the most frequent (popular) words found in the results may provide trivial and not very meaningful groupings that cannot help the searcher locate something interesting. For example, if all photos returned for the query '*california*' are also tagged with the word '*usa*', then the latter is not a good candidate summary word.

With all these questions in mind, in this paper, we focus on *the tag selection problem for summarizing a set of results generated by a query*. (We are not interested in the actual tag layout seen

WSDM'11, February 9–12, 2011, Hong Kong, China.

by the user.) We define *a set of metrics that capture the structural properties of a tag cloud*. For example, we define the *coverage* of a tag cloud to show which fraction of the underlying content is described by the cloud and how significant that part is.

Based on these metrics, we develop a *user satisfaction model* in order to evaluate the quality of a tag cloud for any given query. Our user model is based on the concept of *failure probability*. This probability reflects the likelihood that a user is unable to satisfy his search goal by using the tag cloud.

Using our framework, we provide a *quantitative analysis of several algorithms* for tag selection and we study the quality of their output. We also provide guidelines of how an algorithm should be devised to minimize the probability of failing to satisfy the user's need by empirically examining the failure probability against our defined metrics.

For the evaluation, we use two real datasets. The first is a set of (web) documents and tags from the social bookmarking site `del.icio.us`. The other is the course database used in CourseRank, a social course planning tool we have developed in InfoLab at Stanford [1]. CourseRank displays official university information and statistics, such as bulletin course descriptions and grade distributions, as well as unofficial information, such as user ratings and reviews. As part of the system, we have implemented CourseCloud [8] that allows students to explore courses with the help of tag clouds. In this case, the cloud contains the most significant or representative terms found within the results of a search for courses aggregated over all parts that describe a course, such as the title, the comments, etc.

Contributions. In summary, our contributions are the following:

- We present a framework for reasoning about tag clouds (Section 2).
- We explore algorithms and metrics for generating and evaluating tag clouds (Sections 3 and 4).
- We develop a user model that captures the "usefulness" of a tag cloud (Section 5). We show (Section 6.5) via a user study that our model is good for predicting tag clouds humans prefer.
- We evaluate (Section 6) tag creation algorithms using our user model and two real datasets: one from CourseRank, our course planning site in Stanford, and one from the social bookmarking site `del.icio.us`.

2. SYSTEM MODEL

We consider a set $\mathcal{C}$ of objects (e.g., photos, web pages, etc), and a set $\mathcal{T}$ of tags. These tags may be textual labels assigned to the objects (e.g., tags assigned to photos) or they may be words extracted from the content of the objects (e.g., words found in the text of a web page or in a course description) or pre-existing category labels used to classify documents, and so forth. Each object c in $\mathcal{C}$ is associated with a subset of tags from $\mathcal{T}$.

For simplicity, but without loss of generality, we assume that a query q belongs to $\mathcal{T}$. Given a query q, the set of results is a set $\mathcal{C}_q \subseteq \mathcal{C}$. We will denote by $\mathcal{T}_q$ the set of tags that are associated with at least one of the objects of $\mathcal{C}_q$; obviously $\mathcal{T}_q \subseteq \mathcal{T}$. We also define the *association set* $A_q(t)$ *of tag* t *under query* q:

DEFINITION 1. *The association set* $A_q(t)$ *of a tag* $t \in \mathcal{T}_q$ *under query* q *is the set of objects in* $\mathcal{C}_q$ *associated with tag* t. □

Observe that $A_q(t) \subseteq \mathcal{C}_q$ by definition.

In our work, we will neither delve into how the set $\mathcal{C}_q$ of results is generated for a query q nor examine how it is actually ranked. We will only assume that there is a *partial (scoring) function* $s(t, c) : \mathcal{T} \times \mathcal{C} \to [0, 1]$. An object c belongs to $\mathcal{C}_q$ iff $s(q, c)$ is defined for the partial function $s(\cdot, \cdot)$. For a query q, the scoring function establishes a partial ordering between objects in $\mathcal{C}_q$. For any two objects $c_i, c_j \in \mathcal{C}_q$ if $s(q, c_i) > s(q, c_j)$, then c_i is ranked higher than c_j; if $s(q, c_i) = s(q, c_j)$, then either c_i or c_j is ranked higher.

We also assume that there is a *similarity function* between objects. This similarity function $sim(\cdot, \cdot) : \mathcal{C} \times \mathcal{C} \to [0, 1]$, takes as inputs two objects out of our set $\mathcal{C}$ and outputs a number between 0 and 1. The higher the value of $sim(\cdot, \cdot)$ is, the more similar the two objects are. For example, if we are dealing with documents, the similarity function could be the Jaccard similarity based on the words found in each document. If the objects are photographs, the similarity function could measure the proximity of the places depicted in the photographs taking into account geo-referenced data.

A *tag cloud* is a subset $\mathcal{S}$ of the tags in $\mathcal{T}_q$ for a particular query q. Its purpose is two-fold: it is used for summarizing the query results and helping navigation. Intuitively, one could imagine that a "good" or effective tag cloud should, in some fundamental sense, be representative of the objects in $\mathcal{C}_q$ and should help the user find the information he wants. To help quantify the properties of tag clouds, we define a set of metrics next.

3. METRICS

We first describe some basic quantities used in our definitions. For a set of objects $\mathcal{C}_p \subseteq \mathcal{C}$:

- $|\mathcal{C}_p|$ is the number of elements in $\mathcal{C}_p$.
- $|\mathcal{C}_p|_{s,q} = \sum_{c \in \mathcal{C}_p} s(q, c)$ is the *scored size* of the set $\mathcal{C}_p$ under query q. It is the weighted size of set $\mathcal{C}_p$, where the elements have a weight equal to their score under query q given by the scoring function s. The higher the score of an element is, the more it contributes to the sum.
- $|\mathcal{C}_p|_{1-s,q} = \sum_{c \in \mathcal{C}_p} \{1 - s(q, c)\}$ is the $(1 - s)$-*scored size* of the set $\mathcal{C}_p$ under query q. It is the weighted size of set $\mathcal{C}_p$, where the elements have a weight equal to one minus their score under query q. Thus, the higher the score of an element is, the less it contributes to the sum.

Given a query q, the set $\mathcal{C}_q$ of query results, and the set $\mathcal{T}_q$ of tags associated with objects in $\mathcal{C}_q$, we define the following metrics on a subset (tag cloud) $\mathcal{S}$ of $\mathcal{T}_q$. Note that some of these metrics (or simpler variations) have been used to evaluate tag clouds or other type of sets of terms or documents (see Section 7). Our goal here is to lay out the wide spectrum of choices available, using a consistent terminology and in the specific context of tag clouds.

Extent of $\mathcal{S}$: The extent $ext(\mathcal{S})$ of $\mathcal{S}$ is the cardinality of $\mathcal{S}$, i.e.:

$$ext(\mathcal{S}) = |\mathcal{S}| .$$

Intuitively, the larger the extent of $\mathcal{S}$ is, the more (not necessarily distinct) topics $\mathcal{S}$ potentially covers.

Coverage of $\mathcal{S}$: Some objects in $\mathcal{C}_q$ may not be associated with any tag from $\mathcal{S}$. Then, these objects are not covered by $\mathcal{S}$. Coverage gives us the fraction of $\mathcal{C}_q$ covered by $\mathcal{S}$. When computing this fraction, we weigh covered objects by their scores, since it is more important to cover the high-ranked objects than the low-ranked objects. Thus, instead of using plain cardinality of sets, we use *scored* sized sets, and we define the coverage $cov(\mathcal{S})$ of $\mathcal{S}$ as the scored size of the objects that are associated with $\mathcal{S}$ over the scored size of the original set $\mathcal{C}_q$, i.e.:

$$cov(\mathcal{S}) = \frac{\left| \bigcup_{t \in \mathcal{S}} A_q(t) \right|_{s,q}}{|\mathcal{C}_q|_{s,q}}.$$

This metric can take values between 0 and 1. If $cov(\mathcal{S})$ is close to 0, then $\mathcal{S}$ is associated with a few objects of $\mathcal{C}_q$, and/or with

objects in $\mathcal{C}_q$ that have a very low score. On the contrary, if $cov(\mathcal{S})$ is close to 1, then the set $\mathcal{S}$ is associated with many of the objects in $\mathcal{C}_q$, and/or with objects that have a very high score.

EXAMPLE 1. For our examples in this section, we use our CourseRank setting, where users search for courses. Let us assume that we will pick one tag for summarizing the query results (i.e., extent is 1,) and the query was 'history' (and $s(t,c) = 1$ for all t and c). Two candidate tags are '*Roman*' and '*Ethiopian*'. The tag '*Roman*' gives higher coverage than '*Ethiopian*', since there are many more Roman history courses at Stanford than Ethiopian ones. □

Overlap of $\mathcal{S}$: Different tags in $\mathcal{S}$ may be associated with the same objects in $\mathcal{C}_q$. The overlap metric captures the extent of such redundancy. If tags overlap for low-scored objects, then these objects are over-represented in $\mathcal{S}$, so we want to penalize that. On the other hand, if they overlap on high-scored objects that are more relevant to the given query, it makes sense to give a lower penalty. Thus, we define the overlap $over(\mathcal{S})$ of $\mathcal{S}$ as the average pairwise $(1-s)$-scored size of the intersection of the associated objects of two tags over the minimum $(1-s)$-scored size of the two sets:

$$over(\mathcal{S}) = \operatorname*{avg}_{t_1,t_2 \in \mathcal{S}, t_1 \neq t_2} \left\{ \frac{|A_q(t_1) \cap A_q(t_2)|_{1-s,q}}{\min_{i \in \{1,2\}} \left\{ |A_q(t_i)|_{1-s,q} \right\}} \right\}$$

This metric also takes values in the range $[0,1]$. If $over(\mathcal{S})$ is close to 0, then the intersections of the association sets are very small or contain highly scored objects. In this case, each tag reveals a "different aspect" of the query results from the other tags. On the other hand, if $over(\mathcal{S})$ is close to 1, the intersections of the pairs of association sets usually contain all the objects of the smallest set of the pairs, or they include a lot of the low-scored objects. In these cases, the tags tend to be not very distinctive or they tend to be biased towards the less important objects in $\mathcal{C}_q$.

EXAMPLE 2. Let us assume a query about 'computer science' courses and that we want to select two tags to show in the tag cloud. One could choose the tags '*programming languages*' and '*compilers*' but these obviously overlap to a great degree. On the other hand, choosing '*programming languages*' and '*databases*' may be a better choice since these are clearly different topics (with small if any overlap) and they can provide a clearer distinction between the courses in the result set. □

Cohesiveness of $\mathcal{S}$: Cohesiveness looks at how closely related the objects in each association set of $\mathcal{S}$ are. For this purpose, we use the similarity function defined between the objects in $\mathcal{C}$. First, we define the average (intra)similarity of the association set $A_q(t)$ of a tag $t \in \mathcal{S}$ based on the pair-wise similarities of the objects in $A_q(t)$:

$$\overline{sim}(t) = \frac{\sum_{c_1,c_2 \in A_q(t), c_1 \neq c_2} sim(c_1,c_2)}{|A_q(t)| \times (|A_q(t)|-1)}$$

The cohesiveness $coh(\mathcal{S})$ of $\mathcal{S}$ is defined as the average of (intra)similarities of the association sets of the tags in $\mathcal{S}$[1]:

$$coh(\mathcal{S}) = \text{avg}_{t \in \mathcal{S}} \left\{ \overline{sim}(t) \right\}$$

Since the similarity function sim yields values in $[0,1]$, both $\overline{sim}(t)$ and $coh(\mathcal{S})$ produce values in the same range. The higher $coh(\mathcal{S})$ is, the more similar the objects in the association sets are to each other. If cohesiveness is low, then some tags will describe heterogeneous sets of objects that do not share any distinctive features.

EXAMPLE 3. Assume that our query is 'computer science' and our tag cloud will show one tag. We want to compare two extremes: the tag '*databases*' and the tag '*engineering*'. Courses with both '*computer science*' and '*databases*' tags are likely to be closely related, while '*computer science*' and '*engineering*' courses are likely to cover a much wider spectrum of topics. Thus, we expect $\overline{sim}(\text{'}databases\text{'}) \gg \overline{sim}(\text{'}engineering\text{'})$. In addition, although '*engineering*' may cover more courses than the tag '*databases*', the variety of covered courses may make it hard for a user to locate a course of interest. On the other hand, '*databases*' courses form a coherent set, and thus if the tag seems interesting to the user and is clicked, it will lead to potentially interesting courses. □

Relevance of $\mathcal{S}$: Another interesting question is how relevant the tags in $\mathcal{S}$ are to the original query q. To answer this question, we treat each t in $\mathcal{S}$ as a query and we consider the set $\mathcal{C}_t$ of objects that this query returns. The more $\mathcal{C}_t$ and $\mathcal{C}_q$ overlap, the more related t is to q. If $\mathcal{C}_t \subseteq \mathcal{C}_q$, then t is practically a sub-category of the original query q. Let us first define the relevance $rel(t,q)$ of a tag t to the original query q as the fraction of results in $\mathcal{C}_t$ that also belong to $\mathcal{C}_q$, i.e.:

$$rel(t,q) = \frac{|\mathcal{C}_t \cap \mathcal{C}_q|}{|\mathcal{C}_t|}.$$

The relevance $rel(\mathcal{S})$ of $\mathcal{S}$ is the average relevance of its tags, i.e.:

$$rel(\mathcal{S}) = \operatorname*{avg}_{t \in \mathcal{S}} \{rel(t,q)\}$$

$rel(\mathcal{S})$ lies in $[0,1]$. The closer it is to 1, the more relevant are the tags of $\mathcal{S}$ to the original query q.

EXAMPLE 4. Consider the query 'computer science' again and two potential tags for the cloud: '*theory*' and '*relational databases*'. There are '*theory*' courses related to '*computer science*' but many more are related to other disciplines. Hence, '*theory*' is not exclusively related to '*computer science*'. On the other hand, all courses on '*relational databases*' are computer science courses, hence this tag is a highly-relevant topic in computer science. □

Popularity of $\mathcal{S}$: A tag from $\mathcal{S}$ is popular in $\mathcal{C}_q$ if it is associated with many objects in $\mathcal{C}_q$. We define the popularity $pop(\mathcal{S})$ of $\mathcal{S}$ as the fraction of the sum of the popularities of all tags in $\mathcal{S}$ over the maximum possible sum of popularities of any $|\mathcal{S}|$ tags that are a subset of $\mathcal{T}_q$, i.e.:

$$pop(\mathcal{S}) = \frac{\sum_{t \in \mathcal{S}} |A_q(t)|}{\max_{\{t_1,\ldots,t_{|\mathcal{S}|}\} \subseteq \mathcal{T}_q} \left\{ |A_q(t_1)| + \ldots + |A_q(t_{|\mathcal{S}|})| \right\}}$$

Independence of $\mathcal{S}$: All objects in the association set of a tag t_1 may be similar to each other but they may be also similar to the objects in the association set of a second tag t_2. In this case, the two tags may have very similar semantics. Thus, we evaluate the similarity (or non-similarity) of pairs of different association sets for tags in $\mathcal{S}$. We define a metric similar to cohesiveness, but focused on pairs of association sets.

First, we define the average similarity between two tags $t_1, t_2 \in \mathcal{S}$ as[2]:

$$\overline{sim}(t_1,t_2) = \frac{\sum_{c_1 \in A_q(t_1), c_2 \in A_q(t_2)} sim(c_1,c_2)}{|A_q(t_1)| \times |A_q(t_2)|}$$

[1]We could also use minimum or other aggregate functions to define cohesiveness given the $\overline{sim}$ values. We will not examine other aggregate functions in this work though.

[2]Note that the $\overline{sim}$ operator is overloaded.

Then, we define the independence $ind(\mathcal{S})$ of $\mathcal{S}$ as:

$$ind(\mathcal{S}) = 1 - \left\{ \underset{t_1,t_2 \in \mathcal{S}}{\text{avg}} \left\{ \overline{sim}(t_1, t_2) \right\} \right\}$$

$ind(\mathcal{S})$ takes values in $[0, 1]$. The higher the value of $ind(\mathcal{S})$, the more "independent" different tags are, which means that they refer to dissimilar sets of objects. If it is 0, then many tags in $\mathcal{S}$ will be synonyms or one may be very related to each other.

EXAMPLE 5. The two tags '*programming*' and '*software*' may identify different courses (e.g., 'Introduction to programming' vs. 'Object-Oriented Software Design') but these courses are very similar, hence the two tags are not independent. □

Balance of $\mathcal{S}$: To take into account the relative sizes of association sets, we define the balance $bal(\mathcal{S})$ of $\mathcal{S}$ as follows:

$$bal(\mathcal{S}) = \frac{\min_{t_i \in \mathcal{S}} \{|A_q(t_i)|\}}{\max_{t_j \in \mathcal{S}} \{|A_q(t_j)|\}}$$

By definition, $bal(\mathcal{S})$ takes values in $[0, 1]$. The closer it is to 1, the more balanced our association sets are.

EXAMPLE 6. Assume we have the query 'history', and two candidate tag clouds: {'*Greek*', '*Roman*'} vs. {'*European*', '*Ethiopian*'}. The tag '*Ethiopian*' in the second tag cloud points to very few courses whereas the tag '*European*' is quite generic since it points to many courses and would require further drilling down in this set of courses. Thus, the second tag cloud is highly imbalanced and hence not very useful. The first tag cloud is more balanced. □

4. ALGORITHMS

A tag selection algorithm takes as input a set $\mathcal{C}_q$ of objects to be summarized, a set $\mathcal{T}_q$ of tags that can be used to describe these objects, and generates a tag cloud $\mathcal{S} \subseteq \mathcal{T}_q$.

A tag selection algorithm can be single or multi-objective. A single-objective algorithm has a single goal in mind, e.g., maximizing popularity of the cloud. A multi-objective algorithm tries to strike a balance among two or more goals, e.g., achieving high popularity and coverage. Although multi-objective algorithms have been proposed (mainly [3]), in this paper we focus on four single-objective algorithms that have been used in practice or that are very close to those used. We believe that trends become more apparent with single-objective algorithms, and hence our focus. Although not reported here, we have studied multi-objective algorithms, and their behavior tends to be a combination of simpler algorithms. That is, an algorithm that tries to optimize metrics X and Y performs "somewhere between" an algorithm that optimizes just X and one that optimizes just Y.

The most common algorithm used in social information sharing sites is a popularity-based algorithm, described in Section 4.1. A class of algorithms for structured data is described in [8]. We describe two algorithms in this class in Section 4.2. Our fourth algorithm (Section 4.3) attempts to maximize coverage, using an approximate solution to the maximum coverage problem. All these algorithms take as input the desired number of tags for the cloud, k, i.e., the extent of the tag cloud.

4.1 Popularity algorithm (POP)

Showing popular tags in the cloud allows a user to see what other people are mostly interested in sharing. For a query q and parameter k, the algorithm POP returns the top-k tags in $\mathcal{T}_q$ according to their $|A_q(t)|$. Thus, the algorithm maximizes the popularity metric we defined in Section 3 (giving a value of 1 for this metric).

4.2 Tf-idf based algorithms (TF, WTF)

A different approach is to select good quality tags for the cloud by giving preference to tags that seem very related when compared against the objects in $\mathcal{C}_q$ [8]. Given a query q and a scoring function $s(\cdot, \cdot)$, this approach proceeds as follows.

Algorithm 1: Tf-idf based algorithm

Input: $\mathcal{C}_q, \mathcal{T}_q, tfidf(t, c), s : \mathcal{T}_q \times \mathcal{C}_q \rightarrow [0, 1]$
Output: $\mathcal{S} \subseteq \mathcal{T}_q$

```
/* PQ is a priority queue.                          */
PQ ← ∅, S ← ∅;
// Compute all the aggregate confidences
foreach t ∈ T_q do
    r ← C_t ∩ C_q;
    w ← aggregation_{c∈r} {f(q, t, c)};
    PQ.insert(w, t);
// Select the top-k most confident tags
while (|S| ≥ k) ∧ (PQ.size() > 0) do
    S ← S ∪ {PQ.top().getTag()};
    PQ.pop();
return S;
```

For each tag $t \in \mathcal{T}_q$, it builds the set $A_q(t)$. In order to determine whether t is highly related to the query results and hence it makes a good selection for the tag cloud, the algorithm produces a score for t by aggregating the values of a function $f(q, t, c)$ (which will be explained shortly) over all objects $c \in A_q(t)$; this aggregation yields a score w for tag t. The algorithm returns the k tags with the highest scores. Function $f(q, t, c)$ can have the following forms:

- $f(q, t, c) = s(t, c)$, or
- $f(q, t, c) = s(t, c) \cdot s(q, c)$.

The first form of $f(q, t, c)$ uses the scoring function s to capture how related tag t and an object $c \in A_q(t)$ are. For example, $s(t, c)$ could be a tf-idf scoring function. If t is very related to the objects in $A_q(t)$, then it is probably a good tag to be used for describing these objects. The second form of $f(q, t, c)$ weighs the relatedness between t and c, $s(t, c)$, by how related object c is to the query q. The intuition is that a tag t is more significant if it is very related to objects in the query results that are highly related to the query q.

In our experimental study, we evaluate both forms of function $f(q, t, c)$. Since in our implementation we use as scoring function the tf-idf function between tags and objects, we call the first method *tf-idf* (TF for short) and the second one *weighted tf-idf* (WTF). Also, we use summation as our aggregation function over the objects in $A_q(t)$ (average is another of the many possibilities).

4.3 Maximum coverage algorithm (COV)

The maximum coverage algorithm (COV) attempts to maximize the coverage of the resulting set of tags $\mathcal{S}$ with the restriction that $|\mathcal{S}| \leq k$. Our problem is a classic maximum coverage problem, cast in our setting of tags: Our universe of elements (objects) is $\mathcal{C}_q$. We are given subsets $A_q(t_1), A_q(t_2), \ldots, A_q(t_m)$ and a parameter k, an upper bound of the extent of $\mathcal{S}$. Our goal is to find a set $\mathcal{S}$ of at most k tags that maximizes $cov(\mathcal{S})$. This problem (even without weights/scores) is known to be NP-Complete [15].

Even though our problem is NP-Complete, there are good approximations for it [15]. These approximations have a guaranteed approximation ratio of $\frac{e}{e-1}$. Our COV algorithm (in Algorithm 2) uses a greedy heuristic to find a high coverage tag set. It starts with an empty $\mathcal{S}$. Then, at each round, it adds a tag to $\mathcal{S}$ from $\mathcal{T}_q$ that covers the largest number of uncovered objects by that point. The algorithm stops after adding k tags to $\mathcal{S}$ or when there are no more tags in $\mathcal{T}_q$.

Algorithm 2: Max coverage algorithm

Input: $\mathcal{C}_q, \mathcal{T}_q, k, s : \mathcal{T}_q \times \mathcal{C}_q \rightarrow [0,1]$
Output: $\mathcal{S} \subseteq \mathcal{T}_q$

$V \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$;

// Greedily maximize coverage
while $(|\mathcal{S}| \leq k) \wedge (\mathcal{T}_q \neq \emptyset)$ **do**
  $\hat{t} = \arg\max_{t \in \mathcal{T}_q} \{|V \cup A_q(t)|\}$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{\hat{t}\}$;
  $V \leftarrow V \cup A_q(\hat{t})$;
  $\mathcal{T}_q \leftarrow \mathcal{T}_q - \{\hat{t}\}$;
return $\mathcal{S}$;

5. USER MODEL

A natural question to ask when comparing tag selection algorithms is, which algorithm yields more "useful" clouds? Unfortunately, answering this question is very hard. The goal of tag clouds is to summarize the available data so the user can better understand what is at his disposal. A second goal is to help users discover unexpected information of interest, information that is not reachable via traditional search since the user does not know how to formulate a precise query. Neither of these goals can be stated precisely. Furthermore, it is hard to ask users to quantify how well a cloud summarizes what is available, and what users find "interesting" or "unexpected but interesting" varies widely. The variation can be wide and trends are often hard to observe.

Because of these difficulties, we postulate instead an *ideal user satisfaction model*. This "ideal" user is satisfied in a very predictable and rational way. For instance, a tag cloud that is cohesive and relevant will lead him to objects of interest more often than a tag cloud that is not cohesive and/or relevant (more details below).

It is important to note that we do not use the ideal model to *predict* user satisfaction, we only use it to compare tag clouds. Just like an "average person" does not exist, our ideal user does not exist so his particular satisfaction score (low failure probability as defined below) is not important. The scores are only used to compare clouds, e.g., produced by different algorithms. Our experimental results in Section 6.5 indeed do show that our model is relatively good for predicting clouds users prefer.

5.1 Base model: Coverage

Say our ideal user has posed a query q, and he is interested in just one of the objects in $\mathcal{C}_q$, call it g. We assume that $\mathcal{C}_q$ is a large set; if not, the user could just scan the query results and find what he wants with no need for a tag cloud. We also assume the user does not know how to refine q to narrow down his search to g. If he knew in advance precisely what he wanted, he would again not need the tag cloud.

The tags in the tag cloud $\mathcal{S}$ define a set of objects O covered by the cloud. That is, O is the union of the $A_q(t)$ sets for all $t \in \mathcal{S}$. For now, assume there is no overlap between the $A_q(t)$ sets, and assume the score $s(q,c) = 1$ for all $c \in \mathcal{C}_q$ (i.e., the content of each object is equally relevant to q).

The objects in O may or may not be of more interest to the user than the objects in $\mathcal{C}_q - O$. For instance, if the tags in $\mathcal{S}$ are popular ones, and there is a correlation between popularity and what the user wants, then an object in O may be more likely to be the desired object g than the rest. On the other hand, if the user is browsing for totally unexpected objects, then all objects in $\mathcal{C}_q$ may be of equal interest (will be equally likely to be g).

To capture this range of scenarios, let us assume that each object in $\mathcal{C}_q - O$ has probability p of being the one the user is interested in, while objects in O have probability $r \cdot p$, where $r \geq 1$. Parameter r captures how correlated the content of the tag cloud is to the user intention. If $r = 1$, then we have a "discovery" scenario, where all objects are equally likely to be of interest. The larger r is, the more likely it is that the object of interest, g, will be in O (which will be the case for example if popularity of tags is strongly correlated with user's satisfaction).

EXAMPLE 7. Say $r = 2$, $\mathcal{C}_q$ has 10 objects, and O has 6 objects. Since the user is interested in one object, then $6 \cdot rp + (10-6) \cdot p = 1$, so $p = 1/16 = 0.0625$. With probability $(10-6) \cdot p = 4/16$, the object g will not be reachable through one of the tags in $\mathcal{S}$. We call this $4/16 = 0.25$ the failure probability because if the user only explores by clicking on tags in the cloud, he will not find g. If coverage increases to say 8 objects in O (r is still 2), then $8 \cdot rp + (10-8) \cdot p = 1$, and thus $p = 1/18$. In this case, the failure probability has decreased to $2/18 \approx 0.11$. If instead we go back to 6 objects in O but say that the tags have more influence over user interest and $r = 3$, then we have $6 \cdot rp + (10-6) \cdot p = 1$ and $p = 1/22$, and the probability of failure also decreases to $4/22 \approx 0.18$. If both $r = 3$ and O has 8 objects, then $p = 1/26$ and the failure probability is $2/26 \approx 0.08$. □

As illustrated, our model assigns a probability to each object in $\mathcal{C}_q$, representing the likelihood that the object is g. This distribution is first expressed in terms of p, the probability that an object not covered by the cloud is the one of interest. Then we solve for p knowing that the probabilities must add up to 1. Once we know the distribution, we can compute our satisfaction metric:

DEFINITION 2. *The failure probability ($\mathcal{F}$) for a query q is the probability that the user will not be able to satisfy his search need using the tag cloud $\mathcal{S}$.* □

The failure probability is computed by summing up the probabilities p that an object not covered by the cloud is the one of interest, over all objects not covered. The model has a single parameter r that captures how focused the user exploration is. Although very simple, this initial model already sheds light on how coverage and tag popularity impact user satisfaction.

Summary: We assume that overlap is 0 and that $s(q,c) = 1$ for all $c \in \mathcal{C}_q$. Our first model incorporates only coverage. The probability that an object $c \in \bigcup_{t \in \mathcal{S}} A_q(t)$ is of the user's interest is $r \cdot p$, while the probability that an object $c \in \mathcal{C}_q - \bigcup_{t \in \mathcal{S}} A_q(t)$ is of the user's interest is p.

5.2 Incorporating Relevance

Each tag t appears in two types of objects, those in $A_q(t)$ (which belong in the query result $\mathcal{C}_q$) and those in $\mathcal{C}_t - A_q(t)$ (which do *not* belong in the query result). In the above analysis we ignored the size of $\mathcal{C}_t$. We will incorporate it to our user model in this section. We continue using p and r for the same notions as in Section 5.1.

EXAMPLE 8. Say $q =$'cat' and set $\mathcal{S}$ has two tags, $t_1 =$ '*kitten*' and $t_2 =$ '*animal*' with the following characteristics: $|A_q(t_1)| = 900 > |A_q(t_2)| = 100$ and $|\mathcal{C}_{t_1}| = |\mathcal{C}_{t_2}| = 1000$. Say both tags are equally popular and there is still no overlap.

Notice that most of the objects with tag '*kitten*' also have tag '*cat*', thus '*kitten*' seems very related (relevant) to '*cat*'. On the other hand, few of the '*animal*' objects have the '*cat*' tag. Thus, it seems more likely that the user is interested in one of the $\mathcal{C}_q$ objects that have the tag '*kitten*' than the ones with the '*animal*' tag. To model this effect, we bias the probability that an object is of interest by the ratio $|A_q(t)|/|\mathcal{C}_t|$. In our example, the ratio is $900/1000$ for a '*kitten*' object, and $100/1000$ for a '*animal*' object. □

In this extended model, the probability that an object in $A_q(t)$ is the one the user is interested in is: $p + (r-1) \cdot \frac{|A_q(t)|}{|\mathcal{C}_t|} \cdot p$. That

is, the likelihood ranges (linearly) from rp when $A_q(t) = \mathcal{C}_t$ to p when $|A_q(t)|/|\mathcal{C}_t|$ is near 0.

EXAMPLE 9. Say $r = 2$, $\mathcal{C}_q$ has 10 objects, $|A_q(t_1)| = |A_q(t_2)| = 3$, and $|\mathcal{C}_{t_1}| = |\mathcal{C}_{t_2}| = 10$; Then each of the t_1 (and t_2) objects has a likelihood of $p + 0.3 \cdot (r-1) \cdot p = 1.3p$ of being interesting for the user. Since the probability over all objects is 1, then $6 \cdot (1.3p) + (10-6) \cdot p = 1 \Rightarrow p = 1/11.8 = 0.085$. So the probability of failure is $\mathcal{F} = (10-6) \cdot p = 0.34$. Recall that when all objects for t_1 and t_2 were relevant ($A_q(t_i) = \mathcal{C}_{t_i}$), then the probability of failure was 0.25, as in Example 7. Thus, in our current example the user is less likely to find the object of interest because the tags in the cloud are not that relevant to the query q. □

Summary: We still assume that overlap is 0 and that $s(q, c) = 1$ for all $c \in \mathcal{C}_q$, but now incorporate coverage and relevance. For an object $c \in A_q(t)$ the probability that it is of the user's interest is $p + (r-1) \cdot \frac{|A_q(t)|}{|\mathcal{C}_t|} \cdot p$ and for every object $c \in \mathcal{C}_q - \bigcup_{t \in \mathcal{S}} A_q(t)$ the probability that it is of the user's interest is p.

5.3 Incorporating cohesiveness

To motivate the usage of cohesiveness in our user model we give an example.

EXAMPLE 10. Say we have $\mathcal{S}$ with two tags, t_1 and t_2, where:
- $|A_q(t_1)| = 3$, $|A_q(t_2)| = 3$;
- $|\mathcal{C}_{t_1}| = 3$, $|\mathcal{C}_{t_2}| = 3$ (both tags highly relevant); and
- $|\mathcal{C}_q| = 10$.

Our model so far says that objects in $A_q(t_1)$ and $A_q(t_2)$ are equally likely to be of interest. However, say that t_1 is very cohesive (Section 3) and t_2 is not. □

We now take this difference into account.

One way to model this difference in cohesiveness is to say that when a tag t is not cohesive, some of the objects in $A_q(t)$ will not be of interest to the user, because they are not really related to the "topic" that is of interest to the user (whatever that might be when he clicked on the appropriate tag in the tag cloud). So, for example, if t_1 = '*Microsoft*' and all the objects in $A_q(t_1)$ are closely related to each other (very high cohesiveness), then we can assume that all objects in $A_q(t_1)$ have a probability rp of being the ones of interest. If t_2 = '*Jaguar*' (which has several meanings) and cohesiveness is say $1/3$, then we can say that only $1/3$ of the objects in $A_q(t_2)$ have rp probability, while the rest have probability p (like any object in $\mathcal{C}_q - O$). Of course, we do not know precisely how cohesiveness affects the fraction of $A_q(t_2)$ objects that is more likely to be of interest, but in the interest of simplicity we assume a linear dependence.

To take both cohesiveness and relevance into account, we first compute the probabilities of objects in $A_q(t)$ as done for the relevance model, and then based on cohesiveness we move the right fraction to a p probability. In Example 10 above, we had 6 objects with probability $1.3p$. Say the three corresponding to t_1 have cohesiveness 1, while the three for t_2 have cohesiveness $1/3$. Then we only have $3 \cdot 1.3p + 1 \cdot 1.3p + (4+2)p = 1$ so $p = 1/(4 \cdot 1.3 + 6) = 0.089$. The probability of failure is $4p = 0.36$.

Recall that $\overline{sim}(t)$ is the intra-similarity of the objects in $A_q(t)$ (Section 3). Thus, the probability of the user being interested in an object c in $A_q(t)$ is:
- for $\overline{sim}(t) \cdot |A_q(t)|$ of the objects in $A_q(t)$: $p + (r-1) \cdot p \cdot \frac{|A_q(t)|}{|\mathcal{C}_t|}$
- for the rest objects in $A_q(t)$: p.

Thus, the probability of an "average" object in $A_q(t)$ being interesting is: $p + (r-1) \cdot \overline{sim}(t) \cdot \frac{|A_q(t)|}{|\mathcal{C}_t|} \cdot p$.

Summary: We assume that overlap is 0 and that $s(q, c) = 1$ for all $c \in \mathcal{C}_q$. Our current model incorporates coverage, relevance, and cohesiveness. For an object $c \in A_q(t)$ the probability that it is of the user's interest is $p + (r-1) \cdot \overline{sim}(t) \cdot \frac{|A_q(t)|}{|\mathcal{C}_t|} \cdot p$ and for every object $c \in \mathcal{C}_q - \bigcup_{t \in \mathcal{S}} A_q(t)$ the probability that it is of the user's interest is p.

5.4 Incorporating Overlap

The last measure we incorporate into the model is overlap. Say the object c appears in multiple association sets, $A_q(t_1)$, $A_q(t_2)$, ..., $A_q(t_l)$ and no other. After we take into account cohesiveness and relevance, we get from t_1 that c has probability $p_1 = p + (r-1) \cdot \overline{sim}(t_1) \cdot \frac{|A_q(t_1)|}{|\mathcal{C}_{t_1}|} \cdot p$, from t_2 probability $p_2 = p + (r-1) \cdot \overline{sim}(t_2) \cdot \frac{|A_q(t_2)|}{|\mathcal{C}_{t_2}|} \cdot p$ etc. We assume the object will *not* be of interest if it is *not* of interest as each tag is considered, i.e., with probability $(1-p_1) \cdot (1-p_2) \cdot \ldots (1-p_l)$. Hence, the probability the object is of interest is $1 - (1-p_1) \cdot (1-p_2) \cdot \ldots (1-p_l)$.

In general, the probability of an object c that appears in $A_q(t_1)$, ..., $A_q(t_l)$ will be:

$$1 - \prod_{i=1}^{l} \left(1 - \left(p + \overline{sim}(t_i) \cdot (r-1) \cdot p \cdot \frac{|A_q(t_i)|}{|\mathcal{C}_{t_i}|}\right)\right) \quad (1)$$

Summary: We assume that $s(q, c) = 1$ for all $c \in \mathcal{C}_q$. Our current model incorporates coverage, relevance, cohesiveness and overlap. For an object c that is contained by $A_q(t_1)$, $A_q(t_2)$, ..., $A_q(t_l)$ and no other association sets the probability that it is of the user's interest is the one that can be seen in Equation 1 and for every object $c \in \mathcal{C}_q - \bigcup_{t \in \mathcal{S}} A_q(t)$ the probability that it is of the user's interest is p.

5.5 Taking into account scores

So far we have not taken into account the relevance of an object c in $\mathcal{C}_q$ to the query q (i.e., $s(q, c)$). We incorporate these scores as follows: After we determine for each object c its probability of being of interest to the user, $\Pr[c]$, we determine the final probability $\Pr[c]'$ by first multiplying every $\Pr[c]$ with $s(q, c)$ and then normalizing in order to have $\sum_{c \in \mathcal{C}_q} \Pr[c]' = 1$. Thus, we have:

$$\Pr[c]' = \frac{\Pr[c] \cdot s(q, c)}{\sum_{c \in \mathcal{C}_q} \{\Pr[c] \cdot s(q, c)\}}$$

The values $\Pr[c]'$ represent the final probability distribution, which is used to determine the failure probability of the given tag cloud.

This addition can be applied to any of the models we described so far. In our experiments though, we are going to include the final model that incorporates coverage, relevance, cohesiveness, and overlap as well as the scores.

5.6 Closing Comment

Note that our final model does not take into account extent, independence, popularity and balance. Extent is parameter that the user selects for the tag clouds he evaluates; thus the user model does not have to take this metric under account explicitly. Independence seems to be taking a very specific range of values in practice (higher than 0.8 almost always); also, it is highly correlated to overlap for the CourseRank dataset. Thus, we think it is acceptable to only consider the overlap metric. Popularity is extremely highly correlated to the coverage metric (more than 0.9 for both datasets), thus we can exclude it from our model with no harm. Finally, balance does not affect the failure metric we have proposed. Rather, balance affects a different metric that counts how many iterations we need before the user narrows down the search to the object of interest. (Each iteration involves clicking on a tag in the cloud, and

examining a new cloud that describes the objects that have all of the selected tags so far.) The more balanced a cloud is, on average it will take fewer iterations to converge on the object(s) of interest. The number of iterations metric is even harder to evaluate than failure probability since it involves evaluating multiple clouds at once. We do not consider this metric in this paper.

6. EXPERIMENTAL EVALUATION

In this section, we present experimental results that help us understand how our tag metrics relate to each other, and show how the tag selection algorithms perform. Based on our results, we also provide guidelines of how a tag selection algorithm should be devised. We also evaluate how easy the selection of the best tag cloud is for users and whether our user model can actually help predict the best tag cloud. We use two datasets for our evaluation.

• CourseRank: The first dataset is a set of 18,774 courses that Stanford University offers annually. Each course represents an object in $\mathcal{C}$. We consider all the words in the course title, course description and user comments to be the tags for the object, except for stop words and terms such as 'prerequisite' that are very frequent and do not describe the content of the course.

• `del.icio.us`: The second dataset consists of urls and the tags that the users assigned to them on `del.icio.us`. Our dataset contains a month worth of data (May 25, 2007–June 26, 2007); we randomly sampled 100,000 urls and all the tags that were assigned to these 100,000 urls during that month.

Table 1: Dataset statistics

	Delicious	CourseRank
Number of distinct tags	69,858	107,530
Number of distinct objects	100,000	18.392
Number of all pairs (tag, object)	403,948	11,669,038
Number of unique pairs (tag, object)	324,730	686,443

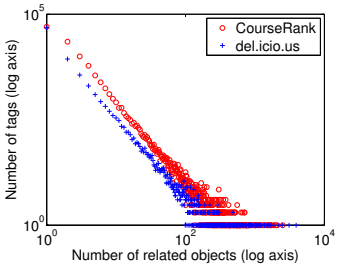


Figure 1: Distribution of tags across objects related to the tags

Table 1 summarizes the dataset statistics. In order to understand some key characteristics of our datasets, we consider Figure 1, which shows the distribution of tags across the number of related objects to the tags. We observe that CourseRank is more dense than `del.icio.us`: the number of tags that have a particular number of related objects ($|A_q(t)|$) is higher for CourseRank than for `del.icio.us` in the top left of the figure. On the other hand, there are some tags that have been assigned to very large numbers of objects for the `del.icio.us` dataset comparing to the CourseRank dataset (see bottom right of the figure). This means that in `del.icio.us` there are tags whose association sets are larger than any association set in CourseRank.

To emulate queries submitted by users, we select 30 tags from each dataset representing a spectrum of associated sets of objects of different size. For instance, the query '*databases*' in the CourseRank dataset returns 46 courses, while the query '*history*' returns 2,073. We refer to our 30 queries as the set $\mathcal{Q}$ for each dataset. As scoring function $s(t, c)$, we used the tf-idf scoring function between tags and objects. We defined the similarity between objects c_1 and c_2, $sim(c_1, c_2)$, as the Jaccard similarity between the set of tags assigned to c_1 and the set of tags assigned to c_2.

Our experimental results are organized as main findings. For each finding we show representative results. We compare the popularity algorithm (`POP`), the plain tf-idf algorithm (`TF`), the weighted tf-idf algorithm (`WTF`) and the maximum covering algorithm (`COV`).

6.1 Most metrics are not correlated

When designing a tag selection algorithm, it is important to understand how different metrics may correlate to each other in order to determine which independent metrics make sense as optimization objectives. In addition, it is possible that metrics may exhibit different correlation trends in different datasets.

Our key finding is that only coverage and popularity are highly correlated between all pairs of our metrics, while most pairs of metrics are not strongly correlated. For example, a tag cloud $\mathcal{S}$ that has high coverage does not necessarily have high relevance. Some exceptions for the `del.icio.us` dataset are:

• Clouds with high coverage tend to have high popularity and the vice versa (correlation is 0.98); and

• Clouds with high cohesiveness tend to have *low* independence and vice versa (correlation is -0.57).

Performing the same experiment on the CourseRank dataset gave different results. For example, overlap and independence were very highly correlated as well as overlap and balance.

These results have implications for the design of tag selection algorithms for tag clouds. In particular, if for a given query one is selecting a cloud that does well under one metric, it is not important to also try to improve a correlated metric. For example, it is probably wasteful to jointly search for a cloud with high coverage *and* high popularity, since if we achieve one of these goals we are likely to achieve the other. On the other hand, it may make sense to jointly optimize for high popularity and say high relevance, since each metric looks at an unrelated aspect of the cloud. Furthermore, the performance of a tag selection algorithm may depend on the characteristics of a dataset. For example, in the case of CourseRank, an algorithm that tries to achieve a high value for balance is bound to end up with high overlap as a side-effect, while in the case of `del.icio.us` this undesired dependence does not exist.

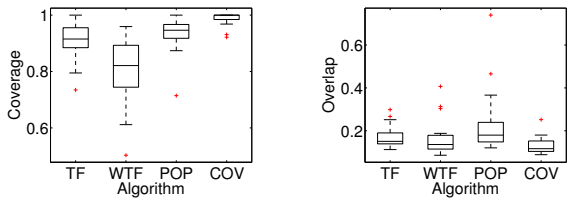
6.2 The algorithms impact metrics differently

The algorithms of Section 4 have a single goal in mind, e.g., high popularity of the resulting cloud, but as a side effect impact the metrics quite differently. To understand their impact, we ran each algorithm on each of our queries in $\mathcal{Q}$ for both datasets and evaluated all the metrics we have defined.

Figure 6.2 shows the results for the CourseRank data in box/whiskers plots. Each figure presents the impact on one metric. For example, Figure 2(a) shows the coverage of the resulting clouds, for each algorithm (horizontal axis). For each algorithm, the bottom of the box is the first quartile (25^{th} percentile), the top of the box is the third quartile (75^{th} percentile), the line in the middle of the box is the median (50^{th} percentile), while the top bar represents the maximum value observed and the bottom bar the lowest. There are also some outlier points that are represented as red plus-shaped points. (For all graphs, the clouds have 30 tags.)

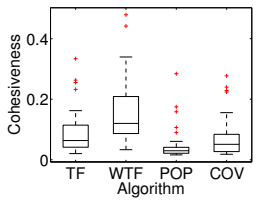
We see that no algorithm completely dominates the other algorithms. Some interesting conclusions include the following:

• The `POP` algorithm of course maximizes the popularity metric (the whole box/whiskers plot is on 1 in Figure 2(f)), but it also does very well with respect to balance, much better than the competition. That is, for the CourseRank dataset, popular tags do not seem to have association sets of very different cardinalities. On the other

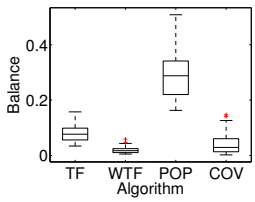


(a) Coverage

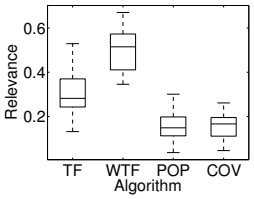
(b) Overlap



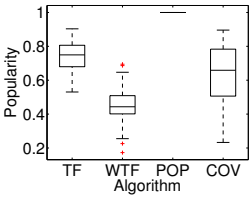
(c) Cohesiveness



(d) Balance



(e) Relevance



(f) Popularity

Figure 2: Box/whiskers plots for various metrics vs. algorithms for the CourseRank dataset (extent = 30)

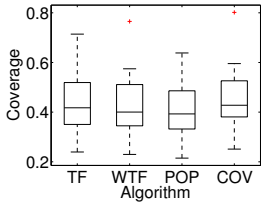


Figure 3: Box/whiskers plot of Coverage for all algorithms (`del.icio.us` dataset)

hand, POP divides query results into association sets $A_q(t)$ that are not highly cohesive or relevant to the query. This observation seems to provide a quantitative explanation for the results of the user study in [8], where users gave low ratings to popular tag clouds as a means for summarizing and refining search results.

- The WTF algorithm does well in relevance and cohesiveness, but is outperformed by the competition for the remaining metrics. In particular, it tends to generate highly imbalanced tag clouds. On the other hand, TF seems to strike a middle ground, since it does not achieve as good cohesiveness and relevance as WTF but still it is better than the other algorithms, and in addition, it performs better than WTF with respect to most of the other metrics.
- All algorithms yield similar overlap, although popularity does a bit worse. Apparently, there are very popular tags that share many objects in common (e.g., '*history*' and '*art*') but their impact is not significant.
- Algorithms POP and COV have similar performance for many metrics, which is expected given that they target correlated metrics. However, there are some marked differences. For instance, POP yields decent coverage, while COV does not perform similarly well with respect to popularity.

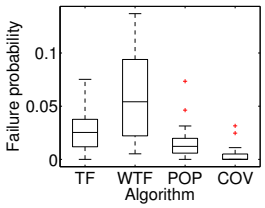
When we ran the same algorithms on the `del.icio.us` dataset, we got similar results to ones above for the CourseRank dataset. The only practical difference exists in the coverage metric. In the CourseRank dataset, we can see that COV gives the highest coverage, then POP and TF and finally WTF follows in the end. In the `del.icio.us` dataset (Figure 3), the overall range of values for the coverage metric is around 0.2–0.8, which is much lower than the same range for the CourseRank dataset. This fact may be attributed to the characteristics of each dataset. Recall that the CourseRank dataset is denser than the `del.icio.us` dataset (Figure 1).

Since the four algorithms exhibit different behavior, they are good for different settings. POP is good for showing tag usage (for example in social bookmarking sites, such as `del.icio.us`) but the generated tag clouds do not have good cohesiveness and relevance properties. This observation provides a quantitative explanation of user study results showing that popular tag clouds do not serve well query result summarization and refinement [8]. On the other hand, the WTF algorithm does well with respect to the relevance and cohesiveness metrics, but does not perform well with respect to other metrics, in particular coverage. In this set of experiments, we have fixed the extent of the tag clouds to 30. One interesting extension for the WTF algorithm would be to try to find the minimum number of relevant and cohesive tags required in order to achieve satisfying coverage.

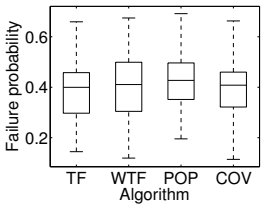
6.3 Algorithms impact failure probability differently

The user model of Section 5 represents a user that is exploring the available objects via the tag cloud. The lower the failure probability is, the more likely it is that one of the displayed tags will lead to an object of interest. To see how the algorithms perform for such a user, we evaluate the tag clouds they generate for queries in $\mathcal{Q}$ using the failure probability (using the full user model).

For this experiment we used both datasets. We gathered the failure probability ($\mathcal{F}$) values for various values of r and extent of our tag clouds and represented $\mathcal{F}$ in box/whiskers plots.



(a) CourseRank



(b) `del.icio.us`

Figure 4: Failure probability ($r = 10$) vs. algorithms ($|\mathcal{S}| = 30$)

Figure 4(a) shows a representative result for the CourseRank dataset. In this case, we have set model parameter r to 10 and the tag clouds contain 30 tags each. (Recall that as r grows, the objects covered by the tag cloud are more likely to be of interest.) In the figure we see clearly that COV performs the best; not only are the failure values very low, but the variability is also low. The next algorithm is POP, followed by TF, followed by WTF.

For the `del.icio.us` dataset, though, the situation is different. For the same parameters (extent $= 30$, $r = 10$) we can see in Figure 4(b) that the performance of all algorithms is much closer. There is no clear ordering between the algorithms, the failure probability values are much larger, and their variability is much larger.

Why the large differences between datasets? The key differences are that there are many fewer tags per object in `del.icio.us`, as compared to CourseRank (Figure 1), and that the `del.icio.us` tags cover many fewer objects. Thus, there are fewer choices avail-

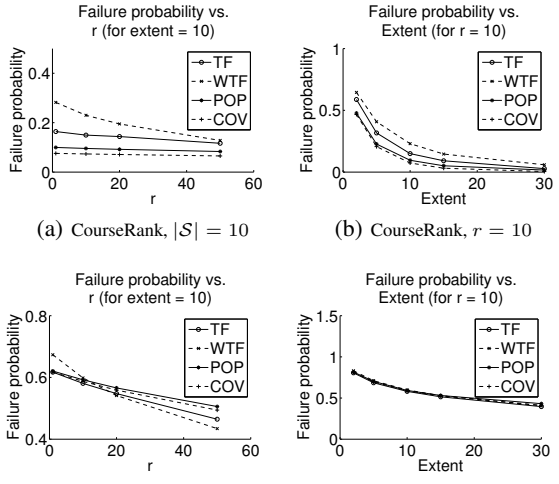
able when selecting tags for del.icio.us, and these choices tend to be not that good in terms of coverage and some of the other metrics. Thus, there are not many "smart" choices to make, and all algorithms tend to perform equally poorly.

On the other hand, for CourseRank, there are plenty of high coverage tags that can easily describe the results in $\mathcal{C}_q$. Furthermore, these very high coverage tags also appear to do well in terms of relevance and cohesiveness (compared to the results obtained in the case of del.icio.us), so COV does best overall.

One can construct datasets where the tags that cover the most have huge overlap and very poor cohesiveness and relevance, and in such scenarios the COV algorithm will not do as well. However, we do not think such sets would occur naturally. Thus, for the CourseRank dataset, it does not seem necessary to develop an algorithm that optimizes coverage jointly with some other metric, as is done by [3]. Such a joint optimization algorithm could try, for instance, to find a cloud that has both high coverage and high cohesiveness. However, given that high cohesiveness by itself does not lead to low failure probability (else WTF would do well), it seems unlikely that sacrificing any coverage for cohesiveness (or any other metric) will pay off. We have experimented with joint optimization algorithms, and did not find any that outperformed the simple COV algorithm.

6.4 How the ordering of the algorithms change for different extent and r values

We have found that in general the ordering of our algorithms does not significantly change due to changes of r or the extent. For instance, in Figure 5(a) we see how the average failure probability varies as a function of r and the extent size for the CourseRank dataset. As expected, failure probability decreases as each of these parameters increase, but more interestingly the ordering of the algorithms stays the same (first COV, then POP, TF and WTF). Same results were observed for other values of extent and r.



(a) CourseRank, $|\mathcal{S}| = 10$ (b) CourseRank, $r = 10$

(c) del.icio.us, $|\mathcal{S}| = 10$ (d) del.icio.us, $r = 10$

Figure 5: Sensitivity of our datasets for extent and r

The equivalent results for the del.icio.us dataset are different, as illustrated by Figure 5(c). Now, when r is less than about 15, the ordering (best to worst) is TF, COV, POP and finally WTF. However, as r increases WTF becomes the best. The reason is twofold. First, as r increases, cohesiveness and relevance play a more important role in our user model. Second, WTF is able to achieve high cohesiveness because in this data set there is a significant number of objects sharing many tags (and hence have high similarity). For instance, it is common for two blogs, irrespectively of their content, to have been labeled by users with overlapping tags: '*blog*', '*blogs*', '*web*', '*web2.0*', '*design*', '*news*'; this is not true though for the course descriptions that form the CourseRank tags.

Note that for the del.icio.us dataset the differences between the algorithms are very small; the difference between the maximum and minimum failure probability for given extent and r is no more than 0.1. In contrast, for the CourseRank dataset the same difference is up to 0.25. Hence, for del.icio.us, one may select the simplest algorithm to implement (i.e., POP) because the differences among algorithms are not great. However, for CourseRank one may opt for COV because the extra implementation (and run time) cost does pay off.

6.5 Evaluating the User Model

In this experiment we see if the failure probabilities our synthetic model are indeed useful for identifying clouds users prefer. We focus on the del.icio.us dataset, mainly because it does not require domain-specific knowledge (as the CourseRank dataset requires for some queries). We used the set of queries $\mathcal{Q}$ and generated tag clouds using the algorithms described in Section 4 for extent equal to 10. (We found that using values larger than 10 made the process very expensive and slow, because of the excessive amount of work that was required by the evaluators/workers.) We started transforming these tag clouds by swapping tags that belonged in the tag clouds with other, random ones (but of good quality, according to their aggregate tf-idf and coverage) that did not belong to the tag cloud. These transformations *always* increased the failure probability of the original tag clouds produced by the algorithms of Section 4.

We generated 150 random pairs of tag clouds for our 30 queries ($\mathcal{Q}$), so that the number of pairs of tag clouds that had a difference of their failure probabilities in each of the intervals $[0, 0.5)$, $[0.5, 0.1)$, $[0.1, 0.125)$, $[0.125, 0.15)$, or $[0.15, 0.25)$ was 30. We then asked evaluators to compare the pairs of tag clouds for the queries; each pair of tag clouds was compared by 5 different workers in Mechanical Turk (www.mturk.com).

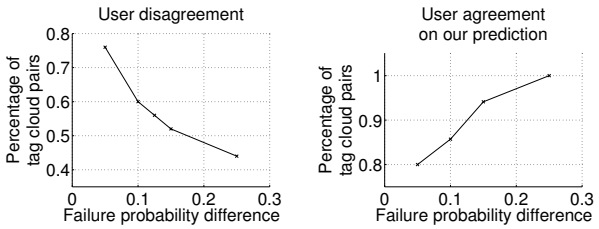
We first tried to explore how often workers agreed on one of the two tag clouds they evaluated. Suppose that a worker had to choose for a query q which of the tag clouds $\mathcal{S}_1$ or $\mathcal{S}_2$ is better. If 4 or more workers believe that $\mathcal{S}_1$ (or $\mathcal{S}_2$) is better than $\mathcal{S}_2$ (or $\mathcal{S}_1$), we say that workers have *agreed* that $\mathcal{S}_1$ (or $\mathcal{S}_2$) is better.

Figure 6(a) shows how often there is *no* agreement in a particular tag cloud (i.e., 3 workers voted for one tag cloud and 2 workers voted for the other tag cloud). The x-axis is the difference in the failure probabilities in pairs of tag clouds; the y-axis is the percentage (out of the 30 pairs of tag clouds that belong in the same interval) of pairs of tag clouds for which there was no agreement. For example, the point ($x = 0.25$, $y = 0.44$) indicates that for cloud pairs with a failure probability in the range 0.15 to 0.25, users did not agree in 44% of the cases. We see that when the failure probabilities are close to each other (small differences), there is rarely an agreement (around 25% of all pairs of tag clouds!). As the difference in failure probabilities increases, there is a clear tendency for at least four out of five users to agree on one of the two tag clouds.

The results of Figure 6(a) illustrate why it is in general hard to evaluate tag clouds via user studies. That is, in many cases users do not have a clear preference among clouds, probably because they have different backgrounds or different interpretations for the search query. However, our failure metric lets us identify cases where there is likely to be agreement among evaluators. We can hence focus in finding good tag clouds for those cases.

We now report on how well our model predicts the users' choices. Using the same user evaluation as before (150 pairs of tag clouds),

we examined how often our user model predicted the tag cloud the users preferred. We present the results in Figure 6(b). The x-axis is the difference in failure probability in a pair of tag clouds, and the y-axis shows the percentage of the queries for which our model predicted the best tag cloud out of those pairs that workers came to an agreement. We can see that we predict correctly for 80% of the pairs, even when the failure probability differences are small; our prediction goes up to 100% for failure probability differences in 0.15–0.25. This means that, if the users can reach an agreement, our model will very often predict the preferred tag cloud.



(a) No agreement on a tag cloud (b) Agreement on our prediction

Figure 6: Can good tag clouds be selected?

Our experiments suggest that users often do not agree on what constitutes a good tag cloud, especially if the clouds in question have similar failure probabilities. However, when there is agreement among users, users clearly tend to prefer the cloud with smallest failure probability. These results suggest that our synthetic user model does capture the preferences of real users, and hence is a useful tool for comparing tag selection algorithms.

7. RELATED WORK

We now briefly discuss related work organized into three groups.

Clustering search results: Clustering can be viewed as a sub-case of our problem where coverage must be 1, overlap must be 0 and the number of clusters (extent) is up to a predetermined number. There has been extensive research on clustering search results [5, 7, 10, 12, 17, 18], and real search engines (like `clusty.com`) use clustering. There is also work on query results labeling [11] and categorizing results of SQL queries [4].

Tag cloud research: Research on tag clouds has mostly focused on presentation and layout aspects of tag clouds [6, 13]. For selecting tags to be displayed in a tag cloud, social information sharing sites mostly use popularity-based schemes. Recently, tag selection algorithms that try to find good, not necessarily popular, tags have been developed for structured data [8].

Problems related to the metrics we have defined: The most related problem to our metrics is a recently proposed problem called Topical Query Decomposition [3]. In this problem, one is given a query and a document retrieval system and wants to produce a small set of queries whose union of resulting documents corresponds approximately to that of the original query. The original query in this problem could be our query q and the small set of queries produced by the algorithms they describe is the small set of tags that our tag cloud will contain. Obviously, the setting in this problem is different from the problem in our case. Reference [3] deals with search engine query logs, trying to decompose one query into other ones; we are trying to decompose our query results into a set of tags in a tag cloud. Nevertheless, reference [3] uses coverage, overlap, cohesiveness and relevance; the definitions used for these metrics are similar to ours. An optimization of [3] was given in [14].

8. CONCLUSIONS

Our goal has been to understand the process of building a tag cloud for exploring and understanding a set of objects. To that end, we presented a formal system model for reasoning about tag clouds and we defined a set of metrics that capture the structural properties of a tag cloud. We evaluated a set of existing algorithms using an ideal user model, which we also evaluated. We concluded, among other things, that:

• Our metrics are not generally correlated to each other. Thus, they capture different important aspects of the tag cloud, and are relevant for studying clouds.

• Under our user model, our maximum coverage algorithm, `COV` seems to be a very good choice for most scenarios we studied — most often it is the best choice. Another good choice is the popularity algorithm `POP` which is easier to implement and performs well in specific contexts, especially when relevance and cohesiveness are not critical.

• Our user model is a useful tool for identifying tag clouds preferred by users.

There is still a large agenda of issues to tackle. Firstly, we would like to extend our model to capture the balance metric and thus hierarchical search using tag clouds. Secondly, we would like to construct an algorithm that would minimize the failure probability for a particular dataset and a given extent. Thirdly, we would like to take into account items with no assigned tags and items where malicious (spam) tags have been assigned.

9. REFERENCES

[1] Courserank. http://www.courserank.com.

[2] Search cloudlet. http://www.getcloudlet.com/.

[3] F. Bonchi, C. Castillo, D. Donato, and A. Gionis. Topical query decomposition. In *KDD*, pages 52–60, 2008.

[4] K. Chakrabarti, S. Chaudhuri, and S.-w. Hwang. Automatic categorization of query results. In *SIGMOD*, pages 755–766, 2004.

[5] F. Gelgi, H. Davulcu, and S. Vadrevu. Term ranking for clustering web search results. In *WebDB*, 2007.

[6] J. Good, C. Shergold, M. Gheorghiu, and J. Davies. Using tag clouds to facilitate search: An evaluation, 1997.

[7] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, pages 76–84, 1996.

[8] G. Koutrika, Z. M. Zadeh, and H. Garcia-Molina. Data clouds: summarizing keyword search results over structured data. In *EDBT*, pages 391–402, 2009.

[9] B. Kuo, T. Hentrich, B. M. Good, and M. Wilkinson. Tag clouds for summarizing web search results. In *WWW*, pages 1203–1204, 2007.

[10] I. Masowska. Phrase-based hierarchical clustering of web search results. In *ECIR*, pages 555–562, 2003.

[11] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, 2000.

[12] S. Osinski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Inf. Sys.*, pages 359–368, 2004.

[13] A. W. Rivadeneira, D. M. Gruen, M. J. Muller, and D. R. Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *CHI*, 2007.

[14] M. Sydow, F. Bonchi, C. Castillo, and D. Donato. Optimising topical query decomposition. In *WSCD*, pages 43–47, 2009.

[15] V. V. Vazirani. *Approximation Algorithms*. Springer, March 2004.

[16] Wikipedia. Tag cloud — wikipedia, the free encyclopedia, 2010. [Online; accessed 27-February-2010].

[17] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. *Computer Networks*, 31(11–16):1361–1374, 1999.

[18] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR*, pages 210–217, 2004.