Clustering the Tagged Web

Daniel Ramage
353 Serra Mall
Stanford, CA, USA
dramage@cs.stanford.edu

Paul Heymann
353 Serra Mall
Stanford, CA, USA
heymann@cs.stanford.edu

Christopher D. Manning
353 Serra Mall
Stanford, CA, USA
manning@cs.stanford.edu

Hector Garcia-Molina
353 Serra Mall
Stanford, CA, USA
hector@cs.stanford.edu

ABSTRACT

Automatically clustering web pages into semantic groups promises improved search and browsing on the web. In this paper, we demonstrate how user-generated tags from large-scale social bookmarking websites such as del.icio.us can be used as a complementary data source to page text and anchor text for improving automatic clustering of web pages. This paper explores the use of tags in 1) K-means clustering in an extended vector space model that includes tags as well as page text and 2) a novel generative clustering algorithm based on latent Dirichlet allocation that jointly models text and tags. We evaluate the models by comparing their output to an established web directory. We find that the naive inclusion of tagging data improves cluster quality versus page text alone, but a more principled inclusion can substantially improve the quality of all models with a statistically significant absolute F-score increase of 4%. The generative model outperforms K-means with another 8% F-score increase.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*clustering*; H.1.2 [**Models and Principles**]: User/Machine Systems—*human information processing*

General Terms

Algorithms, Experimentation, Human Factors, Measurement

1. INTRODUCTION

Modern web search engines are tasked with returning the few most relevant results based on an often ambiguous user query and billions of web documents. Over ten years, ranking techniques harnessing link, anchor text, and user click-through data as well as simply page text have been developed to address this challenge. However, a major challenge is the inherent ambiguity of the user query. These queries are rarely more than a few words in length and may represent many potential information needs. One of the most promising (and common) approaches to handle this ambiguity is through automatic clustering of web pages.

The usefulness of clustering for resolving this ambiguity relies on the *cluster hypothesis* [32]: "the associations between documents convey information about the relevance of documents to requests." Work by Voorhees [34] and Hearst [20] has suggested that the cluster hypothesis is true. There have been a number of successful applications of the hypothesis, including search result clustering [39]; alternative user interfaces [12]; document retrieval using topic-driven language models [25, 35]; and improved information presentation for browsing [27]. Topics may also be associated with temporal trends; for example, users might search for "sleep medication" at night, and "caffeine" during the day [5].

Viewed in a different light, the cluster hypothesis can be seen as a way to increase *diversity* in search results. Diversity is the extent to which the results returned by a search engine pertain to different information needs. If an ambiguous user query can have many meanings, a search engine can assume that documents from different topical clusters represent different information needs. Thus, an engine which returns results from many topical clusters is likely to have greater diversity. Recent work by Song et al. [30] has suggested that this is a reasonable way to increase diversity, especially in the highly ambiguous world of image retrieval.

This paper addresses one central question: How can tagging data be used to improve web document clustering? This is part of a major trend in information retrieval to make more and better use of (the increasingly prevalent) user-provided data. Social bookmarking websites such as del.icio.us and StumbleUpon enable users to tag any web page with short free-form text strings, collecting hundreds of thousands of keyword annotations per day [21]. The set of tags applied to a document is an explicit set of keywords that users have found appropriate for categorizing that document within their own filing system. Thus tags promise a uniquely well suited source of information on the similarity between web documents. While others have argued that tags hold promise for ranked retrieval, [3, 23, 36], including at least one approach that uses clustering [40], this paper is the first to systematically evaluate how best to use tags for the important task of clustering documents on the web. High quality clustering based on user-contributed tags has the potential to improve all of the previously stated applications of the cluster hypothesis, from user interfaces to topic-driven language models to increasing diversity of results.

This paper makes the following contributions. In Section 3.1, we show significant gains in the quality of automatic clustering of web documents by the K-means algorithm when it is also provided tagging data. In Section 3.2, we show that tags are different from "just more" page text by demonstrating that their naive inclusion fails to achieve the full extent of these gains. We then present Multi-Multinomial LDA, an extension of the latent Dirichlet allocation (LDA) clustering algorithm in Section 4 that explicitly models text and tags, significantly outperforming K-means on a broad clustering task. In Section 5.1, we consider whether tags are a qualitatively different type of annotation than the anchor text of backlinks. We conclude that the benefits of including tagging data still stand with the inclusion of anchor text. In Section 5.2, we look at whether tagging only helps for clustering a large collection of general documents, or whether it can help a more specific collection (e.g., documents having to do with programming). We find that tagging data is even more effective for more specific collections. Finally, we conclude with a discussion of tagging data's implications for information retrieval in document clustering and beyond.

2. PROBLEM STATEMENT

Our goal is to determine how tagging data can best be used to improve web document clustering. However, clustering algorithms are difficult to evaluate. Manual evaluations of cluster quality are time consuming and usually not well suited for comparing across many different algorithms or settings [18]. Several previous studies instead use an automated evaluation metric based on comparing an algorithm's output with a hierarchical web directory [31, 28]. Such evaluations are driven by the intuition that web directories, by their construction, embody a "consensus clustering" agreed upon by many people as a coherent grouping of web documents. Hence, better clusters are generated by algorithms whose output more closely agrees with a web directory. Here, we utilize a web directory as a gold standard so that we can draw quantitative conclusions about how to best incorporate tagging data in an automatic web clustering system.

We define the *web document clustering task* as follows:

1. Given a set of documents with both words and tags (defined in Section 2.4), partition the documents into groups (*clusters*) using a candidate *clustering algorithm* (defined in Section 2.1).
2. Create a gold standard (defined in Section 2.2) to compare against by utilizing a web directory.
3. Compare the groups produced by the clustering algorithm to the gold standard groups in the web directory, using an evaluation metric (defined in Section 2.3).

This setup gives us scores according to our evaluation metric that allow us to compare candidate clustering algorithms. We do not assert that the gold standard is the best way to organize the web—indeed there are many relevant groupings in a social bookmarking website necessarily lost in any coarser clustering. However, we argue that the algorithm which is best at the *web document clustering task* is the best algorithm for incorporating tagging data for clustering.

2.1 Clustering Algorithm

A web document clustering algorithm partitions a set of web documents into groups of similar documents. We call the groups of similar documents *clusters*. In this paper, we look at a series of clustering algorithms, each of which has the following input and output:

Input A target number of clusters K, and a set of documents numbered $1, \dots, D$. Each document consists of a bag of words from a word vocabulary W and a bag of tags from a tag vocabulary T.

Output An assignment of documents to clusters. The assignment is represented as a mapping from each document to a particular cluster $z \in 1, \dots, K$.

This setup is similar to a standard document clustering task, except each document has tags as well as words.

Two notable decisions are implicit in our clustering algorithm definition. First, many clustering algorithms make *soft* rather than *hard* assignments. With hard assignments, every document is a member of one and only one cluster. Soft assignments allow for degrees of membership and membership in multiple clusters. For algorithms that output soft assignments, we map the soft assignments to hard assignments by selecting the single most likely cluster for that document. Secondly, our output is a flat set of clusters. In this paper, we focus on flat (nonhierarchical) clustering algorithms rather than hierarchical clustering algorithms. The former tend to be $O(kn)$ while the latter tend to be $O(n^2)$ or $O(n^3)$ (see Zamir and Etzioni [38] for a broader discussion in the context of the web). Since our goal is to scale to huge document collections, we focus on flat clustering.

In our experiments, we look at two broad families of clustering algorithms. The first family is based on the vector space model (VSM), and specifically the K-means algorithm. K-means has the advantage of being simple to understand, efficient, and standard. The second family is based on a probabilistic model, and specifically derived from LDA. LDA-derived models have the potential to better model the data, though they may be more complicated to implement and slower (though not asymptotically).

2.2 Gold Standard: Open Directory Project

We derive gold standard clusters from the Open Directory Project (ODP) [1]. ODP is a free, user-maintained hierarchical web directory. Each node in the ODP hierarchy has a label (e.g., "Arts" or "Python") and a set of associated documents.[1] To derive a gold standard clustering from ODP, we first choose a node in the hierarchy: the root node (the default for our experiments), or "Programming Languages" and "Society" (for Section 5.2). We then treat each child and its descendants as a cluster. For example, say two children of the root node are "Arts" and "Business." Two of our clusters would then correspond to all documents associated with the "Arts" node and its descendants and all documents associated with the "Business" node and its descendants, respectively.

A gold standard clustering using ODP is thus defined by

[1] Documents can be associated with multiple nodes in the hierarchy, but this happens very rarely in our data. When we have to choose whether a document is attached to one node or another, we break ties randomly.

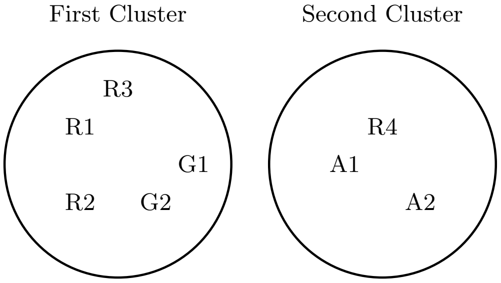

Figure 1: An example of clustering.

a particular node's K' children. When we give the clustering algorithm a value K, this is equal to the K' children of the selected node. In general, the best performing value of K will not be K'. This heuristic is adopted to simplify the parameter space and could be replaced by one of several means of parameter selection, including cross-validation on a development set. We sometimes use the labels in the hierarchy to refer to a cluster, but these labels are not used by the algorithms. When referring to the clusters derived from the gold standard, we will sometimes call these clusters *classes* rather than *clusters*. This is in order to help differentiate clusters generated by a candidate clustering algorithm and the clusters derived from the gold standard. It is also worth noting that the algorithms we consider are unsupervised and are therefore applicable to any collection of tagged documents (as opposed to documents which conform to the categories in ODP).

2.3 Cluster-F1 Evaluation Metric

We chose to compare the generated clusters with the clustering derived from ODP by using the F1 cluster evaluation measure [26]. Like the traditional F1 score in classification evaluation, the F1 cluster evaluation measure is the harmonic mean of precision and recall, where precision and recall here are computed over pairs of documents for which two label assignments either agree or disagree.

2.3.1 Example

Consider the example clustering shown in Figure 1. Two clusters are shown, and each document is denoted by its class in ODP: A for "Arts," G for "Games," R for "Recreation." A2 (for example) denotes a document which is in the ODP class "Arts" that the clustering algorithm has decided is in the second cluster. We think of pairs of documents as being either the same class or differing classes (according to our gold standard, ODP), and we think of the clustering algorithm as predicting whether any given pair has the same or differing cluster. The clustering in Figure 1 has predicted that $(A1, A2) \rightarrow \textit{same cluster}$ and that $(R2, R4) \rightarrow \textit{different cluster}$. If we enumerate all of the $\binom{n}{2} = 28$ pairs of documents in Figure 1, we get four cases:

True Positives (TP) The clustering algorithm placed the two documents in the pair into the same cluster, and our gold standard (ODP) has them in the same class. For example, $(R1, R3)$. There are 5 true positives.

False Positives (FP) The clustering algorithm placed the two documents in the pair into the same cluster, but our gold standard (ODP) has them in differing classes. For example, $(R1, G2)$. There are 8 false positives.

True Negatives (TN) The clustering algorithm placed the two documents in the pair into differing clusters, and our gold standard (ODP) has them in differing classes. For example, $(R2, A1)$. There are 12 true negatives.

False Negatives (FN) The clustering algorithm placed the two documents in the pair into differing clusters, and our gold standard (ODP) has them in the same class. For example, $(R2, R4)$. There are 3 false negatives.

We then calculate precision as $\frac{TP}{TP+FP} = \frac{5}{13}$, calculate recall as $\frac{TP}{TP+FN} = \frac{5}{8}$, and F1 as: $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \approx 0.476$.

2.3.2 Notes on F1

We selected F1 because it is widely understood and balances the need to place similar documents together while keeping dissimilar documents apart. We experimented with several other cluster evaluation metrics, including average pairwise precision, RAND index, and information theoretic measures such as pointwise mutual information and variation of information, finding the results to be consistent across measures.

F1 is a robust metric appropriate for our choice to provide the value K to our clustering algorithms (see Section 2.1). In particular, having the number of clusters K' in the gold-standard as input K does not ease the task of placing similar documents together while keeping dissimilar documents apart. Indeed, there may be many small, specific groupings of the top-level ODP categories—more than the 16 top-level subcategories—which a clustering algorithm would be forced to conflate. These conflations come at the expense of introducing false positives, possibly lowering the F1 score.

Because the clustering algorithms we consider are randomized, their output can vary between runs. To assign a stable F1 score to a particular algorithm, we report the mean F1 score across 10 runs of the algorithm with identical parameters but varying random initialization. In our experiments, we report statistical significance where appropriate. When we refer to a change in F1 score as significant, we mean that the variation between the underlying runs for two algorithms is significant at the 5% level by a two-sample t-test.

2.4 Dataset

Our tagged document collection is a subset of the Stanford Tag Crawl Dataset [21]. The Tag Crawl consists of one contiguous month of the *recent* feed on del.icio.us, a popular social bookmarking website, collected starting May 25th 2007. Each post on the recent feed is the result of a user associating a URL with one or more short text strings, such as *web* or *recipe*. Aggregating across posts, we recovered a dataset of 2,549,282 unique URLs. For many URLs, the dataset also includes a crawl of the page text and backlink page text.

To evaluate the quality of clusterings of the Tag Crawl dataset, we limited consideration to only a subset of 62,406 documents that is also present in ODP. Because these pages were all tagged by a user within the last year, they include some of the most recent and relevant pages in the directory. We discarded URLs in ODP's top-level "Regional" category, as its organizational structure is largely based on the geographical region pertaining to the site. Of the remaining documents, only 15,230 were in English and had their page text crawled as part of the Tag Crawl dataset. The documents are distributed as in Table 1. The documents were further divided into a 2,000 document development set for

ODP Name	#Docs	Top Tags by PMI
Adult	36	blog illustration art erotica sex
Arts	1446	lost recipes knitting music art
Business	908	accounting business lockpicking agency
Computers	5361	web css tools software programming
Games	291	un rpg fallout game games
Health	434	parenting medicine healthcare medical
Home	654	recipes blog cooking coffee food
Kids	669	illusions anatomy kids illusion copyright
News	373	system-unfiled daily cnn media news
Recreation	411	humor vacation hotels reviews travel
Reference	1325	education reference time research dictionary
Science	1574	space dreams psychology astronomy science
Shopping	310	custom ecommerce shop t-shirts shopping
Society	1852	buddhism christian politics religion bible
Sports	146	sport cycling nfl football sports
World	756	speed bandwidth google speedtest maps

Table 1: Intersection of ODP with the Stanford 2007 Tag Crawl dataset. The "regional" category has been elided.

parameter tuning and a 13,230 document test set for evaluating the final configurations reported here.

In our discussion, we differentiate between *types* and *tokens*. A word or tag *token* is an instance of a term being observed either in or annotated to a document, respectively. A word or tag *type* is a single unique term that is observed or annotated to at least one document in the collection, respectively. For example, a document with the text "the fuzzy dog pet the other fuzzy dog" and the tags ("dog", "fuzzy", "fuzzy") has eight word tokens, five word types, two tag types and three tag tokens.

Each document in the intersection of del.icio.us and ODP is represented as two sets of term occurrence counts—one for words and another for tags. Words were extracted from the Tag Crawl dataset and were tokenized with the Stanford Penn Treebank tokenizer, a fairly sophisticated finite state tokenizer. During processing, all word tokens appearing less frequently than the 10 millionth most common distinct word type were dropped as a first-cut term selection criterion [24, 37] as well as for reasons of computational efficiency. On average, a document contains 425 distinct word types and 1,218 word tokens. The tag occurrence counts make up the other data of each document. The complete set of tags was crawled from del.icio.us for each document without additional processing, yielding an average of 131 distinct tag types and 1,307 tag tokens out of a tag vocabulary of 484,499 unique tags (including many non-English tags). Because these documents in the ODP intersection tend to be generally useful websites, they tend to be more heavily tagged than most URLs in del.icio.us [21].

3. K-MEANS FOR WORDS AND TAGS

In this section, we examine how tagging data can be exploited by the K-means [26] algorithm, a simple to implement and highly scalable clustering algorithm that assumes the same vector space model as traditional ranked retrieval. K-means clusters documents into one of K groups by iteratively re-assigning each document to its nearest cluster. The distance of a document to a cluster is defined as the distance of that document to the centroid of the documents currently assigned to that cluster [26]. Distance is the cosine distance implied by the standard vector space model: all documents are vectors in a real-valued space whose dimensionality is the size of the vocabulary and where the sum of the squares of each document vector's elements is equal to 1. Our implementation initializes each cluster with 10 randomly chosen documents in the collection.

A key question in clustering tagged web documents using K-means is how to model the documents in the VSM. We examine five ways to model a document with a bag of words B_w and a bag of tags B_t as a vector V:

Words Only In step one, V is defined as $\langle w_1, w_2, \ldots w_{|W|}\rangle$ where w_j is the weight assigned to word j (based on some function f_w of the frequency of words in W and/or B_w). For example, w_j can be the number of times word j occurs in B_w (term frequency or tf weighting). In step two, V is l_2-normalized so that $||V||_2 = 1$.

Tags Only Analogous to words only, except we use the bag of tags B_t rather than the bag of words B_w and the tag vocabulary T rather than the word vocabulary W in step one.

Words + Tags If we define V_w to be the words only vector, above, and V_t to be the tags only vector, above, then the *Words+Tags* vector $V_{w+t} = \langle \sqrt{\frac{1}{2}}V_w, \sqrt{\frac{1}{2}}V_t\rangle$. In other words, we concatenate the two l_2-normalized vectors, giving words and tags equal weight. The intuition underlying this choice is that tags provide an alternative information channel that can and should be counted separately and weighted independently of any word observations.

Tags as Words Times n Analogous to words only, except in step one, instead of B_w we use $B_w \cup (B_t \times n)$. In other words, we combine the two bags, but we treat each term in the tag bag B_t as n terms. Instead of W we use $W \cup T$ as our vocabulary. For example, a document that has the word "computer" once and the tag "computer" twice would be represented as the word "computer" three times under the *Tags as Words Times 1* model, and five times under the *Tags as Words Times 2* model. This representation is sometimes used for titles in text categorization [11].

Tags as New Words We treat tags simply as additional (different) words. V is defined as:
$\langle w_1, w_2, \ldots w_{|W|}, w_{|W|+1}, w_{|W|+2}, \ldots w_{|W|+|T|}\rangle$
where w_j is the weight assigned to word j for $j \leq |W|$ or the weight assigned to tag $j - |W|$ for $j > |W|$. This is equivalent to pretending that all words are of the form *word#computer* and all tags are words of the form *tag#computer*. Then V is l_2-normalized.

These options do not cover the entire space of possibilities. However, we believe they represent the most likely and common scenarios, and give an indication of what representations are most useful. Nonetheless, it should be noted that one could optimize the relative weight given to words versus tags to maximize per-task performance.

In addition to deciding to model words or tags or both, we also need to answer the following questions:

1. How should the weights be assigned? Should more popular tags be weighted less strongly than rare tags? (Discussed in Section 3.1.)
2. How should we combine the words and tags of a document in the vector space model? Which of the vector representations presented above is most appropriate?

	tf	tf-idf
Words	.131	.152
Tags	.201	.154
Words+Tags	**.209**	.168

Table 2: F-scores of the vector space model document development collection (higher is better). Rows correspond to features and columns present the weighting function used.

(Discussed in Section 3.2.)

3. In the VSM, do tags help in clustering? (Discussed in Sections 3.1 and 3.2.)

3.1 Term Weighting in the VSM

In this subsection, we study the first question above: how should the weights be assigned? We study this question for the first three document models (*Words Only*, *Tags Only*, and *Words+Tags*). In particular, we consider two common weighting functions: raw term frequency (tf) and tf-idf. In computing term frequency, each dimension of the vector is set in proportion to the number of occurrences of the corresponding term (a word or tag) within the document. For tf-idf, each dimension is the term frequency downweighted by the log of the ratio of the total number of documents to the number of documents containing that term. For the *Words+Tags* scheme, we did not bias the weights in favor of words or tags (we normalized the combined vector with no preference towards either words or tags).

Table 2 demonstrates the impact of tf versus tf-idf weighting on the K-means F1 score for 2,000 documents set aside for this analysis. Note that K-means on *Words+Tags* significantly outperforms K-means on words alone under both term frequency and tf-idf. And the best performing model—term frequency weighting on *Words+Tags*—significantly outperforms tf-idf weighting on *Words+Tags*. However, the performance difference of term frequency on both *Words+Tags* does not significantly outperform the clustering on tags alone. As in the analysis of Haveliwala et al., [18], we believe that tf-idf weighting performs poorly in this task because it over-emphasizes the rarest terms, which tend not to be shared by enough documents to enable meaningful cluster reconstruction.

The results of this initial experiment suggest that term frequency weighting is an effective and simple means of assigning weights in our document vectors. We next address how to combine words and tags, using term frequency to assign weights to each vector element.

3.2 Combining Words and Tags in the VSM

Which of the five ways to model a document presented at the beginning of this section work best in the VSM? Table 3 shows the averaged results of ten runs of our best weighting (tf weighting) on the 13,230 documents not used for selecting the term weighting scheme. The *Words* and *Words+Tags* score are similar to the numbers in Table 2—their difference reflects the change in dataset between the two experiments. The inclusion of tags as words improves every condition over baseline, but all are significantly outperformed by the *Words+Tags* model. This suggests convincingly that tags are a qualitatively different type of content than "just more words" as has been suggested recently [6]. By simply normalizing the tag dimensions independently from the word dimensions of the underlying document vectors, K-means can very effectively incorporate tagging data as an independent information channel.

	K-means
Words	.139
Tags as Words ×1	.158
Tags as Words ×2	.176
Tags as New Words	.154
Words+Tags	**.225**

Table 3: F-scores for K-means clustering (tf) with several means of combining words and tags on the full test collection. All differences are significant except Tags as Words ×1 versus Tags as New Words.

4. GENERATIVE TOPIC MODELS

In the previous section, we saw the large impact to be had by appropriately including tagging information in K-means. Here we take that observation one step further by constructing a clustering model whose explicit probabilistic semantics is appropriate for modeling the nature of words and tags as independent sets of observations.

The model is a variation of latent Dirichlet allocation [8], a generative probabilistic topic model that is widely used in the machine learning community. LDA models each document as a mixture of hidden topic variables, where each topic is associated with a distribution over words. In this way, LDA is able to capture the notion that topics should be distinct even when documents themselves are not so clearly delineated. LDA adds fully generative probabilistic semantics to pLSI [22], which is itself a probabilistic version of Latent Semantic Indexing [13]. LSI and pLSI have seen use in the information retrieval community, but only recently have researchers used LDA in IR, such as by Wei and Croft who show in [35] that it can be used to estimate cluster-specific language models to improve retrieval results.

We ask three questions about LDA-derived models:

1. Can we do better than LDA by creating a model (defined in Sections 4.1 and 4.2) that explicitly accounts for tags and words as separate annotations of a document? (Discussed in Section 4.3.)
2. Do the same weighting and normalization choices from the VSM (Section 3) hold for generative models like LDA-derived models, or do they differ?[2] (Discussed in Section 4.3.)
3. Do LDA-derived models better describe the data and hence perform better on the tagged web document clustering task than clustering algorithms based on VSM? (Discussed in Section 4.4.)

4.1 MM-LDA Generative Model

In the context of tagging data, we extend LDA to jointly account for words and tags as distinct sets of observations. Our model takes its inspiration from a similar model for text and images proposed by Blei and Jordan [7]. We call our algorithm Multi-Multinomial LDA. The best way to describe MM-LDA is to outline the process it assumes has generated the dataset. We then maximize the likelihood of the

[2] Note that term weights have no natural interpretation in a conventional LDA-derived model, so we only compare methods of combining tags and words.

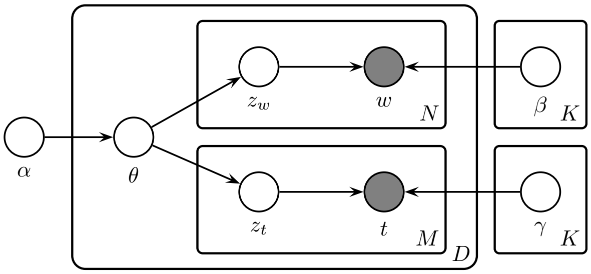

Figure 2: Graphical representation of MM-LDA.

data with respect to that process's parameters to reconstruct each document's cluster association probabilities as well as the probability of each word and tag per cluster. MM-LDA generates a collection of tagged documents from K topics by the process below and shown in Figure 2:

1. For each topic $k \in 1 \ldots K$, draw a multinomial distribution β_k of size $|W|$ from a symmetric Dirichlet distribution with parameter η_w. Each β_k represents the probability of seeing all word types given topic k.
2. Similarly, draw a multinomial γ_k of size $|T|$ from a symmetric Dirichlet with parameter η_t to represent the probability of seeing all tag types given topic k.
3. For each document $i \in 1 \ldots D$ in the collection, draw a multinomial θ_i of size $|K|$ from a Dirichlet distribution with parameter α. Each θ_i represents the probability of a word in that document having been drawn from topic i.
4. For each word index $j \in 1 \ldots N_i$ in document i:
   (a) Draw a topic $z_j \in 1 \ldots K$ from θ_i.
   (b) Draw a word $w_j \in 1 \ldots |W|$ from β_{z_j}.
5. For each tag index $j \in 1 \ldots M_i$ in document i:
   (a) Draw a topic $z_j \in 1 \ldots K$ from θ_i.
   (b) Draw a tag $t_j \in 1 \ldots |T|$ from γ_{z_j}.

Steps one, three, and four, in isolation, are equivalent to standard LDA. In step two, we construct distributions of tags per topic analogously to the construction of the word distributions per topic. In the final step, we sample a topic for each tag in the same way sampling a topic for each word.

4.2 Learning MM-LDA Parameters

One of several approaches can be used to learn the parameters $\beta_k, \gamma_k, \theta_i$. Variational inference [7] and Gibbs sampling [16] are two general techniques that have been used to learn the analogous parameters in LDA. We chose to extend a Gibbs sampling algorithm like the one analyzed by Wei and Croft [35] because its running time is asymptotically competitive with that of K-means. The algorithm is as follows: we iterate repeatedly through the documents in random order. For each word (and then for each tag) in random order, we resample a single topic z_j based on the current topic probabilities for that document and the probability of each proposed cluster assignment having generated the observed word. The Dirichlet prior parameters η and α effectively become pseudocount smoothing on the β and θ distributions, respectively, which we do not resample. This process repeats until convergence of the model's perplexity—a measure of its confusion on its input data—or earlier if a maximum of 100 iterations is reached. On the development set of 2000 documents, our LDA implementation runs in about 22 minutes whereas K-means runs in about 6 minutes. This 4:1 ratio holds up for the larger data sets as well.

	(MM-)LDA
Words	0.260
Tags as Words ×1	0.213
Tags as Words ×2	0.198
Tags as New Words	0.216
Words+Tags	**0.307**

Table 4: F-scores for (MM-)LDA across different tag feature modeling choices.

	(MM-)LDA	K-means
Words	0.260	.139
Tags	0.270	.219
Words+Tags	**0.307**	.225

Table 5: F-scores for (MM-)LDA and K-means on 13,320 documents. Including tags improves both models significantly versus words alone. MM-LDA (bold) significantly outperforms all other conditions.

We tested a wide range of smoothing parameters α, η_w, η_t for the MM-LDA model over 10 runs on the 2000 document validation set. We found that the model was fairly insensitive to the chosen values, except if the word or tag smoothing parameter was substantially smaller than the topic smoothing parameter (less than $\frac{2}{3}$ the other parameter). We chose 0.7 for the smoothing parameter for the word, tag, and topic distributions and used this value throughout.

4.3 Combining Words and Tags with MM-LDA

Does modeling words and tags separately improve performance in MM-LDA over a standard LDA model? Just as renormalizing the tag and word vector components separately improved K-means performance, the inclusion of tags as an alternative type of observation allows MM-LDA to flexibly model the tags and words that co-occur in the dataset. As an alternative, we could have employed a standard LDA model and added tags directly as words (*Tags as Words* ×1); added them as words with multiplicity two (*Tags as Words* ×2); or added them into an expanded region of the word feature space (*Tags as New Words).* By contrast, MM-LDA (*Tags+Words*) keeps distinct multinomial distributions for the occurrence of tags and words under a particular topic. Table 4 presents F-scores of LDA and MM-LDA under these model variations.

MM-LDA's *Words+Tags* model significantly outperforms all other configurations. Interestingly, the addition of tags to the word vectors decreases the performance of the algorithm relative to words alone. We believe this decrease is due in part to the very different distributional statistics observed for words versus tags. In particular, for our dataset there tend to be about 4 times as many word types as tag types and yet a similar number of tokens for each. When combined, the word multinomials for many topics may become disproportionately peaked around common tags at the expense of flexibility in modeling either.

4.4 Comparing K-Means and MM-LDA

How does the probabilistic model of MM-LDA perform compared to the VSM of K-means? In this section, we compare MM-LDA to K-means quantitatively and qualitatively.

Tag-Augmented K-means

	tags	*words*
1	linux security php opensource vpn unix	linux ircd php beware kernel exe
2	games go game sports firefox gaming	dmg munsey ballparks suppes racer game
3	music research finance audio mp3 lyrics	music research redirect nottingham meta laboratory
4	news business newspaper politics media magazine	v business leadership d news j
5	politics activism travel movies law government	aquaculture terrapass geothermal anarchist wwoof cpsc
6	science physics biology astronomy space chemistry	science wildman foraging collembola physics biology
7	css python javascript programming xml webdesign	squeakland sql coq css python flash
8	food recipes cooking shopping tea recipe	recipes food cooking recipe stylist tea
9	blog blogs fashion design art politics	flf blog comments posted my beuys
10	education art college university school teaching	learning gsapp students education school cutecircuit
11	health medical healthcare medicine solar psychology	health napkin cafepress.com medical care folding
12	java programming development compiler c opensource	java c programming goto code language
13	software windows opensource mac freeware osx	software windows mac download os thinkfree
14	dictionary reference language bible writing english	dictionary english words syw dictionaries spanish
15	internet dns search seo google web	internet shutdown sportsbook epra kbs npower
16	history library books literature libraries philosophy	library tarot peopling ursula guin bowdoin

Multi-Multinomial LDA (MM-LDA)

	tags	*words*
1	web2.0 tools online editor photo office	icons uml powerpoint lucid dreams dreaming
2	guitar scanner chemistry military earthquake groupware	grub outlook bittorrent rendering recovery boot
3	health medical medicine healthcare process gardening	exe health openpkg okino dll polytrans
4	bible christian space astronomy religion christianity	gaelic bible nt bone scottish english
5	politics activism environment copyright law government	war shall power prisoners their article
6	social community web2.0 humor fun funny	press f prompt messages ignoring each
7	reference science education research art books	science research information university search site
8	java database programming development mysql sql	java sql mysql schizophrenia testing test
9	dictionary language english reference translation thesaurus	english writing dictionary spanish words bppv
10	travel search maps google reference map	search deadline call flf conference paper
11	time clock timezones world train md5	quantum thu pfb am pm mf
12	food recipes cooking business shopping finance	my food tea wine me recipes
13	news blog music blogs technology system/unfiled	comments blog he posted news pm
14	programming software webdesign web css design	you can if or not use
15	photography photo compression zip photos photoblog	flash camera eos light e-ttl units
16	mac apple osx games unicode game	dmg u x mac b v

Table 6: Highest scoring tags and words from clusters generated by the K-means (above) and MM-LDA (below) from one run of the 2000 document development set. The K-means terms are selected by top tf-idf and the MM-LDA terms are selected by highest interest value.

4.4.1 Quantitative Comparison

We clustered documents using the K-means and LDA models on the 13,320 document test collection under three conditions: just *Words*, just *Tags*, or jointly *Words+Tags*. For K-means, we used tf weighting, which includes the best performing model for K-means, *Words+Tags*. Table 5 shows that the inclusion of tagging data significantly improves the performance of MM-LDA versus tags or words alone. The improvement from moving from Words to Words+Tags was significant for both models. In contrast to K-means, LDA's improvement from *Tags* to *Words+Tags* was also significant. MM-LDA's *Words+Tags* model is significantly better than all other models. From this we conclude that, under some conditions, MM-LDA is better able to exploit the complementary information in the word and tag channels.

4.4.2 Qualitative Comparison

Qualitatively, both K-means and MM-LDA learn coherent clusters in the document collections, as demonstrated by the top scoring words and tags associated with each cluster in Table 6. In addition to associating documents to topics, each algorithm outputs per-cluster affinities to words and to tags. When analyzing the generated affinities, it is important to take into account the underlying model assumed by each algorithm. K-means operates in document vector space, so we extract its top-scoring words and tags per cluster by selecting those terms with the highest tf-idf weighted score. By contrast, MM-LDA outputs multinomial probability distributions, which tend to be highly peaked and inappropriate for tf-idf weighting. For MM-LDA, we select a term t for cluster c if it has one of the highest values of *interest*, defined as be $p(t|c) - p(t)$. The interest operator balances the desire to select terms that have high absolute probability in their cluster with low probability overall.

5. FURTHER STUDIES

Lastly, we consider two questions independent of the clustering algorithm family:

1. Does the addition of anchor text to regular plain text make tags redundant? Do our algorithms that take into account tags still outperform anchor text + plain text together? (Discussed in Section 5.1.)
2. If we look at multiple levels of specificity of clusters, for example, clustering programming language documents rather than clustering general documents, does tagging data help? More or less? (Discussed in Section 5.2.)

5.1 Tags Are Different Than Anchor Text

Do the advantages of tagging data hold up in the presence of anchor text? Anchor text — the text of and around incoming web hyperlinks — has helped in some tasks that use web document corpora like web search [14] and text classification [15]. Like tags, anchors act as free-form document annotations provided by a third party. For each URL in the tag crawl dataset, we extracted words within 15 tokens of hyperlinks to that URL in up to 60 pages returned by a Google API backlink query. This window size was consistent with the best results for anchor text window size for similarity search found in [18].

We experimented with two means of combining page text, anchor text, and tags. *Anchors as Words* adds all words in the extracted anchor text windows to each document's word vector analogously to the *Tags as Words* model in Section 3. *Words+Anchors* weights anchor text words separately from the document words, like the *Words+Tags* model. The results of these model variants on the top-level ODP clustering task, as well as when Tags are added as an independent information channel to each of them, are presented in Table 7.

	(MM-)LDA	K-means
Words	.260	.139
Anchors as Words	.270	.120
(Anchors as Words)+Tags	.281	.214
Words+Anchors	.248	.128
Words+Anchors+Tags	**.306**	.224

Table 7: Inclusion of tags in (MM-)LDA and K-means increases F1 score on the test collection even in the presence of anchor text.

We found that both MM-LDA and K-means gain from the inclusion of tagging data as compared to clustering on *Anchors as Words* or *Words+Anchors* alone. However, the results from the inclusion of anchor text are mixed. While performance of LDA improved when anchors were added as new words (*Anchors as Words*), K-means performance was slightly depressed because of the vector space model's sensitivity to the weights of the now-noisier terms. Neither model did well with *Anchors+Words*, reflecting the difficulty of extracting a quality anchor text signal for text clustering, especially from a relatively small web crawl. We believe that these numbers might be improved by down-weighting anchor words as a function of their distance from the URL or exploiting more advanced term weighting techniques as in [18]. However, even under such transformations, we argue that the inclusion of tagging data would still improve cluster quality.

5.2 Clustering More Specific Subtrees

Does the impact of tags depend on the specificity of the clustering? Clustering the top-level ODP subtrees is a difficult task because many coherent subtopics exist for each top-level ODP category. We believe real-world applications may benefit from clustering either a wide variety of documents, as in the top-level ODP clustering task, or documents that are focused, such as those returned by a search query.

To investigate the applicability of tag-based clustering for more specific document collections, we selected two representative ODP subtrees that each had a substantial number of documents in our dataset. The *Programming Languages* subcategory is the set of documents labeled with a subcategory of ODP's Top/Programming/Languages category. The gold-standard labels for this subset of 1,094 documents are: Java, PHP, Python, C++, JavaScript, Perl, Lisp, Ruby, and C. Documents in this subset tend to share many specific terms related to programming (e.g. words such as *loop*, and *compile*), so clustering this subcategory is not unlike clustering some types of search results.

The *Social Sciences* subcategory (SS) is the set of documents labeled with a subcategory of ODP's Top/Society tree. The 1,590 documents in this subset are each labeled as one of: Issues, Religion & Spirituality, People, Politics, History, Law, or Philosophy. This collection represents a diverse set of topics unified by a common theme with many overlapping terms, but in a broader vocabulary space than the PL subset.

		(MM-)LDA	K-means
Programming Languages	Words	.288	.189
	Tags	.463	.567
	Words+Tags	.297	.556
Social Sciences	Words	.300	.196
	Tags	.310	.307
	Words+Tags	.302	.308

Table 8: F-scores for (MM-)LDA and K-means on two representative ODP subtrees. For these tasks, clustering on tags alone can outperform alternatives that use word information.

Both clustering algorithms performed at least as well in these ODP subsections as they did for the directory as a whole, as shown in Table 8. Tags appear to be better indicators than words in isolation, and, indeed they are so much better that jointly modeling tags and words can actually depress performance. This surprising phenomenon stems in part from the fact that users tend to tag pages at a level of specificity appropriate for their own information needs, which often correspond to the types of distinctions made within ODP subsections. For example, within the Java subcategory of "Programming Languages", the most common tag, "java", covers 488/660 = 73.9% of pages. By contrast, in the top-level "Computers" subcategory, the most common tag "software" covers only 2562/11894 = 21.5% of pages. Because the size of the tag vocabulary within these ODP subsections is substantially reduced from the full tag vocabulary, a higher proportion of the remaining tags are direct indicators of sub-category membership than in the top-level clustering task. We believe that the extra signal present in the words plays a lesser role and, indeed, can reduce the quality of the overall clustering. This factor applies to both models, even when K-means outperforms LDA, as on the Programming Languages cluster, where a smaller set of focused tags plays to the strengths of the vector space model's independence assumptions.

6. RELATED WORK

The impact of social bookmarking data has been explored in several other contexts within information retrieval and the web, including in ranked retrieval e.g., [3, 21, 23, 36] and analysis of blogs [9, 19]. Others have used tags in some clustering contexts, such as Begelman et al. [4] who conclude that clustering of tags should be used in tagging systems, for example, to find semantically related tags.

In modeling, the most closely related work to ours is Zhou et al.'s recent paper [40], which (like ours) looks at the potential to generatively model social annotation to improve information retrieval. That work's evaluation focuses on a specific, promising, application of improving language model based information retrieval. As a result, it produces evidence that good generative models for social annotation can in fact have a positive impact on ranked result quality for language model based information retrieval systems. Our work uses a more general evaluation metric, similarity to a gold standard (inspired by Haveliwala et al. [18]), and further assumes that search engines have access to anchor text. We believe our more general evaluation metric may make our results more applicable to the broader group of applications outlined in Section 1 while still making them convincingly applicable to language model based information retrieval, due to Zhou et al.'s work. Lastly, our MM-LDA generative model is more directly descended from Blei et al.'s work on annotation [7] than is the model in [40], which we hope makes our work more applicable to the popular current area of image retrieval with tags (see, for example, [2, 33, 29]).

A host of applications have grown out of the ability to classify web pages into web directories, including topic-sensitive web link analysis [17] and focused crawling [10]. Our work is related to this work in that we use ODP as a gold standard for our evaluation. However, it is different in that our goal is not to predict ODP classes (for which we might use a supervised method) or to create a hierarchy similar to ODP (for which we might use hierarchical clustering) but rather to improve information retrieval through clustering.

7. DISCUSSION

Many of the newest and most-relevant parts of the web are constantly being tagged on large social bookmarking websites. In this work we have also found that many pages of interest are often those with the most tags. In fact, the pages that are informative and relevant enough to be in both the tag crawl dataset and in ODP have, on average, as many tag annotations as words. And because tagging happens more quickly than links for new content, tags promise to become an increasingly important signal for ranking new pages of high static quality. The baseline clustering algorithms extended in this work are themselves high-performers on traditional document clustering tasks. By exploiting tagging data when available, these techniques promise to improve web document clustering in general, and especially so for the most relevant parts of the web.

In future work, we hope to apply the tag-extended clustering algorithms explored here to directly support search or browsing. We are also interested in the histories and vocabularies of individual users, and hope to explore the time series nature of the tag stream in more detail through more targeted probabilistic graphical models.

As a final note, it is perhaps worth contrasting modern tagging with other types of indexing vocabularies. The traditional comparison in the field has been between controlled indexing languages—characterized by a specific indexing vocabulary structured in advance—and full text indexing, in which the documents themselves provide all indexing terms. In some ways, tagging sits between these two extremes. As in a controlled indexing language, human beings select the terms in a tagging system that characterize a document well. Tags therefore have a level of semantic precision that full text indexing lacks. Yet in other ways, tagging is more like free text indexing in that there is no pre-defined vocabulary or hierarchy, tags can freely have multiple meanings, and different tags can be used for the same topic. Tagging is like free text indexing in some other important respects, as well: with ample tagging data, tags have frequency counts, just like words, and the range of tags applied to popular documents is more exhaustive than what is typical in a controlled vocabulary. In sum, we can at least hope that because tags represent human semantic classification, tagging has the potential to improve the precision of searches as well as the quality of inferred document clusters, while the exhaustivity of tagging means that the technique will avoid the biggest limitation of traditional use of controlled indexing vocabularies.

8. CONCLUSION

This work has demonstrated that social tagging data provides a useful source of information for the general problem of web page clustering, a task core to several IR applications. We have shown that tagging data improves the performance of two automatic clustering algorithms when compared to clustering on page text alone. A simple modification to the widely used K-means algorithm enables it to better exploit the inclusion of tagging data. A novel algorithm—MM-LDA, an extension to latent Dirichlet allocation for use with parallel sets of observations—makes even better use of the complementary similarity information held in a document's words and tags on a general web clustering task.

9. REFERENCES

[1] Open directory project. http://dmoz.org/.

[2] M. Aurnhammer, P. Hanappe, and L. Steels. Integrating collaborative tagging and emergent semantics for image retrieval. *Proc. of the Collaborative Web Tagging Workshop (WWW'06).*

[3] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *WWW '07.*

[4] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. *Proc. of the Collaborative Web Tagging Workshop (WWW'06).*

[5] S.M Beitzel, E.C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *SIGIR '04.*

[6] B. Berendt and C. Hanser. Tags are not Metadata, but "Just More Content"–to Some People. *ICWSM '07.*

[7] D.M. Blei and M.I. Jordan. Modeling annotated data. In *SIGIR '03.*

[8] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2003.

[9] C.H. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW'06.*

[10] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW '99.*

[11] W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. In *SIGIR '99.*

[12] D.R. Cutting, D.R. Karger, J.O. Pedersen, and J.W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *SIGIR '92.*

[13] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[14] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *SIGIR '03.*

[15] Johannes Fürnkranz. Exploiting structural information for text classification on the WWW. In *IDA '99.*

[16] T.L. Griffiths. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, '04.

[17] T. Haveliwala. Topic-sensitive pagerank. In *WWW '02.*

[18] T. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the web. In *WWW '02.*

[19] C. Hayes and P. Avesani. Using tags and clustering to identify topic-relevant blogs. In *ICWSM*, 2007.

[20] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR '96.*

[21] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search. In *WSDM '08.*

[22] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99.*

[23] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. *The Semantic Web: Research and Applications*, 4011:411–426, 2006.

[24] T. Liu, S. Liu, Z. Chen, and W.Y. Ma. An evaluation on feature selection for text clustering. In *ICML '03.*

[25] X. Liu and W.B. Croft. Cluster-based retrieval using language models. In *SIGIR'04.*

[26] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008.

[27] K.R. McKeown, R. Barzilay, D. Evans, V. Hatzivassiloglou, J.L. Klavans, A. Nenkova, C. Sable, B. Schiffman, and S. Sigelman. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *HLT'02.*

[28] S. Osinski and D. Weiss. A concept-driven algorithm for clustering search results. *IEEE Intelligent Systems*, 20(3):48–54, 2005.

[29] T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from Flickr tags. In *SIGIR '07.*

[30] K. Song, Y. Tian, W. Gao, and T. Huang. Diversifying the image retrieval results. In *MULTIMEDIA '06.*

[31] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *AAAI Workshop on AI for Web Search (AAAI 2000).*

[32] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[33] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04.*

[34] Ellen M. Voorhees. The cluster hypothesis revisited. Technical report, Ithaca, NY, USA, 1985.

[35] X. Wei and W.B. Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR '06.*

[36] Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka. Can social bookmarking enhance search in the web? In *JCDL '07.*

[37] Y. Yang and J.O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97.*

[38] Oren Zamir and Oren Etzioni. Web document clustering: a feasibility demonstration. In *SIGIR '98.*

[39] H.J. Zeng, Q.C. He, Z. Chen, W.Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR '04.*

[40] D. Zhou, J. Bian, S. Zheng, H. Zha, and C.L. Giles. Exploring social annotations for information retrieval. In *WWW '08.*