LLNL-TR-517731

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

# Rapid Exploitation and Analysis of Documents

D. J. Buttler, D. Andrzejewski, K. D. Stevens, D. Anastasiu, B. Gao

December 2, 2011

**Disclaimer**

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

# Rapid Exploitation and Analysis of Documents

David Buttler David Andrzejewski Keith Stevens David Anastasiu Byron Gao

## ABSTRACT

Analysts are overwhelmed with information. They have large archives of historical data, both structured and unstructured, and continuous streams of relevant messages and documents that they need to match to current tasks, digest, and incorporate into their analysis. The purpose of the READ project is to develop technologies to make it easier to catalog, classify, and locate relevant information. We approached this task from multiple angles. First, we tackle the issue of processing large quantities of information in reasonable time. Second, we provide mechanisms that allow users to customize their queries based on latent topics exposed from corpus statistics. Third, we assist users in organizing query results, adding localized expert structure over results. Forth, we use word sense disambiguation techniques to increase the precision of matching user generated keyword lists with terms and concepts in the corpus. Fifth, we enhance co-occurence statistics with latent topic attribution, to aid entity relationship discovery. Finally we quantitatively analyze the quality of three popoular latent modeling techniques to examine under which circumstances each is useful.

## 1. OVERVIEW

The analysis of unstructured and structured text documents is a fundamental part of both government and business intelligence. Most of the information analysts need to process information is represented as text, including newspapers and web sites, scientific articles in journals, and proprietary messages and analysis. The amount of such information is staggering, with hundreds of millions of individual documents. Unfortunately, most analysts have very few tools to evaluate this vast amount of unstructured data. While there has been significant advances for new types of analytic tools, such as Palantir [1], the most common tool used today by most analysts is simple Boolean keyword search.

The project described here extends the capability of the the user interfaces that analysts are accustomed to, enabling analysts to assess the relevance of individual documents and find interesting documents from a massive document set.

[1]http://www.palantirtech.com/

There are several aspects of the project, each of which is described below. The main theme is tying together multiple components to create a unified interface that takes advantage of the latest in information retrieval research and systems software. We attempt to address the following analyst problems: managing large number of documents, and keeping everything accessible via search; coming up with the right keywords to find targeted concepts or actors; organizing search results so that they are comprehensible; precisely identifying concepts of interest without having to wade through masses of unrelated terms; and searching by entity networks rather than document sets. Finally we also examine some of the conceptual underpinnigs of our approach, measuring various alternatives that can have a huge impact on the quality of the statistical summary information we both present to analysts to help them understand a corpus and the various mechanisms we use to help them in their search.

**Infrastructure**
Managing large corpora of documents is a difficult task. However, recent years have seen significant advances in open-source software infrastructure that makes the problem more tractable. The major improvements include information retrieval software — specifically Lucene and Solr — and simplified distributed processing systems, such as the Hadoop Map/Reduce implementation.

Solr provides the infrastructure for querying large numbers of documents; it includes faceted search to allow users to refine their search by different aspects of the corpus. However, the facets must be generated and placed there by operators of the system. Creating those facets is often a processing intensive task requiring both a global view of the corpus and the information in the text of the local document. We use Map/Reduce to distribute the processing load across a cluster to make the processing time tractable.

**Enhanced Keyword Search**
Another issue that comes up is assisting the user in understanding a specialized and unfamiliar corpus. Typical search terms may be less useful, and there are fewer standard external resources (concept hierarchies, user query logs, links to wikipedia, etc.), that can be leveraged to provide search guidance in internal information systems. In these cases we exploit statistical structure in the corpus to enhance query strings, and to provide an overal gist for the corpus.

**`ClusteringWiki`**
As soon as a query is submitted, there are several things that can be done to improve the result lists. Updated facet counts provide one digested view of the results. Re-ranked documents, specialized for a task, provide another view. Relevant latent topic themes give

another viewpoint. What these techniques lack is a user manipulatable means to organize the search results. The main mechanism that user feedback has been incorporated in the Internet is through user tagging, creating a folksonomy — a user created taxonomy with no ridgid semantic rules. However, tagging individual documents can be a cumbersome process that users will only participate in given sufficient intrinsic incentive. I.e. users must obtain some value for the effort they invest. Our mechanism, `ClusteringWiki`, allows users to tag groups of documents that either naturally cluster together, or are explicitly marked by users.

The system tries to find relevant similarities in query results, choosing frequent phrases as the default label. By interacting with the default clustering, by renaming clusters, moving items between clusters, or deleting items, users are implicitly making a large number of taggings that can be re-used by others, or the same user, at a later date. The incremental interactions of the users accrete over time to provide a unique user-generated facet for the data. While this type of content is an easy target for spam, and irrelevant noise in a open Internet setting, it becomes much more relevant internal to an organization, where limiting input to a small group of experts and interested users alleviates many of these challenges.

**Word Sense Disambiguation**
While clustering results is one way of attacking polysemy, users often want to approach data from the other direction. Analysts spend considerable time developing keyword lists, and other knowledge artifacts that describe their domain of interests. These keyword lists may be used to generate Boolean queries to search engines, or they may be used in grep-like fashion to find relevant documents. The main problem with this approach is that domain-specific words have unrelated generic meanings. There are at least 72 unique acronyms for the letters “or”; “lead” has a very popular common meaning (e.g. “He is a leader of men”), as well as a useful scientific meaning (“The atomic number of lead, *Pb*, is 82”).

While these examples may seem contrived, this comes up frequently in practice. Any list that contains a term with a common meaning becomes instantly less valuable. It is possible to remove terms with common meanings, but this simply removes the possibily of finding targeted usage of the desired terms. By examining the mechanics of word sense disambiguation, we hope to provide a robust search mechanism that provides fine-grained control – allowing the user to more reliably detect specific meanings of terms, and provide control to the user (or system administrator) for the remaining trade-off between precision and recall.

**Topic Enhanced Entity Co-occurence**
An alternate mechanism for exploring the information available is to focus purely on the entities that occur in documents. Often the ultimate goal is to find individuals or institutions that match some search criteria (such as a person who is expert in a particular field, or an organization involved in a particular activity). Analysts often have a good idea for a starting point, and what they need to do is develop a network of contacts and associations to help them generate a broader understanding of how a particular actor fits into the larger picture.

**Quantitative Comparisons of Topic Modeling Approaches**
Topic modeling has become a popular way to discover themes and structure in a corpus using unsupervised techniques. The basic idea is not new: it is essentially a dimension reduction. Topic models learn bags of similar words from a collection of documents. What has happened over the past decade is that there are now several different mechanisms to create these latent topics, include LSA using SVD or NMF, and LDA. Recently researchers have come up with a user-validated mechanism to measure the semantic coherence of LDA models algorithmicly. These coherence measures score individual topics so they can be ranked based on the semantic similarity of topic words. As topic models play an increasingly large role in enhanced information retrieval systems, particularly in the techniques described above, understanding which topic models are most appropriate for different tasks has become increasingly important. We apply two such metrics to three distinct topic modeling approaches, and explore the differences. These new results have significant implications for how we exploit the other techniques described in this paper going forward.

## 2. INFRASTRUCTURE DEVELOPMENT AND DEPLOYMENT

Traditionally, natural language processing systems were not designed with any concern for scale. This comes from two understandable viewpoints: first, it is very expensive to develop ground truth to validate various algorithms (named-entity recognition, relationship extraction, coreference resolution, etc.), resulting in small training and test sets. Second, since the results were extremely poor, there did not seem to be a need to scale the systems until they could be deemed to be reliable enough for use. Coincidentally, there are two major developments that have rendered those concerns moot. First, the web has developed in such a way as to present us with an enormous quantity of data; concurrently, researchers and industry have leveraged this data to dramatically improve the results of many types of algorithms. As a single example, Google has shown stunningly accurate machine translation results based on their collection and collation of billions of web pages in different languages.

Several open source projects have started to take advantage of the concurrent explosion in data and ideas for leveraging that data for useful purposes. One of the earliest examples is the Lucene[2] (and later Solr[3]) search projects. Lucene provides an indexing and search implementation; Solr extends that capabilities by providing faceted capabilities and a web application container for the core search features.

A second development has been the commoditization of Map/Reduce distributed processing [28], and large-scale distributed file system [38] by Hadoop[4]. The HBase[5] project provides a large NoSQL key-value storage system for the Hadoop file system that provides fast access to hundreds of millions of records.

The Map/Reduce paradigm makes it trivial process hundreds of millions of documents with methods designed for a limited set of documents. We have taken several open source data processing projects and changed their interfaces to operate over a single document object presented in memory. These API’s can then be plugged into Hadoop as simple map operators, and become flexible building blocks for the larger system.

Prior to adopting Hadoop, we used a standard NFS filesystem, on

[2]http://lucene.apache.org
[3]http://lucene.apache.org/solr/
[4]http://hadoop.apache.org
[5]httpd://hbase.apache.org

a NetApp[6] appliance, for our coreference [108] project. One of the tasks we wanted to do is process the New York Times corpus [101] for coreferent mentions. The first computer we tried running the pipeline on was able to process a sentence at approximately the same rate that the New York Times was publishing new sentences. In addition, we used a fairly naive file format that worked very well for a distributed team working on different system components, targeted at standard test sets containing on the order of 1000 documents. Scaling up to the 1.8 million documents in the New York Times corpus added a host of new issues, where a collection of six to ten files per document became a serious impediment. While we had a large cluster of over 100 machines to process the articles, they were all kept in a single shared NFS mount. Launching processing on each of the cluster nodes brought the file system to a standstill – other users were outraged that they could not even list files in a directory.

This experience led us to adopting HBase, which is based on BigTable[21], as our storage architecture for the document set. HBase provides a horizontally partitioned keyspace, automatically distributed and balanced over the cluster. Each key can be associated with an arbitrary set of columns. For convenience we restricted our columns to the text of the documents and annotations over that text. This allows us to incrementally process documents, with simple well defined processes to add individual annotations, building up more complex results at each stage.

This simple architecture is very powerful, and allows arbitrarily complex processing to be broken down into a sequence of simple steps. On a relatively new cluster of only 6 machines, we have been able to process articles from 20 years of the New York Times corpus in just a few hours.

The architecture is also very flexible. Each of the processing components is designed to operate on a generic document record; this means that to add a new source only requires a translation step to convert documents from the orginal source into the generic record format. Open source components, like Apache Tika [7] handle converting standard document types into plain text and associated metadata. The only remaining requirement is data extraction for custom sources (like ProMED mail [8]), which must be dealt with by every data management platform.

In summary, we are able to handle hundreds of millions of records, and tens of terabytes on a small cluster. As with any Hadoop or Solr cluster, there is a well defined and simple plan for expanding, simply by adding more nodes to the cluster. The software handles scaling to pretty much any conceivable size (FaceBook has a cluster that manages over 21PB of data on 2000 machines [111, 16] [9]). Solr provides a search platform to match the size of the data, with the capability to distribute the index over the entire cluster. It provides features, such as faceting, which we can exploit to provide unique capabilities described elsewhere in this report. Just getting these two capabilities working on relevant data allows huge advances over the previous state of the art – capabilities that many groups could profitably leverage.

The remaining sections are excerpts from published conference papers that address each of the problems listed above. Section 3 covers enhanced keyword search; Section 4 covers clustering search results; Section 5 covers techniques in word sense disambiguation; Section 6 covers clustering entities and their relationships by topic; finally, Section 7 discusses the choice of topic modeling approach, a key component of many of the algorithms and technques discussed earlier.

[6]http://netapp.com
[7]http://tika.apache.org
[8]http://www.promedmail.org
[9]http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html

## 3. ENHANCED KEYWORD SEARCH

We consider the problem of a user navigating an unfamiliar corpus of text documents where document metadata is limited or unavailable, the domain is specialized, and the user base is small. These challenging conditions may hold, for example, within an organization such as a business or government agency. We propose to augment standard keyword search with user feedback on latent topics. These topics are automatically learned from the corpus in an unsupervised manner and presented alongside search results. User feedback is then used to reformulate the original query, resulting in improved information retrieval performance in our experiments.

### 3.1 Introduction

This work addresses the problem of *ad hoc* information retrieval and text corpus navigation under the following conditions. First, document metadata may be limited, unreliable, or nonexistent. Second, the domain of the text documents is specialized, using vocabulary ill-covered by general web documents or lexical resources such as WordNet [79]. Finally, while the text corpus itself may be large, the set of users accessing the corpus is not. This is an important problem because these conditions can preclude the use of effective information retrieval techniques such as faceted search or query log mining. These conditions are different from those encountered in general web or e-commerce search, but are realistic *within* organizations which are trying make sense of large quantities of text, such as private enterprises or government agencies.

A central problem in *ad hoc* information retrieval is that users may not be able to formulate the “right” keyword combination in order to retrieve the most relevant documents. Techniques such as real-time query expansion [119] have been developed to directly attack this problem, but often rely upon a dataset of previously submitted queries, which may be sparse without a large user base. Another approach is to solicit alternative types of user input. Faceted document navigation [106] allows users to select documents based on different attributes (e.g., publication venues or hierarchical subject categories) and has emerged as a powerful complement to traditional keyword search. However, the standard assumption is that the facets are manually defined, and that facet values for each document are known.

Because of the challenging scenario we have defined, it is important to exploit all available data. Latent topic models such as Latent Dirichlet Allocation (LDA) [13] provide a means to take advantage of the statistical structure of the corpus itself. LDA assumes that observed documents have been generated by weighted mixtures of unobserved (latent) topics. These topics are learned from the documents and often correspond to meaningful semantic themes present in the corpus. LDA and its extensions have found interesting applications in fields such as natural language processing, computer vision, and social networks analysis [10].

The contribution of this work is a new method for obtaining and exploiting user feedback at the *latent topic* level. Our approach is to learn latent topics from the corpus and construct meaningful representations of these topics. At query time, we then decide *which* latent topics are potentially relevant and present the appropriate topic representations alongside keyword search results. When a user selects a latent topic, the vocabulary terms most strongly associated with that topic are then used to augment the original query. Our experiments with simulated user feedback show improved information retrieval performance. The presentation of relevant topics alongside search results also has the additional benefit of helping the user to understand corpus themes related to the original keyword query.

## 3.2 Related work

Our approach is partially motivated by the successes of faceted search [123]. Castanet [106] and related systems [27] aim to *automatically* construct facet hierarchies, but these techniques depend crucially on the existence of a rich lexical resource such as WordNet [79]. While specialized ontologies or controlled vocabularies have been constructed for some domains such as Gene Ontology (GO) [110] and Medical Subject Headings (MeSH) [15], the constraints of our setting prohibit us from assuming the existence of such a resource.

In light of this issue, topic models such as LDA have the advantage of relying upon corpus statistics alone. Indeed, previous analysis [81] of the digital library of the Open Content Alliance (OCA) directly posited the analogy between latent topics and faceted subjects, although specific mechanisms for exploiting this insight were not explored. The Rexa academic search engine[10] also displays relevant latent topics as tags for a given research article, allowing further investigation of the topics themselves. Another interesting topic modeling approach uses seed words to learn facet-oriented topics which can then be used to construct informative summaries [70].

LDA has previously been used in information retrieval for both document language model smoothing [118, 72] and query expansion [88]. These techniques both exploit the dimensionality reduction provided by LDA "behind the scenes" in order to improve performance, but do not leverage explicit user feedback in the way that our approach does. The approach we propose in this work can be viewed as complementary to these existing enhancements.

The BYU Topic Browser [37] provides an environment for rich explorations of learned LDA topics and how they relate to words and documents within a corpus. However, the tasks supported are more appropriate for advanced analysis by a relatively sophisticated user, as opposed to a general search setting.

## 3.3 Our approach

We propose to present automatically learned topics alongside keyword search results, allowing the user to provide feedback at the latent topic level. While it is well-known that we can learn latent topics with LDA, incorporating these topics into an information retrieval system requires us to address several questions. First, how should these topics be presented? Previous user studies [106] have found that users can become frustrated by raw LDA output. Second, which topics should be presented for a given query? To avoid overwhelming the user, we clearly cannot present all latent topics (potentially hundreds or greater) for every query. Furthermore, not all learned topics truly correspond to meaningful semantic concepts, and the presence of these incoherent topics will not be appreciated by users either. Third, how can we incorporate user latent topic feedback into search results? Ideally, the mechanism used should be simple and easy to integrate with existing search technologies. Finally, can this type of feedback improve information retrieval performance, as measured by standard metrics?

[10]http://rexa.info/

We now describe our approach, beginning with a brief review of latent topic modeling concepts and moving on to address the above questions. All examples shown are actual learned topics from the experimental datasets described in Table 5. In Section 3.4, experimental results demonstrate that our approach can indeed achieve performance gains.

### *3.3.1 Latent Dirichlet Allocation (LDA)*

In LDA [13], it is assumed that observed words in each document are generated by a document-specific mixture of corpus-wide latent topics. We define our corpus of length $N$ with the flat word vector $\mathbf{w} = w_1 \dots w_N$. At corpus position $i$, the element $d_i$ in $\mathbf{d} = d_1 \dots d_N$ designates the document containing observed word $w_i$. Similarly, the vector $\mathbf{z} = z_1 \dots z_N$ defines the hidden topic assignments of each observed word. The number of latent topics is fixed to some $T$, and each topic $t = 1 \dots T$ is associated with a topic-word multinomial $\phi_t$ over the $W$-word vocabulary. Each $\phi$ multinomial is generated by a conjugate Dirichlet prior with parameter $\beta$. Each document $j = 1 \dots D$ is associated with a multinomial $\theta_j$ over $T$ topics, which is also generated by a conjugate Dirichlet prior with parameter $\alpha$. The full generative model is then given by

$$P(\mathbf{w}, \mathbf{z}, \phi, \theta \mid \alpha, \beta, \mathbf{d}) \propto \left(\prod_t^T p(\phi_t|\beta)\right) \left(\prod_j^D p(\theta_j|\alpha)\right) \left(\prod_i^N \phi_{z_i}(w_i)\theta_{d_i}(z_i)\right),$$

where $\phi_{z_i}(w_i)$ is the $w_i$-th element in vector $\phi_{z_i}$, and $\theta_{d_i}(z_i)$ is the $z_i$-th element in vector $\theta_{d_i}$. Given an observed corpus $(\mathbf{w}, \mathbf{d})$ and model hyperparameters $(\alpha, \beta)$, the typical modeling goal is to infer the latent variables $(\mathbf{z}, \phi, \theta)$.

While exact LDA inference is intractable, a variety of approximate schemes have been developed [82, 13, 109]. In this work, we use Markov Chain Monte Carlo (MCMC) inference, specifically collapsed Gibbs sampling [40]. This approach iteratively re-samples a new value for each latent topic assignment $z_i$, conditioned on the current values of all other $z$ values. After running this chain for a fixed number of iterations, we estimate the topic-word multinomials $\phi$ and the document-topic mixture weights $\theta$ from the final $\mathbf{z}$ sample, using the means of their posteriors given by

$$\phi_t(w) \propto n_{tw} + \beta$$
$$\theta_j(t) \propto n_{jt} + \alpha$$

where $n_{tw}$ is the number of times word $w$ is assigned to topic $t$, and $n_{jt}$ is the number of times topic $t$ is used in document $j$, with both counts being taken with respect to the final sample $\mathbf{z}$. The topic-word multinomials $\phi_t$ for each topic $t$ are our learned topics; each document-topic multinomial $\theta_d$ represents the prevalence of topics within document $d$.

Table 1: Example learned topic-word multinomials $\phi$ from three different datasets (see Table 5). For each topic $\phi_t$, the five highest-probability words $w$ are shown.

| FT - Topic 1 | | WSJ - Topic 8 | | LA - Topic 94 | |
|---|---|---|---|---|---|
| Word $w$ | $P(w\|z)$ | Word $w$ | $P(w\|z)$ | Word $w$ | $P(w\|z)$ |
| court | 0.080 | technology | 0.094 | gun | 0.058 |
| case | 0.025 | research | 0.054 | weapons | 0.052 |
| legal | 0.024 | high | 0.025 | assault | 0.034 |
| ruling | 0.018 | development | 0.023 | guns | 0.029 |
| appeal | 0.018 | cray | 0.020 | rifles | 0.018 |

Table 2: Features used to determine "best topic word" labels for each topic. The topic-word posterior $P(z = t|w)$ is computed using Bayes Rule and a uniform prior over topics.

| Description | Score |
|---|---|
| Word probability | $f_1(w) = P(w\|z = t)$ |
| Topic posterior | $f_2(w) = P(z = t\|w)$ |
| PMI | $f_3(w) = \sum_{w' \in W_t \backslash w} PMI(w, w')$ |
| Conditional 1 | $f_4(w) = \sum_{w' \in W_t \backslash w} P(w\|w')$ |
| Conditional 2 | $f_5(w) = \sum_{w' \in W_t \backslash w} P(w'\|w)$ |

### 3.3.2 *Topic representation*

Typically, each learned topic-word multinomial $\phi_t$ is presented as a "Top N" list of the most probable words for that topic, as shown for three example learned topics in Table 1. We define the k-argmax operator to yield the $k$ arguments which result in the $k$ largest values for the given function. We use this operator to define the ten most probable words for topic $t$ as $W_t$, given by the following expression with $k = 10$

$$W_t = \underset{w}{\text{k-argmax}}\, \phi_t(w)$$

We apply techniques from recent topic modeling research to improve on this basic representation. Our post-processing of the learned topics has three components: label generation, $n$-gram identification, and capitalization recovery.

For topic labeling, we assume the availability of a *reference corpus* containing themes similar to the target retrieval corpus. Since only raw text is required, this should be considerably easier to obtain than a full ontology, even for specialized domains. For example, a user exploring a corpus related infectious disease outbreaks could obtain a suitable reference corpus by crawling web resources from the United States Centers for Disease Control and Prevention. Since our experiments use general newswire corpora for evaluation, we use Wikipedia[11] as our reference corpus.

We label each topic using a simplified variant of the "Best Topic Word" [62] method. For a given topic $t$, this method selects a single word label from the top ten most probable words $W_t$, using features designed to test how representative each word is of the topic as a whole. We deviate slightly from Lau et al. to avoid relying upon WordNet, selecting the label word by majority vote among five features shown in Table 2 where each feature $f_i$ casts its vote for the highest scoring word and ties are broken arbitrarily. Several of these features are computed from co-occurrence frequencies among words in $W_t$, counted within ten-word sliding windows taken over the reference corpus. Specifically, we compute the pointwise mutual information (PMI) and conditional occurrence probabilities between each pair of words $(w, w')$ as

$$PMI(w, w') = \log \frac{P(w, w')}{P(w)P(w')}$$
$$P(w|w') = \frac{P(w, w')}{P(w')}$$

where $P(w, w')$ is the probability of jointly observing $w$ and $w'$ within a given sliding window, and $P(w)$ is the probability of observing $w$ within a sliding window. Several example labels can be seen in the "label" column of Table 3.

We then identify statistically significant bigrams and trigrams (e.g., "White House", "President Barack Obama") for each topic using an approach based on the Turbo Topics [11] algorithm. This approach considers adjacent word pairs $(w_i, w_{i+1})$ occurring in the same document and assigned to the same topic (i.e., $d_i = d_{i+1}$ and $z_i = z_{i+1}$) and identifies pairs which occur much more often than we would expect by chance alone, proceeding similarly for trigrams. For each topic, we show the topic label along with the most significant trigram, the two most significant bigrams and the four most probable unigrams. Example latent topic representations are shown in Table 3.

Finally, we restore capitalization to the topic $n$-grams before presenting them to the user. As a pre-processing step, all text is converted to lower-case before doing LDA inference. However, the information conveyed by capitalization can ease user interpretation of topics (e.g., by making proper names obvious). For each $n$-gram, we simply count all occurrences of each possible capitalization occurring in the original documents, and present the most frequent version to the user.

[11]http://www.wikipedia.org

Table 3: Topic representations for example high-PMI (coherent) and low-PMI (incoherent) topics.

| PMI | Label | $n$-grams |
|---|---|---|
| 3.09 | jurors | Deputy Dist Atty |
| | | cross examination, closing arguments |
| | | trial, jury, case, testified |
| 1.68 | Petroleum | state oil company |
| | | North Sea, natural gas |
| | | production, exploration, field, energy |
| -0.09 | things | (no trigrams found) |
| | | pretty good, years ago |
| | | ve, ll, time, don |
| -0.03 | sales | (no trigrams found) |
| | | year earlier, Feb Feb |
| | | December, March, month, rose |

### 3.3.3 *Topic selection*

It will typically be necessary to learn at least hundreds of latent topics in order to get suitably fine-grained topics for user feedback. This makes it impractical to present all topics to the user after every query; we therefore must decide which topics to present.

We use the idea of pseudo-relevance feedback [19] by assuming that the top two documents returned by the original query $q$, which we call $D_q$, are relevant. For each of these documents, we consider the top $k = 2$ topics as determined by the topic weights $\theta$ to be *enriched* topics for the user query. This constitutes a natural set of candidates for latent topic feedback, and can be defined as

$$E = \bigcup_{d \in D_q} t \; \theta_d(t).$$

However, we also show the user topics that are *related* to the enriched topic set $E$, but which may themselves not be present in the highly ranked documents. We identify related topics by looking for topics highly likely to co-occur with the enriched topics $E$, using the $T \times T$ topic covariance matrix $\Sigma$ of the estimated $D \times T$ document-topic $\theta$ matrix. Letting $\Sigma(t_1, t_2)$ be the covariance between $P(z = t_1|d)$ and $P(z = t_2|d)$ computed over all documents $d = 1, \ldots, D$, we take the $k = 2$ topics with the highest covariance with each of our enriched topics in $E$. We define this *related* topic set as

$$R = \bigcup_{t \in E} t' \notin E \; \Sigma(t, t').$$

The candidate topics for feedback are the union of the enriched and related topics $E \cup R$, but we perform a final filter before presenting these topics to the user.

One hazard of presenting automatically discovered latent topics to the user is the threat of incoherent "junk" topics which do not seem to have a single clear theme. We filter out these topics using a recently developed topic evaluation method [85, 84] which has been shown to predict human topic quality judgments at nearly the inter-annotator agreement rate. Similar to the topic labeling technique, this method uses PMI values computed over a reference corpus (again, we use Wikipedia), except that we now apply these scores to the topics themselves. We compute the PMI score of a topic $t$ as the average PMI between all pairs of words within the top $k = 10$ most probable words $W_t$

$$PMI(t) = \frac{1}{k(k-1)} \sum_{(w,w') \in W_t} PMI(w, w').$$

Table 3 shows example high-PMI (coherent) and low-PMI (incoherent) latent topics.

We can use these PMI values to avoid confusing users with incoherent topics. Letting $PMI_{25}$ be the $25^{th}$ percentile PMI score among all learned topics, we define our set of "dropped" topics $D$ as

$$D = \{t | t \in E \cup R \text{ and } PMI(t) < PMI_{25}\}.$$

We present the topics in $\{E \cup R\} \setminus D$ to the user alongside the returned documents for the original keywords query $q$. Note that the union operations and final filtering mean that the number of topics actually presented to the user may vary from query to query. Since we consider the top two topics within the top two documents, along with each of their top two related topics, we will present a maximum of $(2 \times 2) + (2 \times 2 \times 2) = 12$ topics, minus set overlaps and PMI-filtered topics.

### 3.3.4 *Query expansion*

If the user selects a topic as relevant, we reformulate the query by combining the top ten most probable words $W_t$ for that topic with the original query $q$. To preserve the intent of the original query, we use the Indri [77] `#weight()` operator to form a weighted combination of the original query keywords and the highly probable latent topic words. The weight parameter $\gamma \in [0, 1]$ controls the trade-off between the original query keywords and the latent topic words. A larger $\gamma$ value places more weight on the new latent topic words, while setting $\gamma = 0$ is equivalent to the original keyword query.

Each of the $N_q$ words in the original query is given weight $(1 - \gamma)/N_q$ and each new topic $t$ word $w$ is given weight $\gamma * \tilde{\phi}_t(w)$, where $\tilde{\phi}$ is the re-normalized topic-word probability

$$\tilde{\phi}_t(w) = \frac{\phi_t(w)}{\sum_{w' \in W_t} \phi_t(w')}.$$

While our implementation uses the Indri query language, it would be straightforward to achieve similar results in other information retrieval systems and frameworks (e.g., by using term boosting in Apache Lucene[12]).

### 3.3.5 *Example*

[12]http://lucene.apache.org/

Table 4: A detailed example of our approach for the query "euro opposition" on the Financial Times (FT) corpus. The strikethrough topic 466 is not presented to the user due to low PMI coherence score. The bolded topic 79 results in improved information retrieval performance versus the baseline query: NDCG15 increases 0.22, NDCG increases 0.07, and MAP increases 0.02. The prominent term "Emu" appears to be an alternate form of the acronym "EMU" commonly used in Financial Times articles.

| Enriched topic | Terms |
|---|---|
| 196 (debate) | Tory Euro sceptics |
| | social chapter, Liberal Democrat |
| | mps, Labour, bill, Commons |
| 404 (ratification) | ratification Maastricht treaty |
| | Poul Schluter, Poul Rasmussen |
| | Danish, vote, Denmark, ec |
| 466 (business) | PERSONAL FILE Born |
| | years ago, past years |
| | man, time, job, career |

(a) Enriched topics $E$.

| Related topic | Terms |
|---|---|
| **79 (Emu)** | economic monetary union |
| | Maastricht treaty, member states |
| | European, Europe, Community, Emu |
| 377 (George) | President George Bush, White House |
| | Mr Clinton, administration |
| | Democratic, Republican, Washington |
| 115 (powers) | de regulation bill |
| | Sunday trading, Queen Speech |
| | law, legislation, government, act |
| 446 (years) | chairman chief executive |
| | managing director, finance director |
| | Sir, board, group, company |
| 431 (cabinet) | Mr John Major |
| | prime minister, Mr Major |
| | party, tory, government, Conservative |

(b) Related topics $R$.

We now walk through an example query for a corpus of news articles from the Financial Times (FT). The query is "euro opposition", and it targets documents discussing opposition to the introduction of the single European currency. The corpus, query, and relevance judgments used here are drawn from our experimental dataset which will be used in Section 3.4. The number of topics used is $T = 500$.

The *enriched* topics $E$ shown in Table 5a consist of three distinct topics: two topics related to the euro debate within the United Kingdom and Denmark, and a confusing topic vaguely centered around "business" which is *dropped* by our PMI filtering. Within this topic, the interesting trigram "PERSONAL FILE Born" arises from brief biographies sometimes found at the bottom of the articles.

High $\theta$ covariance with topics in $E$ is then used to identify the five *related* topics $R$ shown in Table 5b, which deal with various aspects of business and politics. However the appearance of "economic monetary union" and "Europe" in the topic 79 representation appear highly related to the euro currency union, and indeed selecting this topic as feedback improves retrieval results. Selecting topic 79 as user feedback and setting the feedback weight $\gamma = 0.25$, our approach produces an expanded query containing the most probable words from topic 79

```
#weight(0.375 euro, 0.375 opposition,
        0.031 European, ..., 0.015 Emu).
```

Using ground truth document relevance judgments, we can see that documents returned by this expanded query have superior performance on standard information retrieval measures as described in the caption of Table 4. Figure 1 shows the receiver operating characteristic (ROC) curves for the baseline query (dotted) and the expanded topic 79 query (solid). Points on the ROC curve correspond to the true positive rates (TPR) and false positive rates (FPR) for sets of documents returned at different ranking thresholds. Here we consider the true positive (TP) set to be the union of relevant documents found within the top 500 documents returned by both queries. This plot visually depicts a clear improvement in the ranking of relevant documents. An additional benefit is that users are given the opportunity to see and explore different aspects of "euro opposition" such as the political dimension with respect to the United Kingdom.

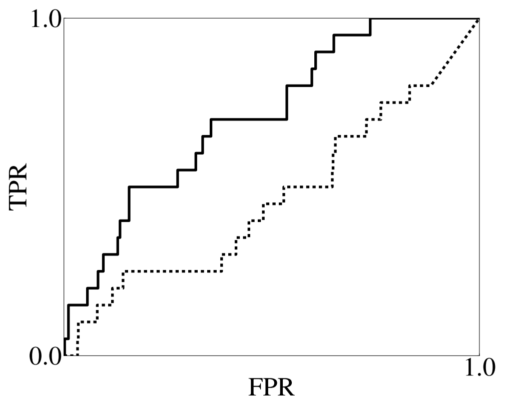


Figure 1: ROC curve of baseline (dashed) versus topic 79 feedback (solid) for the example query "euro opposition".

## 3.4 Experiments

To our knowledge there has been no attempt to use latent topics as a user feedback mechanism in the way we have described. To determine whether our approach could be genuinely useful in practice, we must answer several questions. First, can query expansion with latent topic feedback improve the results of actual queries? While previous work has found that latent topics align well with existing document subject categories [81], it may be that these categories are more "topically coherent" than the relevant result sets for *ad hoc* queries, and therefore more suitable for topic modeling. Second, assuming that for a given query there exists some latent topic which would improve retrieval results, will the topic selection approach described in Section 3.3.3 present it to the user?

Table 5: TREC datasets and queries (known within TREC as "topics") used in experimental evaluations. The number of documents $D$ is given in thousands and $Q$ denotes the number of queries.

| Corpus | Abbrev | $D$ | $Q$ | TREC topics |
|---|---|---|---|---|
| Associated Press | AP | 243 | 100 | 51-150 |
| Financial Times | FT | 210 | 200 | 251-450 |
| Los Angeles Times | LA | 128 | 150 | 301-450 |
| Wall Street Journal | WSJ | 173 | 100 | 51-100<br>151-200 |
| Federal Register | FR | 37 | 150 | 301-450 |
| Foreign Broadcast Information Service | FBIS | 127 | 150 | 301-450 |

Finally, there is a third question which we do not address in this work: if presented with a helpful topic, will a user actually select it? For the following experiments we make the simplifying assumption that the user will always select the most helpful topic (with respect to the information retrieval measure of interest) among those presented. If no topic feedback will improve the set of returned documents, we assume the user will not provide topic feedback.

### *3.4.1 Experiment setup*

While the ultimate goal of this work is to improve search and navigation under the specialized conditions described in Section 3.1, we evaluate our approach by conducting information retrieval experiments on several benchmark datasets from the Text REtrieval Conference (TREC) [114], using Wikipedia as a reference corpus. Each datasets consists of a corpus of text documents, a set of queries, and relevance judgments for each query. For each query, the individual words in the the title field are used as the baseline keyword query (e.g., "Industrial Espionage" is broken up into "Industrial", "Espionage"). Table 5 shows dataset details.

For each corpus, we first apply the LDA model to learn a set of latent topics, using the MALLET topic modeling toolkit [76]. We pre-process documents by downcasing, removing numbers and punctuation, applying a standard stopword list, and finally filtering out rarely occurring terms to yield vocabulary sizes of between 10,000 and 20,000 terms. We run parallelized collapsed Gibbs inference for 1,000 samples, re-estimating the document-topic hyperparameter $\alpha$ every 25 samples. We learn $T = 500$ topics for each corpus in our experimental dataset, except $T = 250$ for the significantly smaller Federal Register (FR) corpus.

For all queries, we use the Galago [26] information retrieval system with default settings to retrieve 500 documents. Galago uses a query language and retrieval model based on Indri [77]. For the topic-expanded queries we set $\gamma = 0.25$, based on trial-and-error experimentation on held-aside preliminary development datasets.

### *3.4.2 Results*

We calculate improvement over the baseline query with respect to three information retrieval measures [26]: mean average precision (MAP), normalized discounted cumulative gain (NDCG), and NDCG calculated with the first 15 results only (NDCG15). These quantitative results are shown in Table 6, along with the average number of feedback candidate topics shown to the user by our topic selection technique (fewer than eight topics per query).

We now return to the experimental questions we had set out to answer. These results demonstrate that latent topic feedback can indeed improve information retrieval results. Across evaluation measures, the results of approximately 40% of queries can be improved by latent topic feedback. However, these gains are irrelevant if we cannot identify potentially helpful topics and present them to the user. Again across measures, we see that our topic selection approach is able present a helpful topic for more than 40% of the queries for which there exists at least one helpful topic. Doing the rough arithmetic, this means that for about 16% of the queries in our experiment the user would be presented with at least one latent topic which would improve the relevance of the returned documents. Furthermore, we stress that even for the "missed" queries where presented topics do not provide quantitative relevance improvement, the corpus theme information conveyed may still be beneficial.

To give a better feel for the nature of these results, Figure 2 shows six queries along with helpful topics which were selected for presentation by our approach. In all cases, the connection between the topic and the query is fairly clear, resulting in gains across retrieval performance measures and visible improvement on ROC curves.

### *3.4.3 Analysis*

First, we observe that for most queries (roughly 60%), there did not exist a single latent topic for which feedback would enhance information retrieval results. From manual inspection, this can occur because either no learned topic is well-aligned with the relevant documents, or because the results of the original query are good and difficult to improve upon.

Second, for queries where there exists one or more topics which *would* improve results, roughly 60% of the time our topic selection approach fails to select them. Minor variations on our topic selection method (i.e., showing more topics) did not correct this – many of the "missed" topics are not even close to making the cutoff. Manual investigations reveal that, interestingly, these topics often appear to be helpful *because* they are somewhat "distant" from the original query and the top few baseline documents returned. Attempts to predict topic feedback gain using linear or logistic regression and features such as $P(\text{query}|\phi_t)$ were unsuccessful, although more sophisticated approaches or richer features could possibly be applied.

It is also instructive to further examine the impact of two key aspects of our topic selection procedure: the inclusion of related topics and the exclusion of incoherent topics. For simplicity we will discuss NDCG15 measurements, but similar results hold for MAP and NDCG. Our selection approach recovers helpful topics for 133 out of 850 queries (15.6%) while presenting an average of 7.76 topics to the user for each query.

If we do *not* use PMI to filter out topics suspected of being incoherent, the number of topics shown per query rises to 9.79, but the number of queries for which helpful topics are presented only increases to 143 out of 850 (16.8%). The presence of incoherent topics may also impose cognitive burdens on the user, and it is uncertain whether users would be able to successfully identify incoherent topics for feedback.

Table 6: Improvement from simulated latent topic feedback calculated only over queries where feedback improves performance. The "avg shown" column indicates the average number of topics actually shown to the user as a result of the topic selection procedure described in Section 3.3.3. For each query and evaluation measure, the "imprv" column shows the number of queries for which there *exists* at least one latent topic which improves performance, "found" shows the number of queries for which a helpful topic is *actually presented* to the user by our selection scheme, and "avg gain" shows the mean improvement when a helpful topic is presented to the user.

| | | | NCDG15 | | | NCDG | | | MAP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Corpus | $Q$ | avg shown | imprv | found | avg gain | imprv | found | avg gain | imprv | found | avg gain |
| AP | 100 | 7.79 | 32 | 16 | 0.165 | 32 | 21 | 0.093 | 31 | 20 | 0.037 |
| FT | 200 | 7.47 | 97 | 43 | 0.238 | 138 | 80 | 0.134 | 137 | 72 | 0.041 |
| LA | 150 | 8.65 | 79 | 27 | 0.090 | 81 | 27 | 0.070 | 82 | 29 | 0.027 |
| WSJ | 100 | 7.73 | 29 | 16 | 0.205 | 30 | 18 | 0.050 | 29 | 18 | 0.026 |
| FR | 150 | 7.22 | 26 | 10 | 0.131 | 39 | 13 | 0.034 | 39 | 11 | 0.024 |
| FBIS | 150 | 7.78 | 62 | 21 | 0.163 | 64 | 25 | 0.037 | 67 | 29 | 0.024 |

If we were to omit the related topics $R$, it would decrease the average number of topics shown to 2.70, but it would decrease substantially the number of queries for which a helpful topic is presented, down to 93 out of 850 (10.9%). Also, we note that the presentation of related topics is potentially useful for exploratory corpus search, giving the user information about corpus themes "adjacent" to the topics present in returned documents.

Taken together, these findings suggest that our topic selection procedure is reasonable. The inclusion of related topics considerably increases the number of queries for which we present helpful topics while presenting novel and possibly interesting corpus themes. The filtering of suspect low-PMI topics does not discard many helpful topics, and should spare users the ordeal of interpreting ill-defined topics.

## 3.5 Discussion

In this work we have developed a novel technique for improving text corpus search and navigation in difficult settings where we do not have access to metadata, rich lexical resources, or large user populations. This is an important problem because these conditions make information retrieval more difficult, and are applicable within organizations that have large quantities of *internal* text documents which they wish to explore, analyze, and exploit.

To enhance search and exploration capabilities in this scenario, we have developed an approach that gives users the ability to provide feedback at the latent topic level. We leverage recent advances in latent topic modeling in order to construct meaningful representations of latent topics while filtering out incoherent "junk topics". We propose a mechanism for deciding on a manageably small set of topics to present to the user, as well as a method for constructing expanded queries based on user topic feedback. Quantitative results on benchmark TREC datasets show that this technique can result in major improvements for a non-trivial proportion of queries. Furthermore, the presentation of enriched and related topics alongside search results can help to deliver insights about corpus themes, which may be beneficial for knowledge discovery as well.

One potential obstacle to this approach is the scalability bottleneck presented by LDA topic inference. However, two factors act to ameliorate these concerns. First, topics can be inferred "offline" in advance; we do not need to do any expensive inference at query-time. Second, there have been significant recent advances along multiple fronts in scalable LDA inference. A distributed system developed at Yahoo! is reported to process 42,000 documents per hour [103]. Alternatively, an online inference algorithm for LDA [47] promises both improved scalability and a principled means of updating topics to reflect new documents. In practice, a hybrid system could update topics in an online fashion as documents are received, periodically performing distributed batch inference to refresh the learned topics.

## 3.6 Future work

There are several promising directions in which to extend this approach. Two obvious areas for improvement are increasing the proportion of queries for which a helpful topic exists and improving the selection method for presenting helpful topics to the user.

It may be possible to improve the alignment between learned topics and user queries by the use of more sophisticated topic models such as the Pachinko Allocation Model (PAM) [67]. While these models were *not* found to be helpful for document smoothing [124], rich hierarchical topics may be beneficial when combined with the *explicit* user feedback present in our approach. Our approach could also exploit prior information such as predefined concepts by using topic model variants which can incorporate domain knowledge [24, 5].

However, learning finer-grained topics can only increase the importance of carefully choosing which topics to show the user. Here it may be instructive to consider the large body of research on "learning to rank" [71], as well as recent work in facet selection [68, 57].

The query expansion mechanism is another potential target for extension. If our underlying information retrieval system supports phrase search terms (e.g., "White House"), it may be helpful to directly use discovered $n$-grams as well.

Further work could also compare the use of topics for explicit feedback in this work versus the implicit use of topics to improve document language models in prior work [118]. It may be that the two techniques could be combined profitably, with some topics being more suitable for explicit feedback while others are better used for smoothing.

Finally, another important step is to validate our user model assumptions. One approach may be to directly evaluate information retrieval performance using actual user feedback, for example via Amazon Mechanical Turk [128]. It may also be interesting to explore the relationship between topic presentation (e.g., topic labeling strategies, whether to display $n$-grams) and user behavior.

## 4. A FRAMEWORK FOR PERSONALIZED AND COLLABORATIVE CLUSTERING OF SEARCH RESULTS

The way search results are organized and presented has a direct and significant impact on the utility of search engines. The common strategy has been using a flat ranked list, which works fine for homogeneous search results.

However, queries are inherently ambiguous and search results are often diverse with multiple senses. With a list presentation, the results on different sub-topics of a query will be mixed together. The user has to sift through many irrelevant results to locate those relevant ones.

With the rapid growth in the scale of the Web, queries have become more ambiguous than ever. For example, there are more than 20 entries in Wikipedia for different well-known individuals under the name of Jim Gray [13] and 74 entries for Michael Smith [14]. Consequently, the diversity of search results has increased to the point that we must consider alternative presentations, providing additional structure to flat lists so as to effectively minimize browsing effort and alleviate information overload [45, 91, 125, 20]. Over the years clustering has been accepted as the most promising alternative.

Clustering is the process of organizing objects into groups or clusters that exhibit internal cohesion and external isolation. Based on the common observation that it is much easier to scan a few topic-coherent groups than many individual documents, clustering can be used to categorize a long list of disparate search results into a few clusters such that each cluster represents a homogeneous sub-topic of the query. Meaningfully labeled, these clusters form a topic-wise non-predefined, faceted search interface, allowing the user to quickly locate relevant and interesting results. There is good evidence that clustering improves user experience and search result quality [74].

Given the significant potential benefits, search result clustering has received increasing attention in recent years from the communities of information retrieval, Web search and data mining. Many clustering algorithms have been proposed [45, 91, 125, 126, 127, 59, 117, 65]. In the industry, well-known cluster-based commercial search engines include Clusty[15], iBoogie[16] and CarrotSearch[17].

Despite the high promise of the approach and a decade of endeavor, cluster-based search engines have not gained prominent popularity, evident by Clusty's Alexa rank [48]. This is because clustering is known to be a hard problem, and search result clustering is particularly hard due to its high dimensionality, complex semantics and unique additional requirements beyond traditional clustering.

As emphasized in [117] and [20], the primary focus of search result clustering is NOT to produce optimal clusters, an objective that has been pursued for decades for traditional clustering with many successful automatic algorithms. Search result clustering is a highly user-centric task with *two unique additional requirements*. First, clusters must form interesting sub-topics or facets from the user's perspective. Second, clusters must be assigned informative, expressive, meaningful and concise labels. Automatic algorithms often fail to fulfill the human factors in the objectives of search result clustering, generating meaningless, awkward or nonsense cluster labels [20].

In this paper, we explore a completely different direction in tackling the problem of clustering search results, utilizing the power of direct user intervention and mass-collaboration. We introduce `ClusteringWiki`, the first prototype and framework for personalized clustering that allows direct *user* editing of the *clustering results*. This is in sharp contrast with existing approaches that innovate on the *automatic* algorithmic *clustering procedure*.

In `ClusteringWiki`, the user can edit and annotate the membership, structure and labels of clusters through a Wiki interface to personalize her search result presentation. Edits and annotations can be implicitly shared among users as a mass-collaborative way of improving search result organization and search engine utility. This approach is in the same spirit of the current trends in the Web, like Web 2.0, semantic web, personalization, social tagging and mass collaboration.

Clustering algorithms fall into two categories: partitioning and hierarchical. Regarding clustering results, however, a hierarchical presentation generalizes a flat partition. Based on this observation, `ClusteringWiki` handles both clustering methods smoothly by providing editing facilities for cluster hierarchies and treating partitions as a special case. In practice, hierarchical methods are advantageous in clustering search results because they construct a topic hierarchy that allows the user to easily navigate search results at different levels of granularity.

Figure 3 shows a snapshot of `ClusteringWiki`[18]. The left-hand *label panel* presents a hierarchy of cluster labels. The right-hand *result panel* presents search results for a chosen cluster label. A logged-in user can edit the current clusters by creating, deleting, modifying, moving or copying nodes in the cluster tree. Each edit will be validated against a set of predefined consistency constraints before being stored.

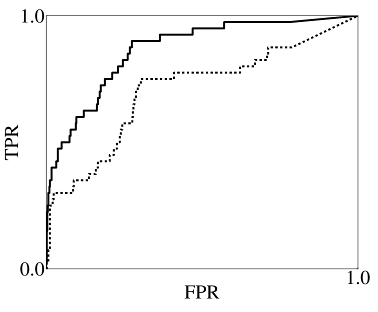
Designing and implementing `ClusteringWiki` poses non-trivial technical challenges. User edits represent user preferences or constraints that should be respected and enforced next time the same query is issued. Query processing is time-critical, thus efficiency must be given high priority in maintaining and enforcing user preferences. Moreover, complications also come from the dynamic nature of search results that constantly change over time.

Cluster editing takes user effort. It is essential that such user effort can be properly reused. `ClusteringWiki` considers two kinds of reuse scenarios, *preference transfer* and *preference sharing*. The former transfers user preferences from one query to similar ones, e.g., from "David J. Dewitt" to "David Dewitt". The latter aggregates and shares clustering preferences among users. Proper aggregation allows users to collaborate at a mass scale and "vote" for the best search result clustering presentation.

[13]http://en.wikipedia.org/wiki/James$_{Gray}$
[14]http://en.wikipedia.org/wiki/Michael$_{smith}$
[15]www.clusty.com
[16]www.iboogie.com
[17]carrotsearch.com
[18]dmlab.cs.txstate.edu/ClusteringWiki/index.html.

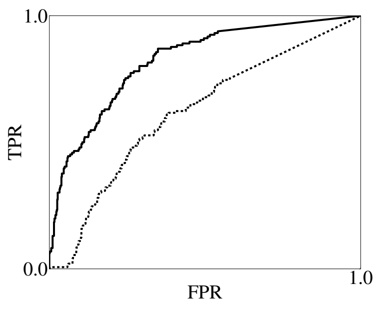
| FBIS query 426<br>“law enforcement dogs” |
|---|
| Topic 321 (heroin)<br>seized kg cocaine<br>drug traffickers, kg heroin<br>police, arrested, drugs, marijuana |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.299 | +0.065 | +0.046 |



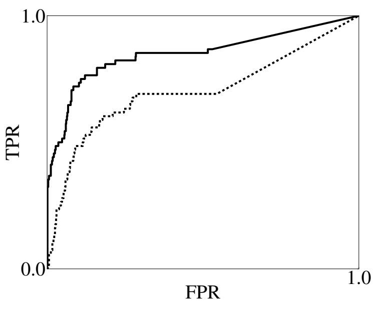
| FBIS query 450<br>“King Hussein, peace” |
|---|
| Topic 293 (Amman)<br>Majesty King Husayn<br>al Aqabah, peace process<br>Jordan, Jordanian, Amman, Arab |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.708 | +0.175 | +0.171 |



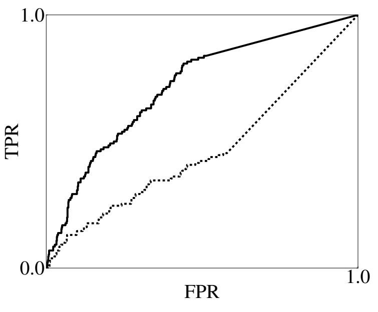
| WSJ query 86<br>“bank failures” |
|---|
| Topic 444 (FDIC)<br>Federal Deposit Insurance<br>William Seidman, Insurance Corp<br>banks, bank, FDIC, banking |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.602 | +0.121 | +0.110 |



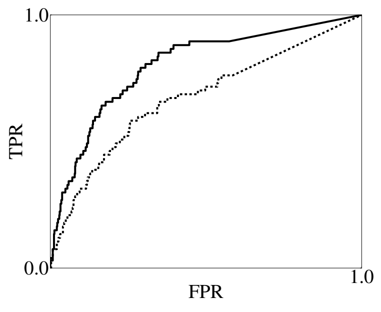
| AP query 127<br>“U.S.-U.S.S.R. Arms<br>Control Agreements” |
|---|
| Topic 232 (missile)<br>Strategic Defense Initiative<br>United States, arms control<br>treaty, nuclear, missiles, range |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.296 | +0.209 | +0.105 |



| AP query 135<br>“Possible Contributions of<br>Gene Mapping to Medicine” |
|---|
| Topic 325 (called)<br>British journal Nature<br>immune system, genetically engineered<br>cells, research, researchers, scientists |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.147 | +0.040 | +0.019 |



| AP query 113<br>“New Space Satellite<br>Applications” |
|---|
| Topic 237 (communications)<br>European Space Agency<br>Air Force, Cape Canaveral<br>satellite, launch, rocket, satellites |

| NDCG15 | NDCG | MAP |
|---|---|---|
| +0.237 | +0.033 | +0.007 |

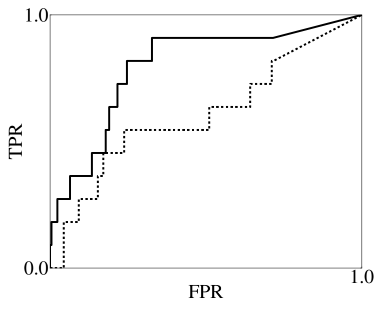
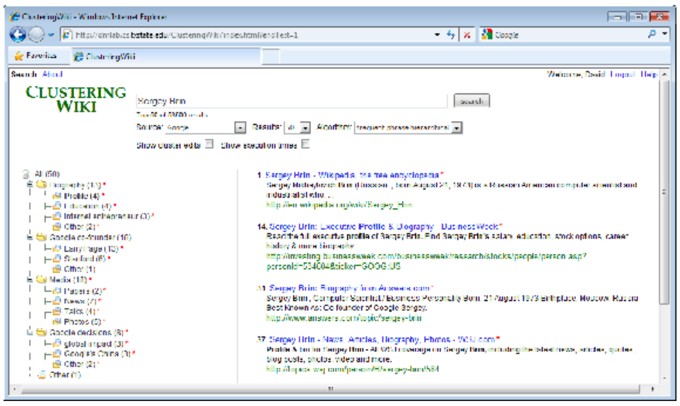


Figure 2: Six example queries with helpful topics and ROC curves. For each ROC curve, the set of true positive (TP) relevant documents is considered to be the union of the relevant documents discovered (i.e., ranked within the top 500) by the baseline query (dashed line) and the expanded query that incorporates latent topic feedback (solid line).

In social tagging, or collaborative tagging, users annotate Web objects, and such personal annotations can be used to collectively classify and find information. `ClusteringWiki` extends conventional tagging by allowing tagging of structured objects, which are clusters of search results organized in a hierarchy.

**Contributions.**

- We introduce `ClusteringWiki`, the first framework for personalized clustering in the context of search result organization. Unlike existing methods that innovate on the automatic clustering procedure, it allows direct user editing of the clustering re-



http://dmlab.cs.txstate.edu/ClusteringWiki/index.html#nTest-1

Sergey Brin

Source: Google   Results: 50   Algorithm: frequent phrase hierarchical

1. Sergey Brin - Wikipedia, the free encyclopedia
Sergey Mikhaylovich Brin (Russian: , born August 21, 1973) is a Russian American computer scientist and industrialist who ...
http://en.wikipedia.org/wiki/Sergey_Brin

14. Sergey Brin: Executive Profile & Biography - BusinessWeek
Read the full executive profile of Sergey Brin. Find Sergey Brin's salary, education, stock options, career history & more Biography
http://investing.businessweek.com/businessweek/research/stocks/people/person.asp?personId=534604&ticker=GOOG:US

31. Sergey Brin: Biography from Answers.com
Sergey Brin, Computer Scientist / Business Personality Born: 21 August 1973 Birthplace: Moscow, Russia Best Known As: Co-founder of Google Sergey...
http://www.answers.com/topic/sergey-brin

37. Sergey Brin - News, Articles, Biography, Photos - WSJ.com
Profile & bio for Sergey Brin - All WSJ coverage on Sergey Brin, including the latest news, articles, quotes, blog posts, photos, video and more.
http://topics.wsj.com/person/B/sergey-brin/584

Figure 3: Snapshot of ClusteringWiki.

sults through a Wiki interface.

- In ClusteringWiki, user preferences are reused among similar queries. They are also aggregated and shared among users as a mass-collaborative way of improving search result organization and search engine utility.

- We implement a prototype for ClusteringWiki, perform experimental evaluation and a user study, and maintain the prototype as a public Web service.

## 4.1 Related Work

**Clustering.** Clustering is the process of organizing objects into groups or clusters so that objects in the same cluster are as similar as possible, and objects in different clusters are as dissimilar as possible. Clustering algorithms fall into two main categories, partitioning and hierarchical. Partitioning algorithms, such as $k$-means [73], produce a flat partition of objects without any explicit structure that relate clusters to each other. Hierarchical algorithms, on the other hand, produce a more informative hierarchy of clusters called a dendrogram. Hierarchical algorithms are either agglomerative (bottom-up) such as AGNES [55], or divisive (top-down) such as DIANA [55].

**Clustering in IR.** As a common data analysis technique, clustering has a wide array of applications in machine learning, data mining, pattern recognition, information retrieval, image analysis and bioinformatics [49, 32]. In information retrieval and Web search, document clustering was initially proposed to improve search performance by validating the *cluster hypothesis*, which states that documents in the same cluster behave similarly with respect to relevance to information needs [97].

In recent years, clustering has been used to organize search results, creating a cluster-based search interface as an alternative presentation to the ranked list interface. The list interface works fine for most navigational queries, but is less effective for informational queries, which account for the majority of Web queries [17, 98]. In addition, the growing scale of the Web and diversity of search results have rendered the list interface increasingly inadequate. Research has shown that the cluster interface improves user experience and search result quality [45, 126, 112, 53].

**Search result clustering.** One way of creating a cluster interface is to construct a static, off-line, pre-retrieval clustering of the entire document collection. However, this approach is ineffective because it is based on features that are frequent in the entire collection but irrelevant to the particular query [39, 100, 20]. It has been shown that query-specific, on-line, post-retrieval clustering, i.e., clustering search results, produces much superior results [45].

Scatter/Gather [45, 91] was an early cluster-based document browsing method that performs post-retrieval clustering on top-ranked documents returned from a traditional information retrieval system. The Grouper system [125, 126] (retired in 2000) introduced the well-known Suffix Tree Clustering (STC) algorithm that groups Web search results into clusters labeled by phrases extracted from snippets. It was also shown that using snippets is as effective as using whole documents. Carrot2 (www.carrot2.org) is an open source search result clustering engine that embeds STC as well as Lingo [87], a clustering algorithm based on singular value decomposition.

Other related work from the Web, IR and data mining communities exists. [127] explored supervised learning for extracting meaningful phrases from snippets, which are then used to group search results. [59] proposed a monothetic algorithm, where a single feature is used to assign documents to clusters and generate cluster labels. [117] investigated using past query history in order to better organize search results for future queries. [65] studied search result clustering for object-level search engines that automatically extract and integrate information on Web objects. [20] surveyed Web clustering engines and algorithms.

While all these methods focus on improvement in the automatic algorithmic procedure of clustering, ClusteringWiki employs a Wiki interface that allows direct user editing of the clustering results.

**Clustering with user intervention.** In machine learning, clustering is referred to as unsupervised learning. However, similar to ClusteringWiki, there are a few clustering frameworks that involve an active user role, in particular, semi-supervised clustering [8, 25] and interactive clustering [115, 50, 9] These frameworks are also motivated by the fact that clustering is too complex, and it is necessary to open the “black box" of the clustering procedure for easy understanding, steering and focusing. However, they differ from ClusteringWiki in that their focus is still on the clustering procedure, where they adopt a constraint clustering approach by transforming user feedback and domain knowledge into constraints (e.g., must-links and cannot-links) that are incorporated into the clustering procedure.

**Search result annotation.** Prototypes that allow user editing and annotation of search results exist. For example, U Rank by Microsoft [19] and Searchwiki by Google [20]. Rants [36] implemented a prototype with additional interesting features including the incorporation of both absolute and relative user preferences. Similar to ClusteringWiki, these works pursue personalization as well as a mass-collaborative way of improving search engine utility. The difference is that they use the traditional flat list, instead of cluster-based, search interface.

**Tagging and social search.** Social tagging, or collaborative tagging, allows users to create and associate objects with tags as a means of annotating and categorizing content. While users are primarily interested in tagging for their personal use, tags in a community collection tend to stabilize into power law distributions [43]. Collaborative tagging systems leverage this property to derive folksonomies and improve search [121]. In ClusteringWiki users tag clusters to organize search results, and the tags can be shared and utilized in the same way as in collaborative tagging. Since clusters are organized in a hierarchy, ClusteringWiki extends conventional tagging by allowing tagging of structured objects. Similar to tag suggestion in social tagging, the base clustering algorithm in ClusteringWiki provides suggested phrases for tagging clusters.

Social search is a mass-collaborative way of improving search performance. In contrast to established algorithmic or machine-based approaches, social search determines the relevance of search results by considering the content created or touched by users in the social graph. Example forms of user contributions include shared bookmarks or tagging of content with descriptive labels. Currently there are more than 40 such people-powered or community-powered social search engines, including Eurekster Swiki [21], Mahalo [22], Wikia [23], and Google social search [24]. Mass collaboration, or crowdsourcing, systems on the Web are categorized and discussed in [30].

## 4.2 Overview

In this section, we overview the main architecture and design principles of ClusteringWiki. Figure 4 shows the two key modules. The *query processing module* takes a query $q$ and a set of stored user preferences as input to produce a cluster tree $T$ that respects the preferences. The *cluster editing module* takes a cluster tree $T$ and a user edit $e$ as input to create/update a set of stored user preferences. Each user editing session usually involves a series of edits. The processing-editing cycle recurs over time.

**Query processing.** ClusteringWiki takes a query $q$ from a user $u$ and retrieves the search results $R$ from a data source (e.g., Google). Then, it clusters $R$ with a default clustering algorithm (e.g., frequent phrase hierarchical) to produce an initial cluster tree $T_{init}$. Then, it applies $P$, an applicable set of stored user preferences, to $T_{init}$ and presents a modified cluster tree $T$ that respects $P$.

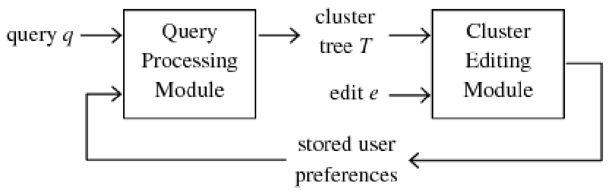


Figure 4: Main architecture of ClusteringWiki.

Note that ClusteringWiki performs clustering. The modification should not alter $R$, the input data.

If the user $u$ is logged-in, $P$ will be set to $P_{q,u}$, a set of preferences for $q$ previously specified by $u$. In case $P_{q,u} = \emptyset$, $P_{q',u}$ will be used on condition that $q'$ is sufficiently close to $q$. If the user $u$ is not logged-in, $P$ will be set to $P_{q,U}$, a set of aggregated preferences for $q$ previously specified by all users. In case $P_{q,U} = \emptyset$, $P_{q',U}$ will be used on condition that $q'$ is sufficiently close to $q$.

In the cluster tree $T$, the internal nodes, i.e., non-leaf nodes, contain cluster labels and are presented on the left-hand *label panel*. Each label is a set of keywords. The leaf nodes contain search results, and the leaf nodes for a selected label are presented on the right-hand *result panel*. A search result can appear multiple times in $T$. The root of $T$ represents the query $q$ itself and is always labeled with *All*. When it is chosen, all search results will be presented on the result panel. Labels other than *All* represent the various, possibly overlapping, sub-topics of $q$. When there is no ambiguity, *internal node*, *label node*, *cluster label* and *label* are used interchangeably in the paper. Similarly, *leaf node*, *result node*, *search result* and *result* are used interchangeably.

**Cluster editing.** If logged-in, a user $u$ can edit the cluster tree $T$ for query $q$ by creating, deleting, modifying, moving or copying nodes. User edits will be validated against a set $C$ of consistency constraints before being written to $P_{q,u}$.

The set $C$ contains predefined constraints that are specified on, for example, the size of clusters, the height of the tree and the length of labels. These constraints exist to maintain a favorable user interface for fast and intuitive navigation. The cluster tree $T$ is *consistent* if it satisfies all the constraints in $C$.

By combining preferences in $P_{q,u}$ for all users who have edited the cluster tree $T$ for query $q$, we obtain $P_{q,U}$, a set of aggregated preferences for query $q$. We use $P_u$ to denote the collection of clustering preferences by user $u$ for all queries, which is a set of sets of preferences such that $\forall q, P_{q,u} \in P_u$. We also use $P_U$ to denote the collection of aggregated preferences by all users for all queries, which is a set of sets of aggregated preferences such that $\forall q, P_{q,U} \in P_U$. $P_u$ and $P_U$ are maintained over time and used by ClusteringWiki in processing queries for the user $u$.

[19] research.microsoft.com/en-us/projects/urank
[20] googleblog.blogspot.com/2008/11/searchwiki-make-search-your-own.html
[21] www.eurekster.com
[22] www.mahalo.com
[23] answers.wikia.com/wiki/Wikianswers
[24] googleblog.blogspot.com/2009/10/introducing-google-social-search-i.html

**Design principles.** In a search result clustering engine, there are significant uncertainties from the data to the clustering algorithm. Wiki-facilitated personalization further adds substantial complications. Simplicity should be a key principle in designing such a complex system. `ClusteringWiki` adopts a simple yet powerful *path approach*.

With this approach, a cluster tree $T$ is decomposed into a set of root-to-leaf *paths* that serve as independent editing components. A path always starts with $All$ (root) and ends with some search result (leaf). In `ClusteringWiki`, maintenance, aggregation and enforcement of user preferences are based on simple path arithmetic. Moreover, the path approach is sufficiently powerful, being able to handle the finest user preference for a cluster tree.

In particular, each edit of $T$ can be interpreted as operations on one or more paths. There are two primitive operations on a path $p$, *insertion* of $p$ and *deletion* of $p$. A modification of $p$ to $p'$ is simply a deletion of $p$ followed by an insertion of $p'$.

For each user $u$ and each query $q$, `ClusteringWiki` maintains a set of paths $P_{q,u}$ representing the user edits from $u$ for query $q$. Each path $p \in P_{q,u}$ can be either *positive* or *negative*. A positive path $p$ represents an insertion of $p$, meaning that the user prefers to have $p$ in $T$. A negative path $-p$ represents a deletion of $p$, meaning that the user prefers not to have $p$ in $T$. Two *opposite* paths $p$ and $-p$ will cancel each other out. The paths in $P_{q,u}$ may be added from multiple editing sessions at different times.

To aggregate user preferences for query $q$, `ClusteringWiki` first combines the paths in all $P_{q,u}$, $u \in U$, where $U$ is the set of users who have edited the cluster tree of $q$. Then, certain statistically significant paths are selected and stored in $P_{q,U}$.

Suppose in processing query $q$, $P$ is identified as the applicable set of paths to enforce. `ClusteringWiki` first combines the paths in $P$ and the paths in $T_{init}$, where $T_{init}$ is the initial cluster tree. Then, it presents the combined paths as a tree, which is the cluster tree $T$. The combination is straightforward. For each positive $p \in P$, if $p \notin T_{init}$, add $p$ to $T_{init}$. For each negative $p \in P$, if $p \in T_{init}$, remove $p$ from $T_{init}$.

**Reproducibility.** It is easy to verify that `ClusteringWiki` has the property of reproducing edited cluster trees. In particular, after a series of user edits on $T_{init}$ to produce $T$, if $T_{init}$ remains the same in a subsequent query, exactly the same $T$ will be produced after enforcing the stored user preferences generated from the user edits on $T_{init}$.

## 4.3 Framework

In this section, we introduce the `ClusteringWiki` framework in detail. In particular, we present the algorithms for the query processing and cluster editing modules and explain their main components.

### *4.3.1 Query Processing*

Algorithm 1 presents the pseudocode for the query processing algorithm of `ClusteringWiki`. In the input, $P_u$ and $P_U$ are used instead of $P_{q,u}$ and $P_{q,U}$ for preference transfer purposes. In processing query $q$, it is likely that $P_{q,u} = \emptyset$ or $P_{q,U} = \emptyset$; then some applicable $P_{q',u} \in P_u$ or $P_{q',U} \in P_U$ can be used. The creation and maintenance of such user preferences will be discussed in Section 4.3.2. The output of the algorithm is a consistent cluster tree $T$.

**Algorithm 1** *Query processing*

```
indent = 1em
Input: $q$, $u$, $C$, $P_u$ and $P_U$: $q$ is a query. $u$ is a user. $C$ is a set
    of consistency constraints. $P_u$ is a collection of preferences by
    user $u$ for all queries, where $\forall q, P_{q,u} \in P_u$. $P_U$ is a collection
    of aggregated preferences for all queries, where $\forall q, P_{q,U} \in P_U$.
Output: $T$: a consistent cluster tree for the search results of query $q$.
 1: retrieve a set $R$ of search results for query $q$;
 2: cluster $R$ to obtain an initial cluster tree $T_{init}$;
 3: $P \leftarrow \emptyset$; //$P$ is the set of paths to be enforced on $T_{init}$
 4: if ($u$ is logged-in) then
 5:     $q' \leftarrow Trans(q, u)$;
 6:     if ($q' \neq NULL$) then
 7:         $P \leftarrow P_{q',u}$; //use applicable personal preferences
 8:     end if
 9: else
10:     $q' \leftarrow Trans(q, U)$;
11:     if ($q' \neq NULL$) then
12:         $P \leftarrow P_{q',U}$; //use applicable aggregated preferences
13:     end if
14: end if
15: $T \leftarrow T_{init}$; //initialize $T$, the cluster tree to present
16: clean $P$; //remove $p \in P$ if its result node is not in $R$
17: for each $p \in P$
18:     if ($p$ is positive) then
19:         $T \leftarrow T \cup \{p\}$; //add a preferred path
20:     else
21:         $T \leftarrow T - \{p\}$; //remove a non-preferred path
22:     end if
23: end for
24: $trim(T, C)$; //make $T$ consistent
25: $present(T)$; //present the set of paths in $T$ as a tree
```

**Retrieving search results.** Line 1 retrieves a set $R$ of search results for query $q$ from a chosen data source. The size of $R$ is set to 50 by default and adjustable to up to 500. The available data sources include Google and Yahoo! Search APIs among others (see Section 4.4 for details). `ClusteringWiki` retrieves the results via *multi-threaded parallel requests*, which are much faster than sequential requests.

The combined titles and snippets of search results retrieved from the sources are preprocessed. In order to extract phrases, we implemented our own tokenizer that identifies whether a token is a word, numeric, punctuation mark, capitalized, all caps, etc. We then remove non-textual tokens and stop words, using the stop word list from the Apache Snowball package [25]. The tokens are then stemmed using the Porter [26] algorithm and indexed as terms. For each term, document frequency and collection frequency are computed and stored. A numeric id is also assigned to each term in the document collection in order to efficiently calculate document similarity, identify frequent phrases, etc.

[25]www.docjar.com/html/api/org/apache/lucene/analysis/snowball/SnowballAn
[26]tartarus.org/ martin/PorterStemmer/

**Building initial tree.** Line 2 builds an initial cluster tree $T_{init}$ with a built-in clustering algorithm. `ClusteringWiki` provides 4 such algorithms: $k$-means flat, $k$-means hierarchical, frequent phrase flat and frequent phrase hierarchical. The hierarchical algorithms recursively apply their flat counterparts in a top-down manner to large clusters.

The $k$-means algorithms follow a strategy that generates clusters before labels. They use a simple approach to generate cluster labels from titles of search results that are the closest to cluster centers. In order to produce stable clusters, the typical randomness in $k$-means due to the random selection of initial cluster centers is removed. The parameter $k$ is heuristically determined based on the size of the input.

The frequent phrase algorithms follow a strategy that generates labels before clusters. They first identify frequent phrases using a suffix tree built in linear time by Ukkonen's algorithm. Then they select labels from the frequent phrases using a greedy set cover heuristic, where at each step a frequent phrase covering the most uncovered search results is selected until the whole cluster is covered or no frequent phrases remain. Then they assign each search result $r$ to a label $L$ if $r$ contains the keywords in $L$. Uncovered search results are added to a special cluster labeled *Other*. These algorithms are able to generate very meaningful cluster labels with a couple of heuristics. For example, a sublabel cannot be a subset of a superlabel, in which case the sublabel is redundant.

`ClusteringWiki` smoothly handles flat clustering by treating partitions as a special case of trees. The built-in clustering algorithms are meant to serve their basic functions. The focus of the paper is not to produce, but to modify, the initial cluster trees.

**Determining applicable preferences.** Lines 3 $\sim$ 14 determine $P$, a set of applicable paths to be enforced on $T_{init}$. Two cases are considered. If the user $u$ is logged-in, $P$ will use some set from $P_u$ representing personal preferences of $u$ (lines 4 $\sim$ 8). Otherwise, $P$ will use some set from $P_U$ representing aggregated preferences (lines 9 $\sim$ 14). The subroutine $Trans()$ determines the actual set to use if any.

The pseudocode of $Trans(q, u)$ is presented in Algorithm 2. Given a user $u$ and a query $q$, it returns a query $q'$, whose preferences stored in $P_{q',u}$ are applicable to query $q$. In the subroutine, two similarity measures are used. *Term similarity*, $termSim(q, q')$, is the Jaccard coefficient that compares the terms of $q$ and $q'$. *Result similarity*, $resultSim(q, q')$, is the Jaccard coefficient that compares the URLs of the top $k$ (e.g., $k = 10$) results of $q$ and $q'$. This calculation requires that the URLs of the top $k$ results for $q'$ be stored.

To validate $q'$, both similarity values need to pass their respective thresholds $\delta_{ts}$ and $\delta_{rs}$. Obviously, the bigger the thresholds, the more conservative the transfer. Setting the thresholds to 1 shuts down preference transfer. Instead of thresholding, another reasonable way of validation is to provide a ranked list of similar queries and ask the user for confirmation.

The subroutine in Algorithm 2 first checks if $P_{q,u}$ exists (line 1). If it does, preference transfer is not needed and $q$ is returned (line 2). In this case, $u$ has already edited the cluster tree for query $q$ and stored the preferences in $P_{q,u}$.

**Algorithm 2** *Trans(q, u)*

```
indent = 1em
Input: $q$, $u$ and $P_u$: $q$ is a query. $u$ is a user. $P_u$ is a collection of
    preferences by user $u$ for all queries, where $\forall q, P_{q,u} \in P_u$.
Output: $q'$: a query such that $P_{q',u}$ is applicable for $q$.
 1: if ($P_{q,u}$ exists) then
 2:     return $q$; //$u$ has edited the cluster tree of $q$
 3: else
 4:     find $q'$ s.t. $P_{q',u} \in P_u \wedge termSim(q, q')$ is the largest;
 5:     if $termSim(q, q') \geq \delta_{ts}$ then //$\delta_{ts}$ is a threshold
 6:         if $resultSim(q, q') \geq \delta_{rs}$ then //$\delta_{rs}$ is a threshold
 7:             $P_{q,u} \leftarrow P_{q',u}$; //copy preferences from $q'$ to $q$
 8:             return $q'$;
 9:         end if
10:     end if
11: end if
12: return $NULL$;
```

Otherwise, the subroutine tries to find $q'$ such that $P_{q',u}$ is applicable (lines 4 $\sim$ 11). To do so, it first finds $q'$ such that $P_{q',u}$ exists and $termSim(q, q')$ is the largest (line 4). Then, it continues to validate the applicability of $q'$ by checking if $termSim(q, q')$ and $resultSim(q, q')$ have passed their respective thresholds (lines 5 $sim$ 6). If so, user preferences for $q'$ will be copied to $q$ (line 7), and $q'$ will be returned (line 8). Otherwise, $NULL$ will be returned (line 11), indicating no applicable preferences exist for query $q$.

The preference copying (line 7) is important for the correctness of `ClusteringWiki`. Otherwise, suppose there is a preference transfer from $q'$ to $q$, where $P_{q,u} = \emptyset$ and $P_{q',u}$ has been applied on $T_{init}$ to produce $T$. Then, after some editing from $u$, $T$ becomes $T'$ and the corresponding edits are stored in $P_{q,u}$. Then, this $P_{q,u}$ will be used the next time the same query $q$ is issued by $u$. However, $P_{q,u}$ will not be able to bring an identical $T_{init}$ to the expected $T'$. It is easy to verify that line 7 fixes the problem and ensures reproducibility.

$Trans(q, U)$ works in the same way. Preference transfer is an important component of `ClusteringWiki`. Cluster editing takes user effort and there are an infinite number of queries. It is essential that such user effort can be properly reused.

**Enforcing applicable preferences.** Back to Algorithm 1, lines 15 $\sim$ 23 enforce the paths of $P$ on $T_{init}$ to produce the cluster tree $T$. The enforcement is straightforward. First $P$ is cleaned by removing those paths whose result nodes are not in the search result set $R$ (line 16). Recall that `ClusteringWiki` performs clustering. It should not alter the input data $R$. Then, the positive paths in $P$ are the ones $u$ prefers to see in $T$, thus they are added to $T$ (lines 18 $\sim$ 19). The negative paths in $P$ are the ones $u$ prefers not to see in $T$, thus they are removed from $T$ (lines 20 $\sim$ 21). If $P = \emptyset$, there are no applicable preferences and $T_{init}$ will not be modified.

**Trimming and Presenting $T$.** The cluster tree $T$ must satisfy a set $C$ of predefined constraints. Some constraints maybe violated after applying $P$ to $T_{init}$. For example, adding or removing paths may result in small clusters that violate constraints on the size of clusters. In line 24, subroutine $trim(T, C)$ is responsible for making $T$ consistent, e.g., by re-distributing the paths in the small clusters. We will discuss the constraint set $C$ in detail in Section 4.3.2.

In line 25, subroutine $present(T)$ presents the set of paths in $T$ as a cluster tree on the search interface. The labels can be expanded or collapsed. The search results for a chosen label are presented in the result panel in their original order when retrieved from the source. Relevant terms corresponding to current and ancestor labels in search results are highlighted.

Sibling cluster labels in the label panel are ordered by lexicographically comparing the lists of original ranks of their associated search results. For example, let $A$ and $D$ be two sibling labels as in Figure 5, where $A$ contains $P_1$, $P_2$, $P_3$ and $P_4$ and $D$ contains $P_1$ and $P_5$. Suppose that $i$ in $P_i$ indicates the original rank of $P_i$ from the source. By comparing two lists $< 1, 2, 3, 4 >$ and $< 1, 5 >$, we put $A$ in front of $D$. “Other" is a special label that is always listed at the end behind all its siblings.

**Discussion.** As [54] suggested, the subset of web pages visited by employees in an Enterprise is centered around the company’s business objectives. Additionally, employees share a common vocabulary describing the objects and tasks encountered in day to day activities. `ClusteringWiki` can be even more effective in this environment as user preferences can be better aggregated and utilized.

### *4.3.2 Cluster Editing*

Before explaining the algorithm handling user edits, we first introduce the essential consistency constraints for cluster trees and the primitive user edits.

**Essential consistency constraints.** Predefined consistency constraints exist to maintain a favorable user interface for fast and intuitive navigation. They can be specified on any structural component of the cluster tree $T$. In the following, we list the essential ones.

- *Path constraint*: Each path of cluster tree $T$ must start with the root labeled *All* and end with a leaf node that is a search result. In case there are no search results returned, $T$ is empty without paths.
- *Presence constraint*: Each initial search result must be present in $T$. It implies that deletion of paths should not result in absence of any search result in $T$.
- *Homogeneity constraint*: A label node in $T$ must not have heterogeneous children that combine cluster labels with search results. This constraint is also used in other clustering engines such as Clusty and Carrot2.
- *Height constraint*: The height of $T$ must be equal or less than a threshold, e.g., 4.
- *Label length constraint*: The length of each label in $T$ must be equal or less than a threshold.
- *Branching constraint*: We call a label node a *bottom label node* if it directly connects to search results. Each non-bottom label node must have at least $T_n$ children. Each non-special bottom label node must have at least $T_m$ children. *Other* is a special bottom label node that may have less than $T_m$ children. *All*, when being a bottom label, could also have less than $T_m$ children in case there are insufficient search results. By default both $T_n$ and $T_m$ are set to 2 in `ClusteringWiki` as in Clusty.

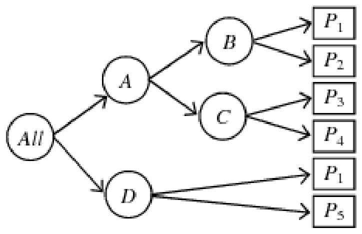


Figure 5: Example cluster tree.

**Primitive user edits.** `ClusteringWiki` implements the following categories of atomic primitive edits that a logged-in user can initiate in the process of tree editing. Each edit $e$ is associated with $P_e$ and $NP_e$, the set of paths to be inserted to the tree and the set of paths to be deleted from the tree after $e$.

- $e_1$: copy a label node to another non-bottom label node as its child. Note that it is allowed to copy a parent label node to a child label node.

  Example: in Figure 5, we can copy $D$ to $A$. For this edit, $P_e = \{All \rightarrow A \rightarrow D \rightarrow P_1, All \rightarrow A \rightarrow D \rightarrow P_5\}$. $NP_e = \emptyset$ for any edit of this type.
- $e_2$: copy a result node to a bottom label node.

  Example: in Figure 5, we can copy $P_3$ to $D$, but not to $A$, which is not a bottom label node. For this edit, $P_e = \{All \rightarrow D \rightarrow P_3\}$. $NP_e = \emptyset$ for any edit of this type.
- $e_3$: modify a non-root label node.

  Example: in Figure 5, we can modify $D$ to $E$. For this edit, $P_e = \{All \rightarrow E \rightarrow P_1, All \rightarrow E \rightarrow P_5\}$ and $NP_e = \{All \rightarrow D \rightarrow P_1, All \rightarrow D \rightarrow P_5\}$.
- $e_4$: delete a non-root node, which can be either a label node or a result node.

  Example: in Figure 5, we can delete $P_5$. For this edit, $NP_e = \{All \rightarrow D \rightarrow P_5\}$. $P_e = \emptyset$ for any edit of this type.
- $e_5$: create a label node, which can be either a non-bottom or bottom label node. In particular, recursive creation of non-bottom labels is a way to add levels to cluster trees.

  Example: in Figure 5, we can add $E$ as parent of $D$. For this edit, $P_e = \{All \rightarrow E \rightarrow D \rightarrow P_1, All \rightarrow E \rightarrow D \rightarrow P_5\}$ and $NP_e = \{All \rightarrow D \rightarrow P_1, All \rightarrow D \rightarrow P_5\}$.

The editing framework results in several *favorable properties*. Firstly, the primitive user edits are such that, with a series of edits, a user can produce *any* consistent cluster tree. Secondly, since $e_1$ only allows a label node to be placed under a non-bottom node and $e_2$ only allows a result node to be placed under a bottom node, the homogeneity constraint will not be violated after any edit given the consistency of $T$ before the edit. Thirdly, the framework uses *eager* validation, where validation is performed right after each edit, compared to *lazy* validation, where validation is performed in the end of the editing process. Eager validation is more user-friendly and less error-prone in implementation.

Note that, user editing can possibly generate *empty labels*, i.e., labels that do not contain any search results and thus not on any path. Such labels will be trimmed.

**Algorithm 3** *Cluster editing*

```
indent = 1em
Input: $q$, $u$, $T$, $C$, $P_{q,u}$, $P_{q,U}$ and $e$: $q$ is a query. $u$ is a user. $T$ is
    a cluster tree for $q$. $C$ is a set of consistency constraints for $T$.
    $P_{q,u}$ is a set of paths representing the preferences by $u$ for $q$.
    $P_{q,U}$ is a set of paths representing the aggregated preferences
    for $q$. $e$ is an edit by $u$ on $T$.
Output: updated $T$, $P_{q,u}$ and $P_{q,U}$
 1: if (pre-validation fail) then
 2:     return;
 3: end if
 4: identify $P_e$;
 5: identify $NP_e$;
 6: if (validation fail) then
 7:     return;
 8: end if
 9: update $T$;
10: add $P_e$ as positive paths to $P_{q,u}$;
11: add $NP_e$ as negative paths to $P_{q,u}$;
12: update $P_{q,U}$;
```

To add convenience, `ClusteringWiki` also implements several other types of edits. For example, move (instead of copy as in $e_1$) a label node to another non-bottom label node as its child, or move (instead of copy as in $e_2$) a result node to a bottom label node. Such a move edit can be considered as a copy edit followed by a delete edit.

**Editing algorithm.** Algorithm 3 presents the pseudocode of the cluster editing algorithm in `ClusteringWiki` for a single edit $e$, where $e$ can be any type of edit from $e_1$ to $e_4$.

Lines $1 \sim 3$ perform pre-validation of $e$ to see if it is in violation of consistency constraints. Violations can be caught early for certain constraints on certain edits, for example, the label length constraint on $e_1$ type of edits. If pre-validation fails, the algorithm returns immediately.

Otherwise, the algorithm continues with lines $4 \sim 5$ that identify $P_e$ and $NP_e$. Then, lines $6 \sim 8$ perform full validation of $e$ against $C$, the set of consistency constraints. If the validation fails, the algorithm returns immediately.

Otherwise, $e$ is a valid edit and $T$ is updated (line 9). Then, the personal user preferences are stored by adding $P_e$ and $NP_e$ to $P_{q,u}$ as positive paths and negative paths respectively (lines $10 \sim 11$). In adding these paths, the opposite paths in $P_{q,u}$ cancel each other out. In line 12, the aggregated preferences stored in $P_{q,U}$ are updated. We further discuss preference aggregation in the following.

**Preference sharing.** Preference sharing in `ClusteringWiki` is in line with the many social-powered search engines as a mass-collaborative way of improving search utility. In `ClusteringWiki`, $U$ is considered as a special user and $P_{q,U}$ stores the aggregated user preferences.

In particular, we use $P^0_{q,U}$ to record the paths specified for query $q$ by all users. Each path $p \in P^0_{q,U}$ has a *count* attribute, recording the total number of times that $p$ appears in any $P_{q,u}$. All paths in $P^0_{q,U}$ are grouped by leaf nodes. In other words, all paths that end with the same search result are in the same group. For each group, we keep track of two *best* paths: a positive one with the most count and a negative one with the most count. We mark a best path if its count passes a predefined threshold. All the marked paths constitute $P_{q,U}$, the set of aggregated paths that are used in query processing. Note that, here `ClusteringWiki` adopts a conservative approach, making use of at most one positive path and one negative path for each search result.

**Editing interface.** Cluster editing in `ClusteringWiki` is primarily available through context menus attached to label and result nodes. Context menus are context aware, displaying only those operations that are valid for the selected node. For example, the *paste result* operation will not be displayed unless the selected node is a bottom label node and a result node was previously copied or cut. This effectively implements pre-validation of cluster edit operations by not allowing the user to choose invalid tasks.

Users can drag and drop a result node or cluster label in addition to cutting/copying and pasting to perform a move/copy operation. A label node will be tagged with an icon if the item being dragged can be pasted within that node. An item that is dropped outside a label node in which it could be pasted simply returns to its original location.

## 4.4 Evaluation

`ClusteringWiki` was implemented as an AJAX-enabled Java Enterprise Edition 1.5 application. The prototype is maintained on an average PC with Intel Pentium 4 3.4 GHz CPU and 4Gb RAM running Apache Tomcat 6.

### *4.4.1 Methodology and Metrics*

We performed two series of experiments: system evaluation and utility evaluation. The former focused on the correctness and efficiency of our implemented prototype. The latter, our main experiments, focused on the effectiveness of `ClusteringWiki` in improving search performance.

**Data sources.**
Multiple data sources were used in our empirical evaluation, including Google AJAX Search API [27], Yahoo! Search API [28], and local Lucene indexes built on top of the New York Times Annotated Corpus [102] and several datasets from the TIPSTER (disks 1-3) and TREC (disks 4-5) collections [29]. The Google API can retrieve a maximum of 8 results per request and a total of 64 results per query. The Yahoo! API can retrieve a maximum of 100 results per request and a total of 1000 results per query. Due to user licence agreements, the New York Times, TIPSTER and TREC datasets are not available publicly.

**System evaluation methodology.**
For system evaluation of `ClusteringWiki`, we focused on *correctness* and *efficiency*. We tested the correctness by manually executing a number of functional and system tests designed to test every aspect of application functionality. These tests included cluster reproducibility, edit operation pre-validations, cluster editing operations, convenience features, applying preferences, preference transfer, preference aggregation, etc.

[27] code.google.com/apis/ajaxsearc
[28] developer.yahoo.com/search/web/webSearch.html
[29] www.nist.gov/tac/data/data_desc.html

In order to have repeatable search results for same queries, we used the stable New York Times data source. We chose queries that returned at least 200 results.

We evaluated system efficiency by monitoring query processing time in various settings. In particular, we considered:

- 2 data sources: Yahoo! and New York Times
- 5 different numbers of retrieved search results: 100, 200, 300, 400, 500
- 2 types of clusterings: flat (F) and hierarchical (H)

For each of the combinations, we executed 5 queries, each twice. The queries were chosen such that at least 500 search results would be returned. For each query, we monitored 6 portions of execution that constitute the total query response time:

- Retrieving search results
- Preprocessing retrieved search results
- Initial clustering by a built-in algorithm
- Applying preferences to the initial cluster tree
- Presenting the final cluster tree
- Other (e.g., data transfer time between server and browser)

For the New York Times data source, the index was loaded into memory to simulate the server side search engine behavior. The time spent on applying preferences depends on the number of applicable stored paths. For each query, we made sure that at least half the number of retrieved results existed in a modified path, which is a practical upper-bound on the number of user edits on the clusters of a query.

**Utility evaluation methodology.**
For utility evaluation, we focused on the *effectiveness* of `ClusteringWiki` in improving search performance, in particular, the time users spent to locate a certain number of relevant results. The experiments were conducted through a user study with 22 *paid* participants. The participants were primarily undergraduate, with a few graduate, colleage students.

We compared 4 different search result presentations:

- Ranked list (RL): search results were not clustered and presented as a traditional ranked list.
- Initial clustering (IC): search results were clustered by a default built-in algorithm (frequent phrase hierarchical).
- Personalized clustering (PC): search result clustering was personalized by a logged-in user after a series of edits, taking on average 1 and no more than 2 minutes per query.
- Aggregated clustering (AC): search result clustering was based on aggregated edits from on average 10 users.

Navigational queries seek the website or home page of a single entity that the user has in mind. The more common [17, 98] informational queries seek general information on a broad topic. The ranked list interface works fine for the former in general but is less effective for the latter, which is where clustering can be helpful [74]. In practice, a user may explore a varied number (e.g., 5 or 10) of relevant results for an informational query. Thus, we considered 2 types of informational queries. In addition, we argue that for some *deep* navigational queries where the desired page “hides" deep in a ranked list, clustering can still be helpful by skipping irrelevant results. Thus, we also considered such queries:

- $R_{10}$: Informational. To locate any 10 relevant results.
- $R_5$: Informational. To locate any 5 relevant results.
- $R_1$: Navigational. To locate 1 pre-specified result.

For each query type, 10 queries were executed, 5 on Google results and 5 on the AP Newswire dataset from disk 1 of the TIPSTER corpus. The AP Newswire queries were chosen from TREC topics 50-150, ensuring that they returned at least 15 relevant results within the first 50 results. For $R_1$ queries, the topic descriptions were modified to direct the user to a single result that is relatively low-ranked to make the queries “deep". Google queries were chosen from topics that participants were familiar with. All queries returned at least 50 results. These queries and their descriptions and narratives can be found at [3].

Each user was given 15 queries, 5 for each query type. Each query was executed 4 times for the 4 presentations being compared. Thus, in total each user executed $15 \times 4 = 60$ queries. For each execution, the user exploration effort was computed.

*User effort* was the metric we used to measure the search result exploration effort exerted by a user in fulfilling her information need. [58] used a similar metric under a probabilistic model instead of user study. Assuming both search results and cluster labels are scanned and examined in a top-down manner, user effort $\Omega$ can be computed as follows:

- Add 1 point to $\Omega$ for each examined search result.
- Add 0.25 point to $\Omega$ for each examined cluster label. This is because labels are much shorter than snippets.
- Add 0.25 point to $\Omega$ for each *uncertain result*. Based on our assumption, all results before a tagged relevant result are examined. However, results after the last tagged result remain uncertain. For linked list presentation, there is no uncertainty because the exploration ends at a tagged result due to the way the queries are chosen (more relevant results than needed).

  Uncertainty could occur for results within a chosen cluster $C$. As an effective way of utilizing cluster labels, most users would partially examine a few results in $C$ to evaluate the relevance of $C$ itself. If they think $C$ is relevant, they must have found and tagged some relevant results in $C$. If they think $C$ is irrelevant, they would ignore the cluster and quickly move to the next label. Thus, each uncertain result has a probability of being examined. Based on our observation for this particular user study, we empirically used 0.25 for this probability.

### 4.4.2 System Evaluation Results

For correctness, all functional and system tests were executed successfully. A detailed description of these tests can be found at [3]. In the following, we focus on the efficiency evaluation results.

We recorded and averaged (over 10 queries) the runtime in seconds for all 6 portions of total response time. In addition, we also computed the average *total execution time*, which includes preprocessing, initial clustering, applying preferences and presenting the

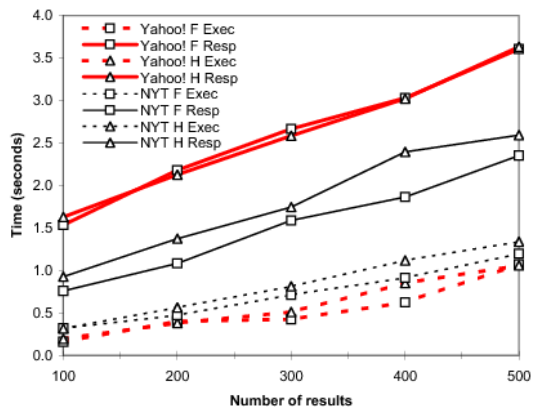


Figure 6: Efficiency evaluation.

final tree. This is the time that our prototype is responsible for. The remaining time is irrelevant to the way our prototype is designed and implemented. While the details are reported in [3], Figure 6 shows the trends of the average total execution time (Exec in the figure) and response time (Resp) for both flat (F) and hierarchical (H) presentations over 2 sources of Yahoo! (Yahoo!) and New York Times (NYT). From the figure we can see that:

- Response and execution time trends are linear, testifying to the scalability of our prototype. In particular, for both flat and hierarchical clustering, the total execution time is about 1 second for 500 results and 0.4 second for 200 results from either source. Note that most existing clustering search engines, e.g., iBoogie [30] and CarrotSearch [31], cluster 100 results by default and 200 at maximum. Clusty [32] clusters 200 results by default and 500 at maximum.
- Hierarchical presentation (H) takes comparable times to flat presentation (F), showing that recursive generation of hierarchies does not add significant cost to efficiency.
- There is a bigger discrepancy between response and execution times for the Yahoo! data source compared to New York Times, suggesting a significant efficiency improvement by integrating our prototype with the data sources.
- Execution times for Yahoo! are shorter than New York Times due to the shorter titles and snippets.

In addition, we observe (and report in [3] with supporting data) that applying preferences takes less than $1/10$ second in all test cases, which certifies the efficiency of our "path approach" for managing preferences. Moreover, presenting the final tree takes the majority (roughly 80%) of the total execution time, which can be improved by using alternate user interface technologies.

### 4.4.3 Utility Evaluation Results

Figure 7 shows the averaged user effort (over $22 \times 5 = 110$ queries) for each of the 4 presentations (RL, IL, PC, AC) and each of the query types ($R_1$, $R_5$, $R_{10}$) on the Google data source. Similar trends can be observed from the AP Newswire data source (see [3] for details). From the figure we can see that:

[30]www.iboogie.com
[31]carrotsearch.com
[32]www.clusty.com

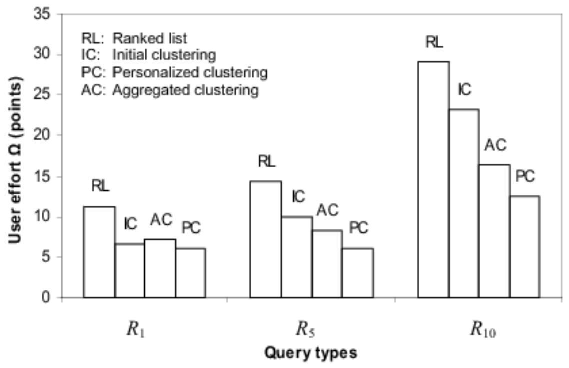


Figure 7: Utility evaluation on Google data source.

- Clustering saves user effort in informational and deep navigational queries, with personalized clustering the most effective, saving up to 50% of user effort.
- Aggregated clustering also significantly benefits, although it is not as effective as personalized clustering. However, it is "free" in the sense that it does not take user editing effort, and it does not require user login.

  In evaluating aggregated clustering, we made sure that the users using the aggregated clusters were not the ones who edited them.
- The effectiveness of clustering is related to how "deep" the relevant results are. The lower they are ranked, the more effective clustering is because more irrelevant results can be skipped.

The hierarchy of cluster labels plays a central role in the effectiveness of clustering search engines. From the data we have collected as well as the user feedback, we observe that:

- Cluster labels should be short and in the range of 1 to 4 terms, with 2 and 3 the best. The total levels of the hierarchy should be limited to 3 or 4.
- There are two types of cluster edits, (1) assigning search results to labels and (2) editing the hierarchy of labels. Both types are effective for personalized clustering. However, they respond differently for aggregated clustering. For type 1 edits, there is a ground truth (in a loose sense) for each assignment that users tend to agree on. Such edits are easy to aggregate and be collaboratively utilized. For type 2 edits, it can be challenging (and a legitimate research topic) to aggregate hierarchies because many edited hierarchies can be good but in diverse ways. A good initial clustering (e.g., frequent phrase hierarchical) can alleviate the problem by reducing the diversity.

As part of the user study, we also surveyed on the effectiveness of general, personalized and aggregated clustering in helping with search result exploration. On a scale of 1 to 10 with 10 as the best, users responded with an average rating of 8.21. Most users found `ClusteringWiki` efficient and useful in reducing their search effort.

## 4.5 Conclusion

Search engine utility has been significantly hampered due to the ever-increasing information overload. Clustering has been considered a promising alternative to ranked lists in improving search result organization. Given the unique human factor in search result clustering, traditional automatic algorithms often fail to generate clusters and labels that are interesting and meaningful from the user's perspective. In this paper, we introduced `ClusteringWiki`, the first prototype and framework for personalized clustering, utilizing the power of direct user intervention and mass-collaboration. Through a Wiki interface, the user can edit the membership, structure and labels of clusters. Such edits can be aggregated and shared among users to improve search result organization and search engine utility.

There are many interesting directions for future work, from fundamental semantics and functionalities of the framework to convenience features, user interface and scalability. For example, in line with social browsing, social network can be utilized in preference aggregation.

## 5. WORD SENSE DISAMBIGUATION

Word sense disambiguation (WSD) is the process of using automated tools to distinguish different usages for the same term. This often, although not always, lines up with different dictionary senses of a wordl However, dictionary alignment is not necessary for many important uses of word sense disambiguation, such as locating an uncommon usage for a common term.

There are many different techniques to do WSD, but often the ultimate goal is to produce a fine-grained understanding of a the terms in a specific corpus. Given the exisiting resources, it often makes more sense to create algorithms and tools to customize existing resources, rather than trying to generate a new resource purely algorithmicly.

Here, we present the C-Cat Wordnet package, an open source library for using and modifying Wordnet. Later projects intend to leverage this work to provide a complete WSD package with integrated corpus-specific concept hierarchies. The package includes four key features: an API for modifying Synsets; implementations of standard similarity metrics, implementations of well-known Word Sense Disambiguation algorithms, and an implementation of the Castanet algorithm. The library is easily extendible and usable in many runtime environments. We demonstrate it's use on two standard Word Sense Disambiguation tasks and apply the Castanet algorithm to a corpus.

### 5.1 Introduction

Wordnet [33] is a hierarchical lexical database that provides a fine grained semantic interpretation of a word. Wordnet forms a diverse semantic network by first collecting similar words into synonym sets ($Synset$), for example “drink” and “imbibe” are connected under the verb $Synset$ defined as “take in liquids.” Then, $Synsets$ are connected by relational links, with the IS-A link being the most well known.

Applications typically access Wordnet through one or more libraries. Every popular programming language has at least one library: the original for C++, JWNL [33] for Java, and WordNet::QueryData [34] for Perl are just a few examples. While these libraries are robust and provide many features, they cannot be easily applied to two new use cases: direct modification and serialization of the database and use in a parallel processing framework, such the Hadoop [35] framework. The first has become a popular research topic in recent years, with [104] providing a well known method for adding new lexical mappings to Wordnet, and the second will increasingly become important as Wordnet applications are applied to massive web-scale datasets.

[33] http://sourceforge.net/projects/jwordnet/

[34] http://people.csail.mit.edu/jrennie/WordNet/

We developed the C-Cat Wordnet package to address these use cases as part of a larger information extraction and retrieval system that requires word sense information for new, domain specific terms and novel composite senses on web-scale corpora. One example includes adding new lexical mappings harvested from New York Times articles. Without support for saving additions to Wordnet and parallel processing, we would be unable to leverage existing valuable sense information. Our package solves these issues with a new API focused on modifying the database and by storing the entire Wordnet database in memory.

We designed the package to be a flexible library for any Wordnet application. It is written in Java and defines a standard Java interface for core data structures and algorithms. All code has been heavily documented with details on performance trade-offs and unit tested to ensure reliable behavior. While other Wordnet libraries exist, we hope that the release of ours facilitates the development of new, customized Wordnets and the use of Wordnet in large highly parallelized systems. The toolkit is available at http://github.com/fozziethebeat/C-Cat, which include a wiki detailing the structure of the package, javadocs, and a mailing list.

### 5.2 The C-Cat Wordnet Framework

Fundamentally, Wordnet acts as a mapping from word forms to possible word senses. Terms with similar senses are collapsed into a single $Synset$. The $Synset$ network is then formed by linking a $Synset$ to others via semantic links such as IS-A, PART-OF, and SIMILAR-TO. Our package makes two significant contributions: a collection a standardized reference implementations of well known algorithms and a new API for directly modifying and serializing the $Synset$ network. In addition, it provides features found in comparable libraries such as JWNL.

The C-Cat library is split up into four packages:

1. The Core Api contains data format readers, writers, and $Synsets$;
2. Similarity Metrics;
3. Word Sense Disambiguation algorithms;
4. and Castanet [107], a method for automatically learning document facets using Wordnet.

#### 5.2.1 Core Api

The core API is centered around two interfaces: an $OntologyReader$ and a $Synset$. The $OntologyReader$ is responsible for parsing a Wordnet file format, building a linked $Synset$ network, and returning $Synsets$ based on query terms and parts of speech. The $Synset$ maintains all of the information for a particular word sense, such as it's definitions, examples, and links to other $Synsets$. Both interfaces provide mechanisms for modifying the sense information, $Synset$ links, and lexical mappings. We store this entire structure in memory due to the minimal size of Wordnet, for example, version 3.0 is only 37 Megabytes on disk, and so that users

[35] http://hadoop.apache.org/

```
OntologyReader reader = ...
Synset cat = reader.getSynset("cat.n.1");
for (Synset rel : cat.allRelations())
  cat.merge(rel);
System.out.println(cat);
```

Figure 8: A simple merge example using the $OntologyReader$ and $Synset$ interfaces.

can use Wordnet on novel distributed file systems, such as Hadoop, that do not use standard file system APIs.

*$Synsets$.* are defined by three sets of values: word forms, links to other $Synsets$, and a part of speech. Each $Synset$ may have multiple word forms and multiple links, but only one part of speech. We use both standard Wordnet relations and arbitrary relations to label a directed link between two $Synsets$, with the relation being stored in only the source $Synset$. We provide several methods for accessing relations and related $Synsets$: $getKnownRelationTypes()$, $allRelations()$, and $getRelations()$. In addition, each $Synset$ can have a set of example sentences and a definition. To modify each $Synset$, the interface includes additive methods for relations, word forms, and examples. Furthermore, we provide a $merge()$ method that takes all information from one $Synset$ and adds it to another $Synset$. Figure 8 provides a simple example using this merge API; after the code has been run, "cat.n.1" will contain all of the information from it's related $Synsets$. Lastly, the interface also permits arbitrary objects, such as ranking values, feature vectors, or additional meta data, to be attached to any $Synset$ as an $Attribute$. Any $Attributes$ are also merged on a call to $merge$.

*$OntologyReader$.* defines an interface that maps word forms to $Synset$s. Implementations are designed to be initialized once and then used ubiquitously throughout an application. The interface provides methods for getting all $Synset$s for a word or a specific sense, for example, the query "cat.n.1" in figure 8 retrieves the first noun $Synset$ for the term "cat". To modify the sense network, we provide two key methods: $addSynset(new)$ and $removeSynset(old)$. $addSynset(new)$ adds a mapping from each of $new$'s word forms to $new$. $removeSynset(old)$ removes all mappings from $old$'s word forms to $old$, thus removing it from the lexical mapping completely.

### 5.2.2 Similarity Metrics

While the semantic network of Wordnet is interesting on it's own, many applications require sense similarity measures. As such, we provide the $SynsetSimilarity$ interface that returns a similarity score between two $Sysnet$s. This is, in short, a Java based implementation of the Wordnet::Similarity package [90], which is in Perl. Figure 9 provides a naive, but short, code sample of our API that computes the similarity between all noun $Synset$s using multiple metrics.

Below, we briefly summarize the measures from [90] that we implemented. Several measures utilize the Lowest Common Subsumer (LCS), i.e. the deepest parent common to two $Synset$s using IS-A relations. Each measure takes in two $Synset$s, $A$ and $B$, as arguments and returns a double value, typically between 0 and 1.

*Path Based Methods.* measure the similarity based on a path connecting $A$ and $B$. $Path$ simply returns the inverse length of the shortest path between $A$ and $B$. $Leacock\&Chodorow$ [63] returns the length of the shortest path scaled by the deepest depth in the hierarchy. $Wu\&Palmer$ [120] returns the depth of the LCS scaled by the cumulative depth of $A$ and $B$. $Hirst\&St.Onge$ [46] uses all links in the hierarchy and measures the length of the path that is both short and has very few link types.

*Lexical methods.* measure the amount of lexical overlap between $A$ and $B$. $Lesk$ [66] returns the number of words overlapping in $A$ and $B$'s glosses. $ExtendedLesk$ [7] extends Lesk by also comparing the glosses between any $Synset$s related to $A$ or $B$.

*Information based Methods.* utilize the Information Content (IC) of a $Synset$, which measures the specificity of the terms in a $Synset$ as measured in a sense tagged corpus. $Resnick$ [94] returns the IC of the LCS. $Jiang\&Conrath$ [51] returns the inverse difference between the total IC of $A$ and $B$ and the IC of their LCS. $Lin$ [69] returns the IC of the LCS scaled by the total IC of $A$ and $B$.

In addition to the raw similarity metrics, we provide a utility classes that return meta information about a pair of $Sysnets$ such as their shortest path, their LCS, and several other helpful methods.

### 5.2.3 Word Sense Disambiguation

Word Sense Disambiguation is perhaps the most standard application of Wordnet. Disambiguation models attempt to select a $Synset$ for a given word that best matches a given context. For example, an algorithm might select the river bank $Synset$ of "bank" for the context "he sat on the bank of the river" rather than the financial institution $Synset$. We provide a $WordSenseDisambiguation$ interface that applies word sense labels to tokenized sentences. Currently, we only provide a small number of unsupervised algorithms, but plan on adding more. Below, we briefly describe each algorithm.

*Lexical Methods.* rely on lexical information in Wordnet to disambiguate words. $Lesk$ [66] selects the $Synset$ that has the highest total Lesk similarity to the $Synset$s for other context words. $ExtendedLesk$ [7] extends Lesk by using the Extended Lesk similarity metric for all comparisons. $MostFrequentSense$ selects the first $Synset$ returned by Wordnet for a given term. This serves as a canonnical baseline which is often challenging to outperform.

*Graphical Methods.* treat the network as a graph and disambiguate using a number of measurements. $PersonalizedPageRank$ [1] ($PPR$) runs the PageRank algorithm over an undirected graph composed from the entire Wordnet network. Words needing disambiguation are given "artificial" nodes that link to their possible $Synset$s. For each ambiguous word, the algorithm selects the highest ranking $Synset$. $DegreeCentrality$ [83] ($DC$) forms a subgraph from the Wordnet network composed of ambiguous content words in a sentence and the $Synset$s that connect their possible $Synset$s. It assigns to each word the target $Synset$ with the highest degree in the subgraph. $PageRankCentrality$ [83] ($PRC$)

```
OntologyReader reader = WordNetCorpusReader.initialize(...);
Set<Sysnet> nouns = reader.allSynsets(PartsOfSpeech.NOUN);
SynsetSimilarity sims[] = {new PathSimilarity(), new LeskSimilarity(), ...};
for (Synset s1 : nouns)
  for (Synset s2 : nouns)
    for (SynsetSimilarity sim : sims)
      System.out.printf("%s %s %f\n", s1, s2, sim.similarity(s1, s2));
```

Figure 9: Code for computing the pairwise similarity over every noun $Synset$ using multiple metrics

composes the same subgraph as $DegreeCentrality$, but performs PageRank on this subgraph and selects the $Synset$ with the highest rank for each ambiguous word.

### *5.2.4 Castanet*

Amazon.com and other online retailers often display manually crafted facets, or categories, for product navigation. A customer can start browsing from the Book category and dive down into more specific categories such as Fiction, Entertainment, or Politics. These facets form a hierarchy of categories and each category is subdivided until a narrow set of interesting items are found. Unfortunately, not all datasets have well structued meta data. The Castanet algorithm automatically learns this hierarchical facted meta data (HFC) for a set of documents by using discriminative keywords [107], making structured navigation possible for abritrary document sets.

Castanet takes advantage of Wordnet's IS-A hierarchy to automatically create HFC. Castanet first extracts keywords from the set of documents (we use term-frequency inverse document frequency, TF-IDF, by default, but our API allows for other methods). For each extracted keyword, Castanet then creates a chain of words that lead from the root of the hierarchy to the keyword's $Synsets$. Each keyword chain is then merged together to form a "backbone" tree which is later reduced by eliminating redundant or non-discriminative nodes, such as those with one child.

Our Castanet API is both simple and flexible. To create a Castanet tree, one calls $Castanet.buildTree()$ with a directory path to a set of text documents. Our implementation will automatically extract keywords, extract the backbone tree, and finally index each document under it's learned facets. The returned result allows users to fully navigate the documents via the learned facets. We also provide an example Java web service for exploring the hierarchy in a browser.

## 5.3 Benchmark

To evaluate our library, we apply our six WSD implementations against two standard evaluations: the all words disambiguation tasks from SenseEval 3 [105] and SemEval 2007 [92], these use Wordnet version 1.7.1 and 2.1 respectively. We answer all test instances except those that do not have any mapping in Wordnet. Before processing, we apply part of speech tags to each token using the Open NLP MaxEnt Tagger ver 1.5.0[36]. We use the original databases as a baseline, called Base, in our experiments and test our modification API by adding the eXtended Wordnet (XWN) relations [78] to each database and disambiguate using these extended Wordnets[37].

Table 7 presents the F1 score for each algorithm using the original and extended databases. As expected, the MFS baseline outperforms each unsupervised algorithm. Although our scores do not match exactly with previous publications of these algorithms, we still see similar trends and the expected gains from adding new relations to the hierarchy. For $DegreeCentrality$ and $PageRankCentrality$, our different results are likely due to a implementation difference: when extracting a subgraph from Wordnet, we only use directed links as opposed to undirected links for computational efficiency. Other variations are possibly due to different methods of handling multi-word expressions and our part of speech tags. Still, $DC$ gains about 4% points with WXN relations and $PPR$ gains about 7% points on Senseval-3. Unexpectedly, $ExtendedLesk$ actually does worse with the additional relations.

| Model | Ver | SenseEval-3 | SemEval-07 |
|---|---|---|---|
| MFS | Base | 59.8 | 49.4 |
| Lesk | Base | 35.2 | 27.7 |
| E-Lesk | Base | 47.8 | **37.6** |
| PPR | Base | 42.9 | 32.8 |
| DC | Base | 43.2 | 33.3 |
| PRC | Base | 31.7 | 22.7 |
| Lesk | XWN | 35.2 | 27.7 |
| E-Lesk | XWN | 39.9 | 33.9 |
| PPR | XWN | **50.3** | 36.7 |
| DC | XWN | 47.3 | 37.1 |
| PRC | XWN | 33.0 | 24.0 |

Table 7: F1 Word Sense Disambiguation scores on the two test sets

We also performed a visual test of our Castanet implementation. We ran the algorithm over 1,021 articles extracted from the BBC World News using Wordnet 3.0. The articles came from a diverse set of categories including world, business, technology, and environmental news. Figures 10 and 11 show snapshots of our Castanet web application. Figure 10 displays the top level facets displayed to a new user. The top bar of this screen can break down the facets alphabetically to facilitate facet selection. Figure 11 shows a snapshot of several documents found after selecting several facets. It displays the selected facets, document titles, document text, and interesting key words. While this is only a simple interface, it provides an example of what our implementation can accomplish and how to use our API.

## 5.4 Future Work

We have presented our Java Wordnet library that provides two new key features: maintenance of an in memory database and an API centered around modifying the network directly. Additionally, we've provided implementations of several well known similarity metrics, disambiguation algorithms, and the Castanet information retrieval algorithm. All code is unit tested, heavily documented, and released under the GPL Version 2 license. We are currently working to extend our newest APIs, such as those for WSD and Castanet, to handle more interesting use cases. In the future work we hope

[36]http://opennlp.sourceforge.net/models-1.5/

[37]Note that we added XWN 1.7 relations to Wordnet 1.7.1 and XWN 2.0 relations to Wordnet 2.1, some links were discarded due to updates in Wordnet.

to expand this library with an evaluation framework for customized Wordnets, such as those generated by [104].

## 6. MIXED-CONTEXT ENTITY CO-OCCURRENCE MODELING (MC-ECO)

### 6.1 Introduction

The work described in this section aims to facilitate browsing and discovery of *mixed-type* entity contexts. For example, a pair of prominent individuals could be connected by both *business* and *politics*. We achieve this by applying latent topic models to contexts in which pairs of entities co-occur, giving the user a concise summary of entity co-occurrence. This representation could also be used to cluster relationships across different entity pairs.

### 6.2 Entity Co-Occurrence (ECO)

Given an appropriately annotated text corpus (e.g., the New York Times), the Entity Co-Occurrence (ECO) browser [42] allows the user to perform several useful tasks:

- given an entity (e.g., “George Bush”), find entities that frequently co-occur in the same context (e.g., Saddam Hussein)
- given a pair of entities (e.g., “Bush” and “Hussein”), examine the actual contexts in which they co-occur
- given a pair of entities (e.g., “Bush” “Hussein”), examine the *tf-idf* [75] representation of the aggregate of all contexts in which they co-occur

We refer to a single context in which a pair of entities co-occur as a *co-context*. An example choice of context might be the sentence, but different choices are possible (e.g., within a ten-token window).

### 6.3 Adding context types

A natural extension of this idea is to use these co-contexts to assign a *type* or *label* to each context. For example, “Bush” and “Hussein” could be said to have a *politics* context, while the context between “Larry Page” and “Sergey Brin” could be categorized as *business*. Even without provided ground truth labels, this could be achieved using *unsupervised* machine learning techniques [31]. For example we could simply apply $k$-means clustering to the *tf-idf* representations of the co-contexts of each entity pair. Clustering entity pairs by context in this way [44] would enhance user browsing capabilities; for example the user could choose to see only the entities which co-occur in *business* contexts with a given entity.

### 6.4 Mixed-context modeling

However this extension raises the issue of handling contexts that do not fall neatly into a single category. For example, the context between “Putin” and “Khodorkovsky” cannot neatly be categorized as either *business* or *politics*, but must be considered to be a mixture of both aspects.

### 6.5 Related work

Early work on this problem [44] clustered pairs of named entities by co-context cosine similarity. The resulting clusters were labeled and evaluated against manually labeled entity pairs.

The Semantic Network Extractor (SNE) [56] is based on Markov Logic Networks (MLN) [95] and extracts tuples of the form $(relation, arg1, arg2)$. The specific MLN is similar to co-clustering - arguments are clustered by their relation slots and relations are clustered by their arguments. Results are evaluated against manual gold standard relations.

The Mixed-Membership Stochastic Blockmodel (MMSB) [2] models graph data. Each node has a distribution $\theta$ over latent roles $z$. For each candidate edge $(i, j)$, each node samples latent roles $(z_i, z_j)$. The edge is generated with probability $\pi_{(z_i,z_j)}$, and is absent otherwise.

Nubbi [23] models two types of text: entity contexts and entity pair co-contexts. An entity context is formed by concatenating all contexts in which an entity is mentioned, and is modeled similar to an LDA document. An entity pair co-context is formed similarly from contexts in which the entities co-occur, and is modeled with a special “switching” variant of LDA that selects between *entity1* topics, *entity2* topics, and *entity pair* topics.

BlockLDA [6] combines LDA with ideas from MMSB, and also allows the generation of different data types associated with a given document such as words, entities, or tags.

Recent work on modeling relations *without* labeled text [96] leverages the use of *distant supervision*, where individual entity mentions are *not* labeled with the presence or absence of a given relation. Instead, distant supervision provides a KB of entity relations (from Freebase) and a text collection of entity mentions (from the New York Times) which *may or may not* discuss the relation. A constrained graphical model is then used to learn from this supervision. Follow-on work [122] improves performance further by *joint* inference of entity *types* and relation preferences with respect to those types. For example, $\texttt{isCitizenOf}(x, y)$ only makes sense where $x$ is a *person* and $y$ is a *country*.

Relational Topic Models (RTM) [22] builds on Supervised LDA (SLDA) [12], predicting the presence or absence of inter-document edges (e.g., citations). As in SLDA, a link between documents $i$ and $j$ depends on the empirical topic frequencies $\bar{z_i}$ and $\bar{z_j}$ via an element-wise product within a link prediction function $\psi(\eta^T(\bar{z_i} \odot \bar{z_j}))$.

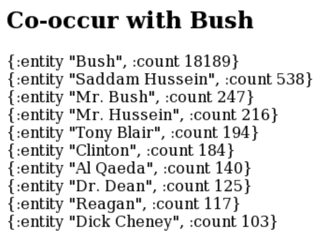
The concept of *link homophily* refers to the tendency for two edges with a common endpoint to be similar. This tendency can be exploited to characterize computer network traffic [35], even in the presence of packet obfuscation.

Using topic models over relations to assist in discovering new relations has been explored in [116]. That work leverages a large catalog of existing relations from Wikipedia to learn a background model, and leverages that to improve the extraction of new relations.

### 6.6 Goals

Table 8: Research project goals.

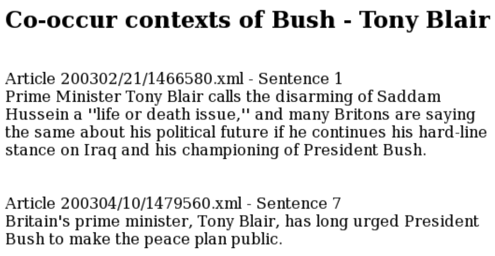
| Goal | Atom | Explanation |
|---|---|---|
| Edge label (mix) | (Entity,Entity) | Relation browsing |
| Node label (mix) | Entity | Entity browsing |



**Co-occur with Bush**

```
{:entity "Bush", :count 18189}
{:entity "Saddam Hussein", :count 538}
{:entity "Mr. Bush", :count 247}
{:entity "Mr. Hussein", :count 216}
{:entity "Tony Blair", :count 194}
{:entity "Clinton", :count 184}
{:entity "Al Qaeda", :count 140}
{:entity "Dr. Dean", :count 125}
{:entity "Reagan", :count 117}
{:entity "Dick Cheney", :count 103}
```

(a) Co-occurring entities.



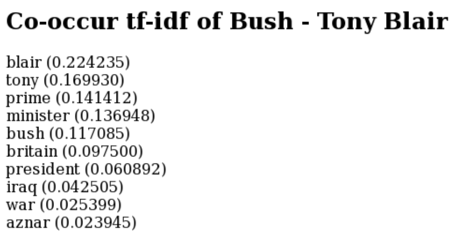
**Co-occur contexts of Bush - Tony Blair**

Article 200302/21/1466580.xml - Sentence 1
Prime Minister Tony Blair calls the disarming of Saddam Hussein a ''life or death issue,'' and many Britons are saying the same about his political future if he continues his hard-line stance on Iraq and his championing of President Bush.

Article 200304/10/1479560.xml - Sentence 7
Britain's prime minister, Tony Blair, has long urged President Bush to make the peace plan public.

(b) Co-occurrence contexts.

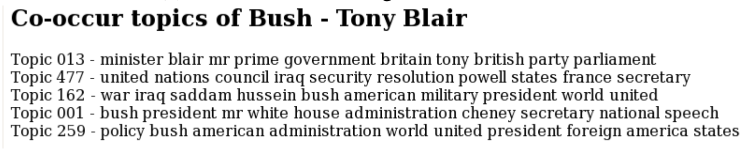
Figure 12: Investigating "Bush" entity co-occurrence.



**Co-occur tf-idf of Bush - Tony Blair**

```
blair (0.224235)
tony (0.169930)
prime (0.141412)
minister (0.136948)
bush (0.117085)
britain (0.097500)
president (0.060892)
iraq (0.042505)
war (0.025399)
aznar (0.023945)
```

(a) tf-idf of co-occurring contexts.



**Co-occur topics of Bush - Tony Blair**

```
Topic 013 - minister blair mr prime government britain tony british party parliament
Topic 477 - united nations council iraq security resolution powell states france secretary
Topic 162 - war iraq saddam hussein bush american military president world united
Topic 001 - bush president mr white house administration cheney secretary national speech
Topic 259 - policy bush american administration world united president foreign america states
```

(b) Prominent topics of co-occurring contexts.

Figure 13: tf-idf and LDA representations of Bush-Blair co-occurrence contexts.

Brief summaries of different desired system outputs are shown in Table 8.

## 6.7 Our approach

Our approach is to use LDA to model the co-occurrence contexts of entity pairs, treating individual sentences as the unit of context. That is, we first run LDA over the entire corpus. The final sample $\mathbf{z}$ then assigns each token in the corpus to a particular latent topic $z$. We can then combine these topic assignments with the entity pair co-occurrence sentences in order to identify topics associated with the entity pair. We have implemented a prototype of this system as a web application, which we use generate the results for the following example.

Say that we are a historian examining New York Times articles from 2003 and we are interested in the entity "Bush". We first query the web interface to get entities which often co-occur with Bush (Figure 12a). We would then see "Tony Blair" as a commonly co-occurring entity, and we could directly examine the sentences in which they co-occur (Figure 12b).

As in ECO, we can examine the tf-idf representation of their contexts, taken in aggregate over all sentences in which they co-occur (Figure 13a). By applying LDA topic modeling, we can also examine the prevalent topics from their co-occurrences (Figure 13b).

By grouping related words together, the latent topics give us a more informative summary than the tf-idf representation alone. Furthermore, the topics themselves are also associated with weights which can be used for entity-relationship clustering.

## 6.8 Future extensions

Because we are modeling the contexts in which a pair of entities co-occur, it is very important that these contexts are as complete and correct as possible. That is, we would like to resolve different references to the same entity (e.g., "Saddam Hussein" and "Hussein"). A further challenge is that we require these mentions to be identified as referring to the same entity even if the mentions occur in different documents. This problem is known as *cross-document coreference resolution*, and there exist highly scalable streaming algorithms for this purpose [93].

Beyond this practical concern, there are a variety of exciting theoretical directions in which to extend this approach.

- **Distant supervision:** labels at a coarser granularity than the level considered by our target task. For example, we may know that a relation exists between two entities, but not which mentions refer to that relation. This kind of knowledge may be effectively exploited with different types of **constraints**.
- **Partial supervision:** use an essentially unsupervised technique with some "seed" instances. For example some prototypical mentions could be hard-forced into a given cluster.
- **Non-parametric Bayes:** the number of underlying relations is probably not clear *a priori*, so explicitly modeling this uncertainty may be advantageous.
- **Hierarchical Bayes:** many entities appear only in a single relation, leading to a data sparsity challenge. This problem may benefit from the evidence sharing effects induced by hierarchical Bayesian models, e.g., pooling word contexts across many mentions via a latent parameter variable may reveal useful patterns. This formulation would also naturally encode a notion of **link homophily**.
- **Sparsity:** a parsimonious representation of a given co-context type or cluster has obvious advantages, both computational and cognitive. Tools from sparse dictionary learning work may be useful.

## 7. EXPLORING TOPIC COHERENCE OVER MANY MODELS AND MANY TOPICS

Topic models learn bags of related words from large corpora without any supervision. Based on the words used within a document, they mine topic level relations by assuming that a single document covers a small set of concise topics. Once learned, these topics should correlate well with human concepts, for example, one model might produce topics that cover ideas such as government affairs, sports, and movies. With these unsupervised methods, we can utilize useful semantic information in a variety of tasks that depend on identifying unique topics or concepts, such as distributional semantics [52], word sense induction [113, 18], and information retrieval [4].

When using a topic model, we are primarily concerned with the degree to which the learned topics match human judgements and help us differentiate between ideas. But until recently, the evaluation of these models has been ad hoc and application specific. In some cases a single approach has been compared to human judgements of semantic similarity or relatedness, see [52] for one set of evaluations. But these evaluations are costly to generate for domain specific topics. In other cases, automated intrinsic measures such as perplexity have been used, but it's been noted that improving the perplexity of a model may not correlate with learning more semantically coherent topics. Furthermore, few evaluations have used the same metrics to compare distinct approaches such as Latent Dirichlet Allocation [14], Latent Semantic Analysis [60], and Non-negative Matrix Factorization [64]. This has made it difficult to know which method is most useful and which parameters should be used.

We now provide a comprehensive evaluation of these three unique base models for automatically learning semantic topics. While these base models are not always used, they represent the core differences between each approach to modeling topics. For our evaluation, we use two recent automated metrics originally designed for LDA that try to bridge the gap between comparisons to human judgements and intrinsic measures such as perplexity [80, 86]. Using these metrics, we consider several key questions, such as

1. How many topics should be learned?
2. How many learned topics are useful?
3. How do these topics relate to often used semantic tests?
4. How well do these topics identify similar documents?

We first begin by summarizing the three topic models and highlight their key differences. We then describe the two metrics. Afterwards, we focus on a series of experiments that address our four key questions and finally conclude with some overall remarks.

## 7.1 Topic Models

We evaluate three latent factor models that have seen widespread usage:

1. Latent Dirichlet Allocation
2. Latent Semantic Analysis with Singular Value Decomposition
3. Latent Semantic Analysis with Non-negative Matrix Factorization

Each of these models have been designed with different goals and are supported by different statistical theories. And while the two forms of LSA have not typically been referred to as Topic Models, they have been used in a variety of similar contexts such as distributional similarity [52], word sense induction [113, 18], and information retrieval [4]. Based on these similar use cases, we consider it useful to compare these models with a consistent evaluation that matches well with our overall goal: latent factors should bring together similar words and separate unrelated words and latent factors should help distinguish between documents covering distinct topics.

To focus on our two goals, we are interested in two sets of relations that are learned by each model: how words interact with topics and how topics interact with documents. We generalize these two sets of relations as two distinct matrices: (1) $W$, a word by topic matrix that indicates the strength each word has in each topic, and (2) $H$, a document by topic matrix that indicates the strength each topic has in each document. In two of the models (LDA and NMF), these matrices can represent the relations as probabilities, while the SVD uses eigen vectors to represent these relations.

### 7.1.1 *Latent Dirichlet Allocation*

Latent Dirichlet Allocation [14] learns the relationships between words, topics, and documents by making an assumption on how documents are generated. It first assumes that there are a fixed set of topics that are used throughout a corpus and each topic can use all observed words. Then, each document, $D_i$ is generated by the following process

1. Choose $\Theta_i \sim Dir(\alpha)$, a topic distribution for $D_i$
2. For each word $w_j \in D_i$:
   (a) Select a topic $z_j \sim \Theta_i$
   (b) Choose $\Phi_{z_j} \sim Dir(\beta)$, a word distribution for a topic
   (c) Select the word $w_j \sim \Phi_{z_j}$

In this model, the $\Theta$ distributions represent the probability of each topic appearing in each document and the $\Phi$ distributions represent the probability of words being used for each topic. These two sets of distributions match exactly with our $H$ and $W$ matrices, respectively. The model uses one parameter, the number of topics, and two hyper parameters that guide the distributions, $\alpha$ and $\beta$. While the process above is a generative model, we use collapsed Gibbs sampling to infer these distributions [41].

### 7.1.2 *Latent Semantic Analysis*

Latent Semantic Analysis [60, 61] attempts to find descriptive latent factors that can compactly represent word distributions in observed in documents. The model first represents the data set as a large term by document matrix $M$ that simply records how many times each word occurs in each document. It then smooths the counts so that frequent, but uninformative, words, such as determiners and conjunctions, are given less weight while simultaneously boosting the weight of less frequent, but more informative words[38]. LSA then uses one of various dimensionality reduction techniques to learn a smaller sub-space that generalizes observed relations between word and documents. Traditionally, LSA has used the Singular Value Decomposition, but we also consider Non-negative Matrix Factorization as we've seen NMF applied in similar situations [89] and others have found a connection between NMF and Probabilistic Latent Semantic Analysis [29], an extension to LSA. We later refer to these two LSA models simply as SVD and NMF to signify the difference in factorization method.

*Singular Value Decomposition.* decomposes $M$ into three smaller matrices

$$M = U\Sigma V^T$$

[38]Based on the original LSA model, we use the Log-Entropy transform for this smoothing

such that $M$ can be reconstructed with minimal noise. Interestingly, the decomposition is agnostic to the number of desired dimensions. Instead, the rows and columns in $U$ and $V^T$ are ordered based on their descriptive power, i.e. how well they remove noise, which is encoded by $\Sigma$. As such, reduction is done by simply removing lower ranked rows and columns from $U$ and $V^T$. For our generalization, we use $W = U\Sigma$ and $H = \Sigma V^T$. We note that values in $U$ and $V^T$ can be both negative and positive, preventing them from being directly interpreted as probabilities.

*Non-negative Matrix Factorization.* factorizes $M$ with different constraints. It attempts to find two latent topic matrices that minimizes the euclidean least squares difference while only using non-negative values. In this respect, we can consider it to be learning probability distributions over topics. We use the original Euclidean least squares definition of NMF, but we note that the alternative KL-Divergence form of NMF has been directly linked to PLSA [29]. Formally, NMF is defined as

$$M = WH$$

Where $H$ and $W$ map directly onto our generalization. We learn these probabilities by initializing each set of probabilities at random and update them according to the following iterative update rules

$$W = W\frac{MH^T}{WHH^T}$$

$$H = H\frac{W^TM}{W^TWH}$$

## 7.2 Coherence Metrics

Topic Coherence metrics score a single topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference. For example consider the sets of topics in Tables 7.2 and 7.2. For our evaluations, we consider two new coherence metrics designed for LDA, both of which have been shown to match well with human judgements of topic quality:

1. The UCI metric [86]
2. The UMass metric [80]

Since both metrics compare distributional similarity between the top N words in a topic, we generalize the two metrics to compute the coherence of a topic $W$ as follows

$$coherence(W) = \sum_{(w_i,w_j)\in W} score(w_i, w_j, \epsilon)$$

Where $\epsilon$ indicates a smoothing factor which guarantees that $score$ returns real numbers.

*The UCI metric.* defines a word pair's score to be the point wise mutual information between two words, i.e.

$$score(w_i, w_j, \epsilon) = log\frac{p(w_i, w_j) + \epsilon}{p(w_i)p(w_j)}$$

The word probabilities are computed by counting word co-occurrence frequencies in a sliding window over an external corpus, such as Wikipedia. To some degree, this metric can be thought of as an external comparison to known semantic evaluations.

*The UMass metric.* defines the score to be based on document co-occurrence:

$$score(w_i, w_j, \epsilon) = log\frac{D(w_i, w_j) + \epsilon}{D(w_j)}$$

Where $D(x, y)$ counts the number of documents containing words $x$ and $y$ and $D(x)$ counts the number of documents containing $x$. Significantly, the UMass metric computes these counts over the *original corpus* used to train the topic models, rather than an external corpus. This metric is more intrinsic in nature; it attempts to confirm that the models learned data known to be in the corpus.

| Model | Metric | $\epsilon$ | Top Words |
|---|---|---|---|
| LDA | UMASS | 1.0 | told asked wanted thought time |
| | | $10^{-12}$ | told asked wanted thought time |
| | UCI | 1.0 | restaurant menu sauce food dishes |
| | | $10^{-12}$ | vaccine health smallpox flu vaccines |
| NMF | UMASS | 1.0 | browned servings tablespoons garlic oven |
| | | $10^{-12}$ | browned servings tablespoons garlic oven |
| | UCI | 1.0 | kitchen dining fireplace 1-bath 3-bedroom |
| | | $10^{-12}$ | loans borrowers refinance borrower refinancing |
| SVD | UMASS | 1.0 | sister struggling property welfare decade |
| | | $10^{-12}$ | sister struggling property welfare decade |
| | UCI | 1.0 | gains diagramed environmentalist organizing tricks |
| | | $10^{-12}$ | explain power rush noon shelter |

Table 9: The top 5 words for the best topics as measured by each metric

## 7.3 Evaluation

We have designed four sets of experiments to evaluate how each of these models perform, with a focus on ow the models vary based on the number of requested topics and how the models vary with respect to each other. We also take into consideration the $\epsilon$ value for the two metrics. We apply both metrics, and aggregate versions of each metric, to each learned model.

| Model | Metric | $\epsilon$ | Top Words |
|---|---|---|---|
| LDA | UMASS | 1.0 | front page 27 28 20 |
| | | $10^{-12}$ | front page 27 28 20 |
| | UCI | 1.0 | show students photographs 6 objects |
| | | $10^{-12}$ | hours day time night days |
| NMF | UMASS | 1.0 | lists 6 witchcraft murder 7 |
| | | $10^{-12}$ | 27 21 d5 f5 cd |
| | UCI | 1.0 | officials inside chemical game field |
| | | $10^{-12}$ | renovated 2-bath tax-deductible exposures doormen |
| SVD | UMASS | 1.0 | taxes wears innocent summoned approached |
| | | $10^{-12}$ | gains diagramed environmentalist organizing tricks |
| | UCI | 1.0 | pop contracts steering chaos grande |
| | | $10^{-12}$ | gains diagramed environmentalist organizing tricks |

Table 10: The top 5 words for the worst topics as measured by each metric

We trained all models on 92,600 New York Times articles from 2003 [101]. For all articles, we removed stop words and any words that occurred less than 200 times in the corpus, which left 35,836 tokens. All documents were tokenized based on whitespace. For the UCI metric, we used the UkWac Wikipedia corpus and considered all terms in Wikipedia while computing the sliding window with 10 words before and after the focus word. In all experiments, we used the top 10 words from each topic that had the highest weight, in terms of LDA and NMF this corresponds with a high probability of the term describing the topic but for SVD there is no clear semantic interpretation.

In our experiments we

1. explore several views of topic coherence such as average coherence, best coherence, and coherence entropy;
2. explore topic uniqueness metrics to evaluate how distinct each model becomes;
3. compare average topic coherence to previous semantic similarity evaluations;
4. compare topic coherence with classification strength.

### 7.3.1 Topic Coherence Views

Before we can compare topics models against one another, we require a coherence metric for complete models, rather than individual topics. We consider four possible aggregates: the coherence of the best topic for a model, the coherence of the median topic, the average coherence of all topics, and the entropy of the coherence for all topics. Each aggregate method gives us slightly different information, in particular the entropy can cleanly differentiate between two interesting cases that the others cannot: cases where all topics have relatively similar scores and cases where some topics are highly rated but others are poorly rated.

Figures 14, 15, and 16 show the scores for the average coherence, best coherence, and entropy, respectively, when setting $\epsilon = 1.0$ for the smoothing factor[39]. The average and best scores indicate a simple relationship between the three models: the SVD is constantly worse than LDA and NMF, and the NMF is often times better than LDA. Surprisingly though, with the UCI metric, the entropy shows an unexpected variation: the NMF has distinctly non-uniform scores. Based on a manual inspection of the topics, we noticed that some of the high scoring topics appeared incoherent and composed of rare words.

We further explored the impact of the smoothing factor and set $\epsilon = 10^{-12}$ and then re-evaluated the models. Figures 18 and 17 show the average and median coherence scores for the modified metrics. Interestingly, these plots show a starkly different relationship between NMF and LDA: under both metrics, the average NMF score is regularly less than the average LDA score and the median NMF score for both metrics begins to degrade as more topics are requested. Also of equal interest, we see that the SVD performs terribly, generating scores well below NMF and LDA. Even more interestingly, if we focus on the best 10% topics, we see the original relationship: NMF appears to generate better topics than the other two models.

We further explore this performance variation by focusing on a single set of models trained for 300 topics. Figures 22 and 23 show the spread of all topic scores for each model when setting $\epsilon$ to 1.0 and $10^{-12}$, respectively. In all cases, LDA has a reasonably narrow range. The SVD again has a narrow range when using a high $\epsilon$ and a wide range of poor scores for a low $\epsilon$. The NMF plots confirm our suspicion: the model continues to generate a set of high quality topics, but the majority of the topics learned have a wide range of low scores. We similarly see this variation in Figure 25, which plots the variation when taking the average over a subset of the best topics.

### 7.3.2 Topic Uniqueness

As our second experiment, wanted to evaluate whether or not the models are successfully learning new distinct topics as we request more. While the coherence metrics were initially designed to evaluate coherence, the first experiment indicated that they can similarly be used to evaluate incoherence. With that in mind, we developed a simple metric that rates the semantic disagreement between sets of topics. This is simply defined as

$$uniqueness(W) = \sum_{w_i \in W, w_j \notin W} score(w_i, w_j, \epsilon)$$

This measures the similarity between words in a topic and words in other topics. Ideally, each topic should have a low $uniqueness$ score, which indicates that each topic is semantically distinct. For complete models, we would expect the total $uniqueness$ over all topics to decrease as we request more topics until we reach a point of topic saturation, i.e. the model is no longer able to discover more unique topics.

Figure 26 reports the total $uniqueness$ scores for each model. Surprisingly, we see no saturation point for any model, indicating that

[39]To the best of our knowledge, the original metrics used this same smoothing factor

we could learn well over 500 topics and still extract new information. The SVD provides good justification that this metric makes sense, since each new topic returned by the SVD is guaranteed to be orthogonal and distinct from previous topics, the total $uniqueness$ decreases steadily. Both LDA and SVD similarly show a steady decrease with some variation that is likely due to random starting points.

#### *7.3.3 Word Similarity Tasks*

As our third experiment, we wished to compare the coherence scores to two standard word similarity tests, the [99] semantic similarity task and the [34] relatedness task. Both tasks were created by giving human judges a set of word pairs. Evaluators were asked to determine the similarity or relatedness of a word pair. The rubenstein65wordsim task used 65 word pairs while Finklestein et. al used 353 word pairs. Both have been used as a standard metric of distributional word spaces, jurgens10sspace provide a good collection of how traditional distributional semantic models fare on this task.

For each learned model, we use the $W$ word by topic matrix as a reduced representation of each word. For each test word pair, we use the cosine similarity between the reduced representations of each word and record the correlation between the similarity scores and the known human evaluations. A high correlation between similarity scores and human judgements indicate that the word by topic distributions closely model human expectations.

Figure 27 displays the results. Surprisingly, NMF and LDA both outperform SVD by a wide margin. Also, LDA does slightly better, especially as we request more topics, than NMF, which matches well with our observations from the first experiment. With the Rubenstein & Goodenough test, we see regular improvement in performance as we request more topics, while performance on the Finklestein et. al test levels out after about 100 topics, and even starts to degrade for NMF after 100 topics.

### 7.4 Discussion

Through our experiments, we made several interesting discoveries. First, we discovered that the coherence metrics depend heavily on the smoothing factor $\epsilon$. The original value used by the creators of the metric, 1.0 created a positive bias towards NMF models from both metrics, even when NMF generated incoherent topics. Our manual investigation suggests that the metrics do not accurately compare two words that are both rare and unrelated. A smaller $\epsilon$ correctly scores these cases and appears to have little affect on the common case, and so we recommend using a small smoothing factor. We also see that the SVD underperformed in all experiments, indicating that both LDA and NMF provide a better representation of semantics.

Second, we note that while fewer topics are faster to create, we have not exhausted the ability of automated methods to extract semantically coherent topics from a single year of the New York Times corpus, even with a relatively large number of topics (500) for a fairly small corpus. However, the quantity of low quality topics increases much more rapidly, and the computational cost increases dramatically as well.

Overall, it appears that LDA maintains a edge over NMF, and holds the promise of much more semantically motivated tuning. Over time we expect continual improvements in available LDA algorithms, enhancing both their speed and semantic utility.

## 8. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Funding was provided by LDRD #09-ER-017.

## 9. REFERENCES

[1] E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *JMLR*, 9:1981–2014, 2008.

[3] D. C. Anastasiu, D. Buttler, and B. J. Gao. Clusteringwiki technical report. *dmlab.cs.txstate.edu/ClusteringWiki/pdf/cw.pdf*, 2011.

[4] D. Andrzejewski and D. Buttler. Latent topic feedback for information retrieval. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 600–608, New York, NY, USA, 2011. ACM.

[5] D. Andrzejewski, X. Zhu, and M. Craven. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *International Conference on Machine Learning*, pages 25–32. Omnipress, 2009.

[6] R. Balasubramanyan and W. W. Cohen. Combining stochastic block models and topic models. In *SDM 2011: SIAM Conference on Data Mining*, 2011.

[7] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 805–810, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[8] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, 2004.

[9] R. Bekkerman, H. Raghavan, J. Allan, and K. Eguchi. Interactive clustering of text collections according to a user-specified criterion. In *IJCAI*, 2007.

[10] D. Blei, L. Carin, and D. Dunson. Probabilistic topic models. *Signal Processing Magazine, IEEE*, 27(6):55 –65, 2010.

[11] D. Blei and J. Lafferty. Visualizing topics with multi-word expressions. Technical report, arXiv, 2009. `arXiv:0907.1013v1 [stat.ML]`.

[12] D. Blei and J. McAuliffe. Supervised topic models. In *Advances in Neural Information Processing Systems*, pages 121–128. MIT Press, 2008.

[13] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[14] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[15] O. Bodenreider. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32(suppl 1):D267–D270, 2004.

[16] D. Borthakur, J. Gray, J. S. Sarma, K. Muthukkaruppan, N. Spiegelberg, H. Kuang, K. Ranganathan, D. Molkov, A. Menon, S. Rash, R. Schmidt, and A. Aiyer. Apache hadoop goes realtime at facebook. In *Proceedings of the*

*2011 international conference on Management of data*, SIGMOD '11, pages 1071–1080, New York, NY, USA, 2011. ACM.

[17] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[18] S. Brody and M. Lapata. Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 103–111, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[19] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *TREC*, pages 69–80. NIST, 1994.

[20] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):1–38, 2009.

[21] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. In *OSDI'06: Seventh Symposium on Operating System Design and Implementation*, Seattle, WA, November 2006.

[22] J. Chang and D. Blei. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, volume 5, pages 81–88. Journal of Machine Learning Research, 2009.

[23] J. Chang, J. Boyd-Graber, and D. M. Blei. Connections between the lines: augmenting social networks with text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 169–178, New York, NY, USA, 2009. ACM.

[24] C. Chemudugunta, A. Holloway, P. Smyth, and M. Steyvers. Modeling documents by combining semantic concepts with unsupervised statistical learning. In *International Semantic Web Conference*, pages 229–244. Springer, 2008.

[25] D. Cohn, A. K. McCallum, and T. Hertz. Semi-supervised clustering with user feedback. In K. L. Wagstaff, I. Davidson, and S. Basu, editors, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman and Hall/CRC, 2009.

[26] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley, 2010.

[27] W. Dakka, P. G. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *CIKM*, pages 768–775. ACM, 2005.

[28] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[29] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Comput. Stat. Data Anal.*, 52:3913–3927, April 2008.

[30] A. Doan, R. Ramakrishnan, and A. Halevy. Crowdsourcing systems on the World Wide Web. *Communications of the ACM*, 54(4), 2011.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[32] B. S. Everitt, S. Landau, and M. Leese. *Cluster analysis*. Oxford University Press, 2001.

[33] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, illustrated edition edition, May 1998.

[34] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20:116–131, January 2002.

[35] B. Gallagher, M. Iliofotou, T. Eliassi-Rad, and M. Faloutsos. Link homophily in the application layer and its usage in traffic classification. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –5, march 2010.

[36] B. J. Gao and J. Jan. Rants: a framework for rank editing and sharing in web search. In *WWW*, 2010.

[37] M. J. Gardner, J. Lutes, J. Lund, J. Hansen, D. Walker, E. Ringger, and K. Seppi. The topic browser: An interactive tool for browsing topic models. In *NIPS Workshop on Challenges of Data Visualization*. MIT Press, 2010.

[38] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *19th ACM Symposium on Operating Systems Principles*, Lake George, NY, October 2003.

[39] A. Griffiths, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Sciences*, 37(1):3–11, 1986.

[40] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[41] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[42] K. Halliday. ECO: A Framework for Entity Co-Occurrence Exploration with Faceted Navigation. Technical report, California State University, Chico, 2010.

[43] R. V. S. H. Halpin H. The complex dynamics of collaborative tagging. *WWW*, 2007.

[44] T. Hasegawa, S. Sekine, and R. Grishman. Discovering relations among named entities from large corpora. In *ACL*, pages 415–422, 2004.

[45] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR*, 1996.

[46] G. Hirst and D. St-Onge. Lexical chains as representation of context for the detection and correction malapropisms, 1997.

[47] M. Hoffman, D. Blei, and F. Bach. Online learning for latent Dirichlet allocation. In *NIPS*, pages 856–864. MIT Press, 2010.

[48] A. Iskold. Overview of clustering and clusty search engine. *www.readwriteweb.com/archives/overview_of_clu.php*, 2007.

[49] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

[50] X. Ji and W. Xu. Document clustering with prior knowledge. In *SIGIR*, 2006.

[51] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.

[52] D. Jurgens and K. Stevens. The s-space package: an open source package for word space models. In *Proceedings of the ACL 2010 System Demonstrations*, ACLDemos '10, pages 30–35, Stroudsburg, PA, USA, 2010. Association for

Computational Linguistics.

[53] M. Käki. Findex: search result categories help users when document ranking fails. In *CHI*, 2005.

[54] A. Kale, T. Burris, B. Shah, T. L. P. Venkatesan, L. Velusamy, M. Gupta, and M. Degerattu. icollaborate: harvesting value from enterprise web usage. In *SIGIR*, 2010.

[55] L. Kaufman and P. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 1990.

[56] S. Kok and P. Domingos. Extracting semantic networks from text via relational clustering. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 624–639, Berlin, Heidelberg, 2008. Springer-Verlag.

[57] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW*, pages 477–486, New York, NY, USA, 2008. ACM.

[58] J. Koren, Y. Zhang, and X. Liu. Personalized interactive faceted search. In *WWW*, 2008.

[59] K. Kummamuru, R. Lotlikar, S. Roy, K. Singal, and R. Krishnapuram. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *WWW*, 2004.

[60] T. K. Landauer and S. T. Dutnais. A solution to platoâĂŹs problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, pages 211–240, 1997.

[61] T. K. Landauer, P. W. Foltz, and D. Laham. An Introduction to Latent Semantic Analysis. *Discourse Processes*, (25):259–284, 1998.

[62] J. H. Lau, D. Newman, S. Karimi, and T. Baldwin. Best topic word selection for topic labelling. In *Coling 2010: Posters*, pages 605–613. Coling 2010 Organizing Committee, 2010.

[63] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press, 1998.

[64] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.

[65] J. Lee, S.-w. Hwang, Z. Nie, and J.-R. Wen. Query result clustering for object-level search. In *KDD*, 2009.

[66] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[67] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. In *International Conference on Machine Learning*, pages 577–584. Omnipress, 2006.

[68] S. Liberman and R. Lempel. Approximately optimal facet selection. In *(submission)*, 2011.

[69] D. Lin. An information-theoretic definition of similarity. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.

[70] X. Ling, Q. Mei, C. Zhai, and B. Schatz. Mining multi-faceted overviews of arbitrary topics in a text collection. In *KDD*, pages 497–505. ACM, 2008.

[71] T.-Y. Liu. Learning to rank for information retrieval. In *SIGIR Tutorials*, page 904, 2010.

[72] Y. Lu, Q. Mei, and C. Zhai. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, pages 1–26, 2010.

[73] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *5th Berkeley Symposium on mathematics, Statistics and Probability*, 1967.

[74] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

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

[76] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[77] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40:735–750, 2004.

[78] R. Mihalcea and D. I. Moldovan. extended wordnet: progress report. In *in Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, pages 95–100, 2001.

[79] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[80] D. Mimno, H. Wallach, E. Talley, M. Leenders, and A. McCallum. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Emperical Methods in Natural Language Processing*, pages 262–272, Edinburgh, Scotland, UK, 2011. Association of Computational Linguistics.

[81] D. M. Mimno and A. McCallum. Organizing the OCA: learning faceted subjects from a library of digital books. In *JCDL*, pages 376–385. ACM, 2007.

[82] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Uncertainty in Artificial Intelligence*, pages 352–359. Morgan Kaufmann, 2002.

[83] R. Navigli and M. Lapata. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:678–692, April 2010.

[84] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *HLT-NAACL*, pages 100–108, Morristown, NJ, USA, 2010. ACL.

[85] D. Newman, Y. Noh, E. Talley, S. Karimi, and T. Baldwin. Evaluating topic models for digital libraries. In *JCDL*, pages 215–224. ACM, 2010.

[86] D. Newman, Y. Noh, E. Talley, S. Karimi, and T. Baldwin. Evaluating topic models for digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, JCDL '10, pages 215–224, New York, NY, USA, 2010. ACM.

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

[88] L. A. Park and K. Ramamohanarao. The sensitivity of latent Dirichlet allocation for information retrieval. In *ECML PKDD*, pages 176–188. Springer-Verlag, 2009.

[89] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons. *Text mining using nonnegative matrix factorizations*, volume 54, pages 452–456. SIAM, 2004.

[90] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*,

HLT-NAACL–Demonstrations '04, pages 38–41, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[91] P. Pirolli, P. Schank, M. Hearst, and C. Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *CHI*, 1996.

[92] S. S. Pradhan, E. Loper, D. Dligach, and M. Palmer. Semeval-2007 task-17: English lexical sample, srl and all words. In *In Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007*, pages 87–92, 2007.

[93] D. Rao, P. McNamee, and M. Dredze. Streaming cross document entity coreference resolution. In *COLING (Posters)*, pages 1050–1058, 2010.

[94] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[95] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[96] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 148–163, Berlin, Heidelberg, 2010. Springer-Verlag.

[97] C. J. V. Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 1979.

[98] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW*, 2004.

[99] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8:627–633, October 1965.

[100] G. Salton. *The SMART Retrieval System*. Prentice-Hall, 1971.

[101] E. Sandhaus. The new york times annotated corpus, 2008.

[102] E. Sandhaus. The New York Times Annotated Corpus. *Linguistic Data Consortium, Philadelphia*, 2008.

[103] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proc. VLDB Endow.*, 3:703–710, 2010.

[104] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, ACL-44, pages 801–808, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[105] B. Snyder and M. Palmer. The english all-words task. In R. Mihalcea and P. Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[106] E. Stoica, M. Hearst, and M. Richardson. Automating creation of hierarchical faceted metadata structures. In *HLT-NAACL*, pages 244–251, Rochester, New York, April 2007. ACL.

[107] E. Stoica and M. A. Hearst. Automating creation of hierarchical faceted metadata structures. In *In Procs. of the Human Language Technology Conference (NAACL HLT*, 2007.

[108] V. Stoyanov, C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D. Hysom. Coreference resolution with reconcile. In *Proceedings of the Conference of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, 2010.

[109] Y. W. Teh, D. Newman, and M. Welling. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 1353–1360. MIT Press, 2006.

[110] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[111] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu. Data warehousing and analytics infrastructure at facebook. In *Proceedings of the 2010 international conference on Management of data*, SIGMOD '10, pages 1013–1020, New York, NY, USA, 2010. ACM.

[112] A. Tombros, R. Villa, and C. J. Van Rijsbergen. The effectiveness of query-specific hierarchic clustering in information retrieval. *Inf. Process. Manage.*, 38(4):559–582, 2002.

[113] T. Van de Cruys and M. Apidianaki. Latent semantic word sense induction and disambiguation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1476–1485, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[114] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[115] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, 2001.

[116] C. Wang, J. Fan, A. Kalyanpur, and D. Gondek. Relation extraction with relation topics. In *EMNLP*, 2011.

[117] X. Wang and C. Zhai. Learn from web search logs to organize search results. In *SIGIR*, 2007.

[118] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR*, pages 178–185. ACM, 2006.

[119] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Inf. Process. Manage.*, 43:685–704, 2007.

[120] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[121] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. *SIGIR*, pages 155–162, 2008.

[122] L. Yao, S. Riedel, and A. McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1013–1023, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[123] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI*, pages 401–408. ACM, 2003.

[124] X. Yi and J. Allan. A comparative study of utilizing topic models for information retrieval. In *ECIR*, pages 29–41. Springer-Verlag, 2009.

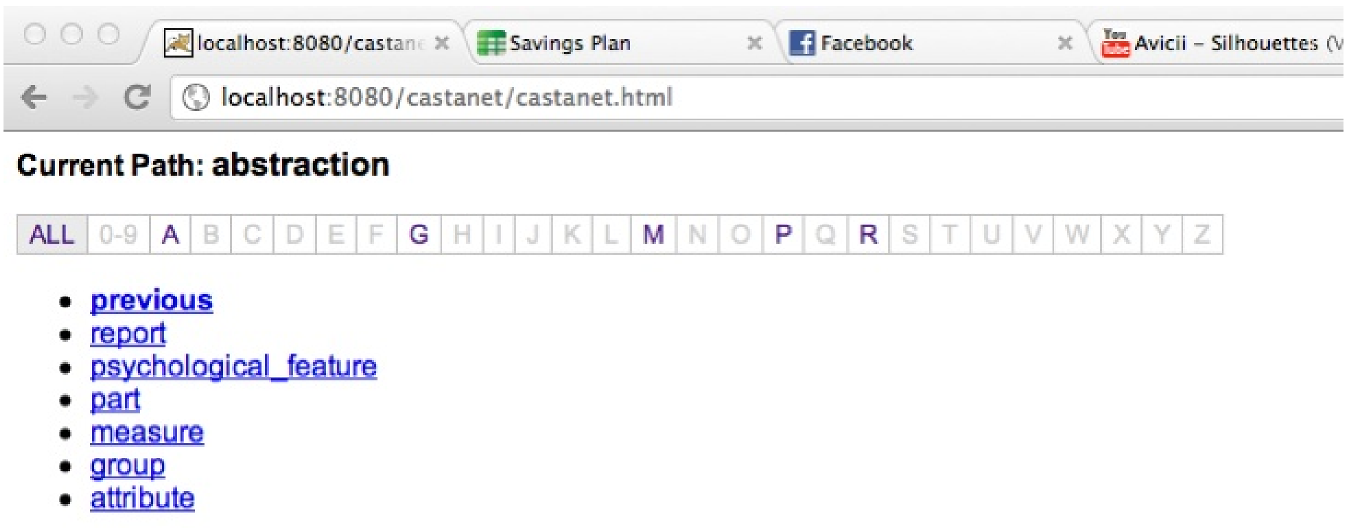
[125] O. Zamir and O. Etzioni. Web document clustering: a

feasibility demonstration. In *SIGIR*, 1998.

[126] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to web search results. In *WWW*, 1999.

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

[128] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *SIGIR*, pages 363–370. ACM, 2010.

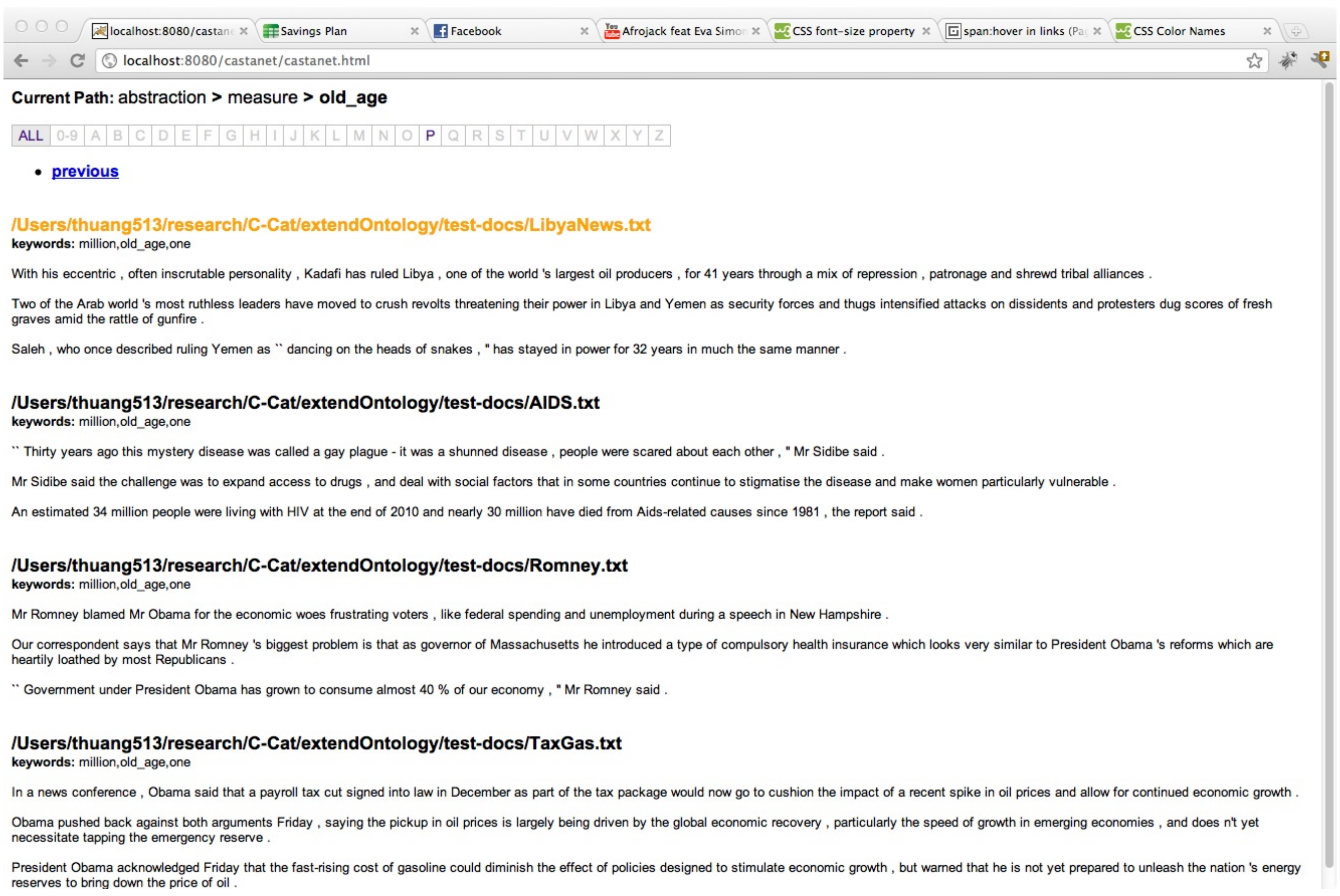


Current Path: abstraction

ALL 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- previous
- report
- psychological_feature
- part
- measure
- group
- attribute

Figure 10: A sample view of learned Castanet facets for the BBC Word News data set



**Current Path:** abstraction > measure > **old_age**

ALL 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- **previous**

**/Users/thuang513/research/C-Cat/extendOntology/test-docs/LibyaNews.txt**
**keywords:** million,old_age,one

With his eccentric , often inscrutable personality , Kadafi has ruled Libya , one of the world 's largest oil producers , for 41 years through a mix of repression , patronage and shrewd tribal alliances .

Two of the Arab world 's most ruthless leaders have moved to crush revolts threatening their power in Libya and Yemen as security forces and thugs intensified attacks on dissidents and protesters dug scores of fresh graves amid the rattle of gunfire .

Saleh , who once described ruling Yemen as `` dancing on the heads of snakes , '' has stayed in power for 32 years in much the same manner .

**/Users/thuang513/research/C-Cat/extendOntology/test-docs/AIDS.txt**
**keywords:** million,old_age,one

`` Thirty years ago this mystery disease was called a gay plague - it was a shunned disease , people were scared about each other , '' Mr Sidibe said .

Mr Sidibe said the challenge was to expand access to drugs , and deal with social factors that in some countries continue to stigmatise the disease and make women particularly vulnerable .

An estimated 34 million people were living with HIV at the end of 2010 and nearly 30 million have died from Aids-related causes since 1981 , the report said .

**/Users/thuang513/research/C-Cat/extendOntology/test-docs/Romney.txt**
**keywords:** million,old_age,one

Mr Romney blamed Mr Obama for the economic woes frustrating voters , like federal spending and unemployment during a speech in New Hampshire .

Our correspondent says that Mr Romney 's biggest problem is that as governor of Massachusetts he introduced a type of compulsory health insurance which looks very similar to President Obama 's reforms which are heartily loathed by most Republicans .

`` Government under President Obama has grown to consume almost 40 % of our economy , '' Mr Romney said .

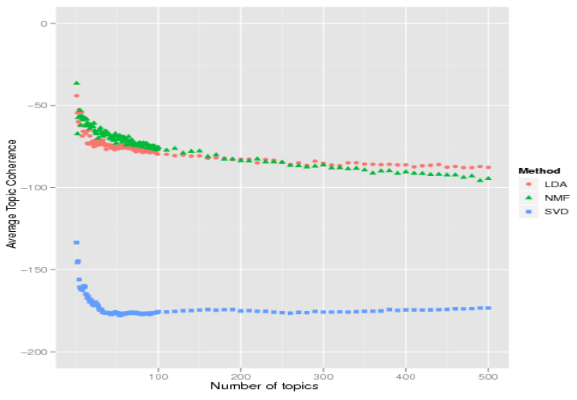
**/Users/thuang513/research/C-Cat/extendOntology/test-docs/TaxGas.txt**
**keywords:** million,old_age,one

In a news conference , Obama said that a payroll tax cut signed into law in December as part of the tax package would now go to cushion the impact of a recent spike in oil prices and allow for continued economic growth .

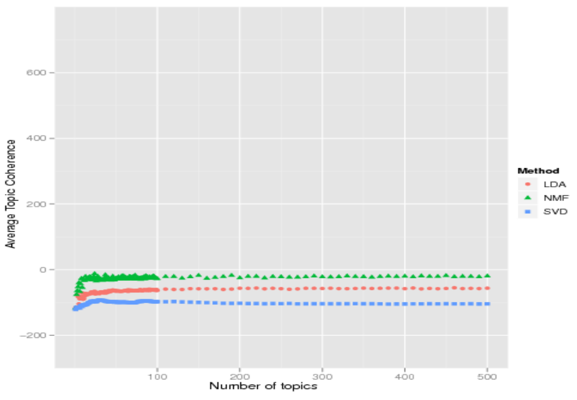
Obama pushed back against both arguments Friday , saying the pickup in oil prices is largely being driven by the global economic recovery , particularly the speed of growth in emerging economies , and does n't yet necessitate tapping the emergency reserve .

President Obama acknowledged Friday that the fast-rising cost of gasoline could diminish the effect of policies designed to stimulate economic growth , but warned that he is not yet prepared to unleash the nation 's energy reserves to bring down the price of oil .

Figure 11: A sample view of a discovered document after navigating the Castanet facets

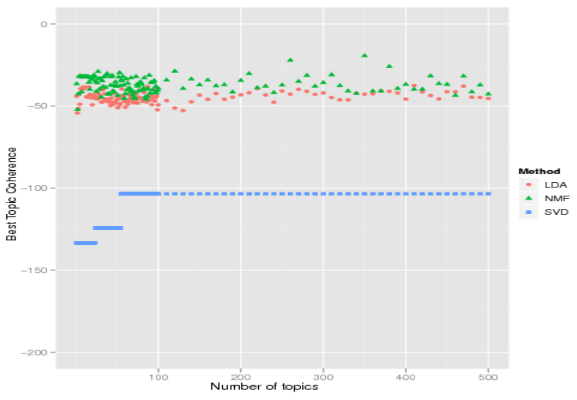


(a) UMass

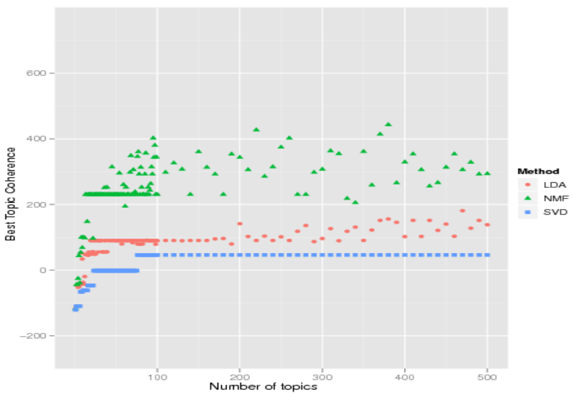


(b) UCI

Figure 14: Average Topic Coherence for each model

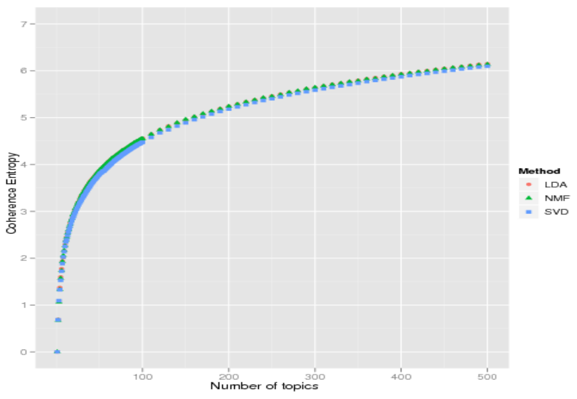


(a) UMass

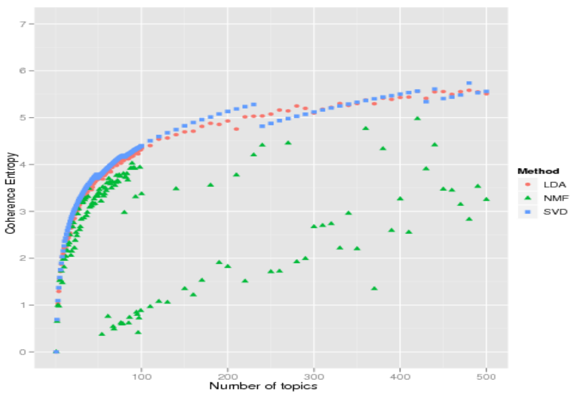


(b) UCI

Figure 15: Topic Coherence of the best topic for each model

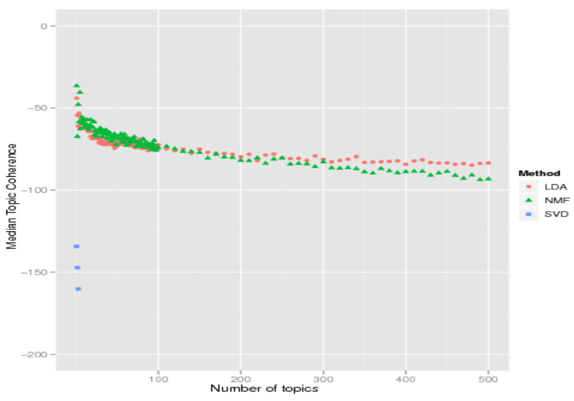


(a) UMass

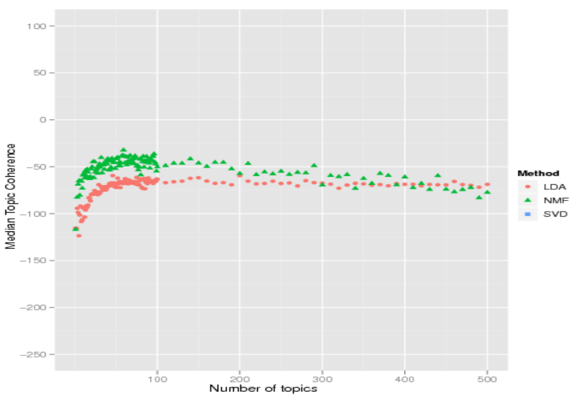


(b) UCI

Figure 16: Entropy of the Topic Coherence for each model

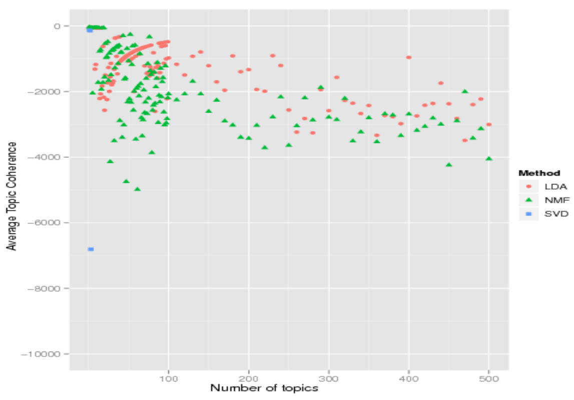


(a) UMass

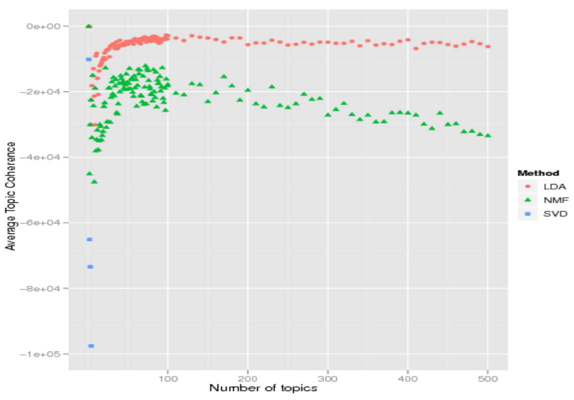


(b) UCI

Figure 17: Median Topic Coherence with $\epsilon = 10^{-12}$

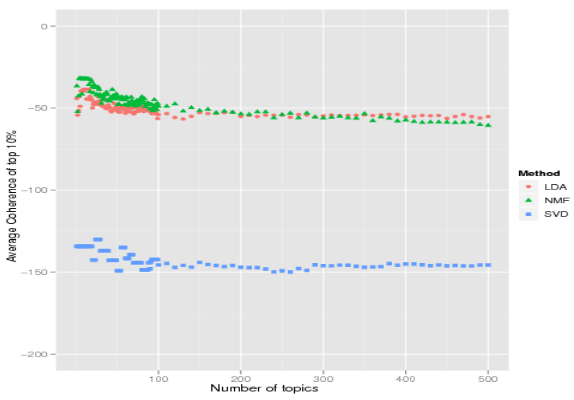


(a) UMass

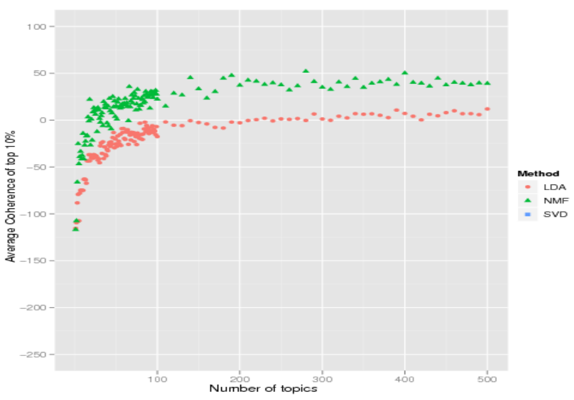


(b) UCI

Figure 18: Average Topic Coherence with $\epsilon = 10^{-12}$

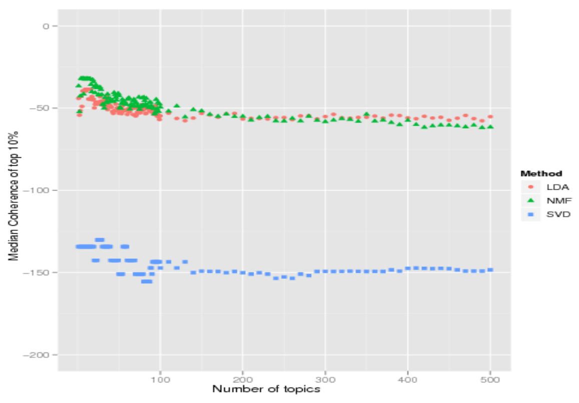


(a) UMass

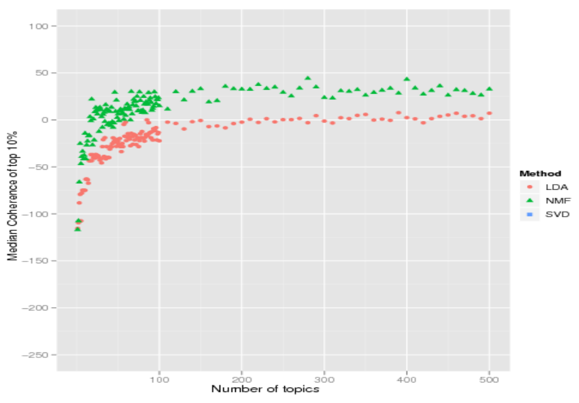


(b) UCI

Figure 19: Average Topic Coherence of the top 10% topics with $\epsilon = 10^{-12}$

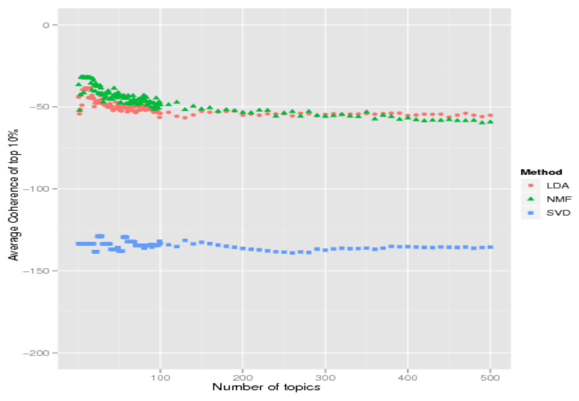


(a) UMass

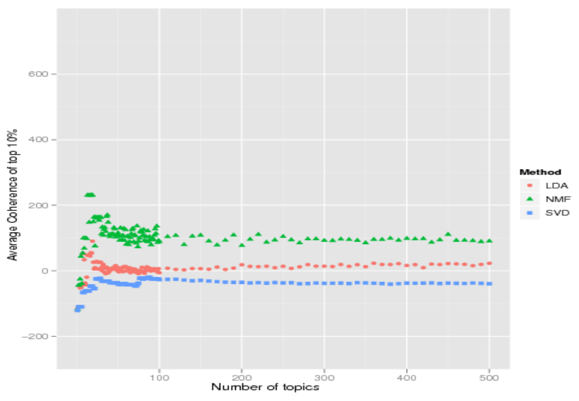


(b) UCI

Figure 20: Median Topic Coherence of the top 10% topics with $\epsilon = 10^{-12}$

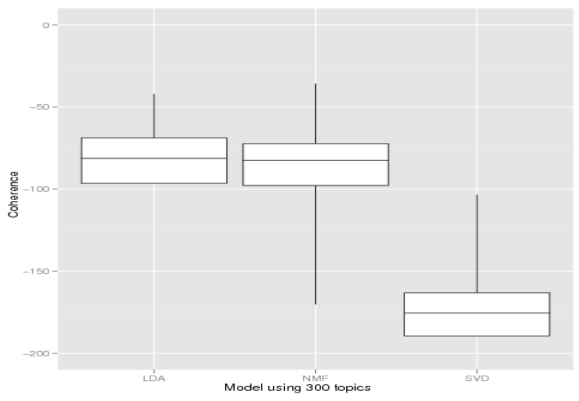


(a) UMass

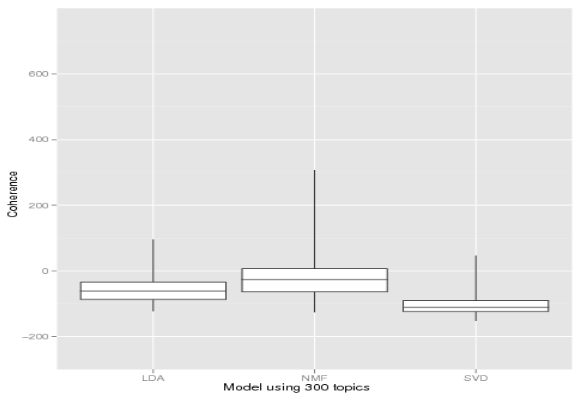


(b) UCI

Figure 21: Average Topic Coherence of the top 10% topics for each model

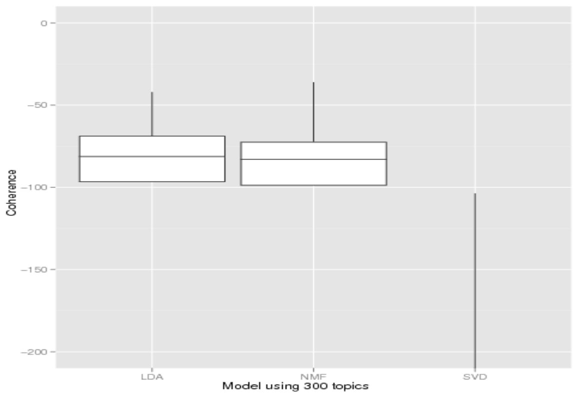


(a) UMass

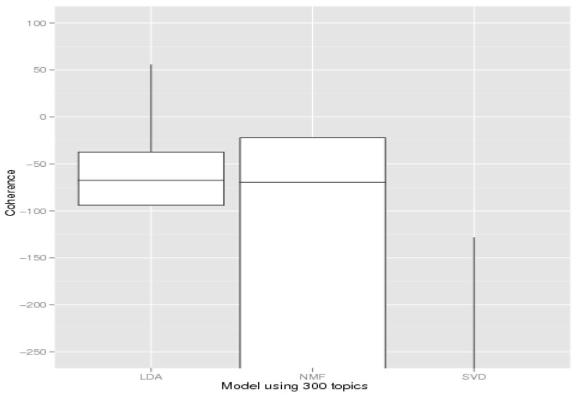


(b) UCI

Figure 22: Topic Coherence quartiles for models trained on 300 topics

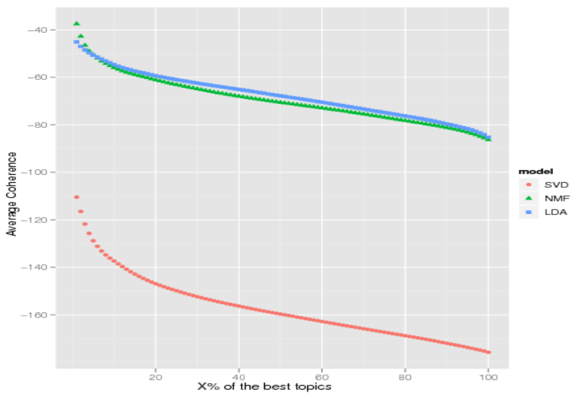


(a) UMass

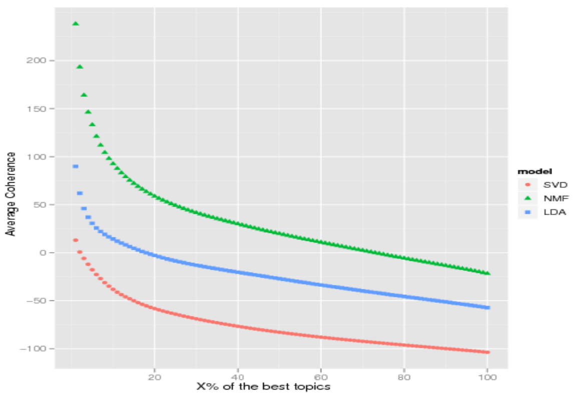


(b) UCI

Figure 23: Topic Coherence quartiles with $\epsilon = 10^{-12}$ for models trained on 300 topics

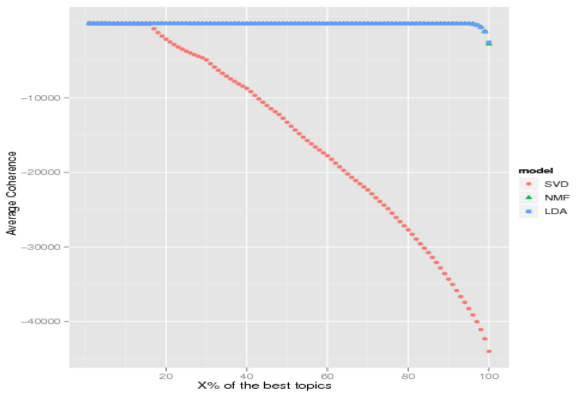


(a) UMass

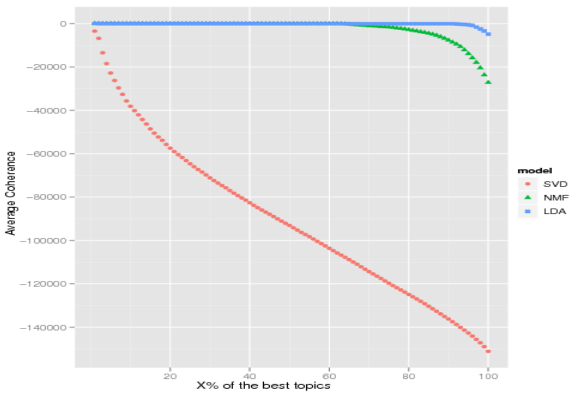


(b) UCI

Figure 24: Topic Coherence for the top X% topics out of 300 topics

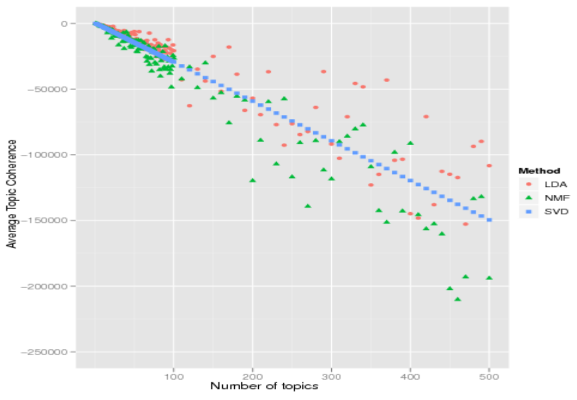


(a) UMass

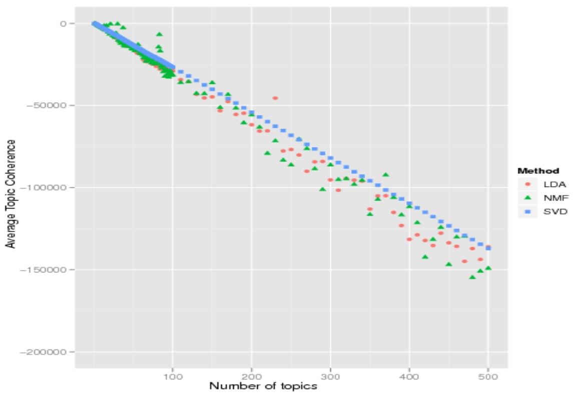


(b) UCI

Figure 25: Topic Coherence for the top X% topics out of 300 topics with $\epsilon = 10^{-12}$

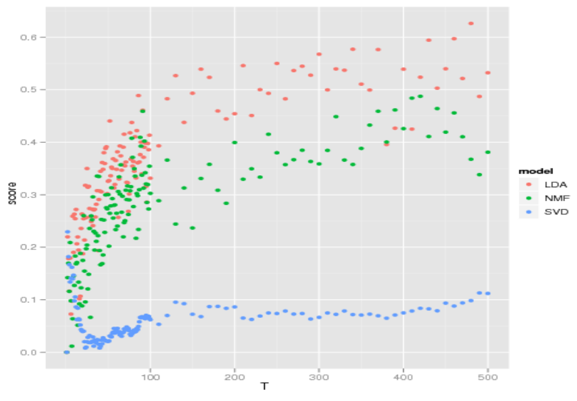


(a) UMass

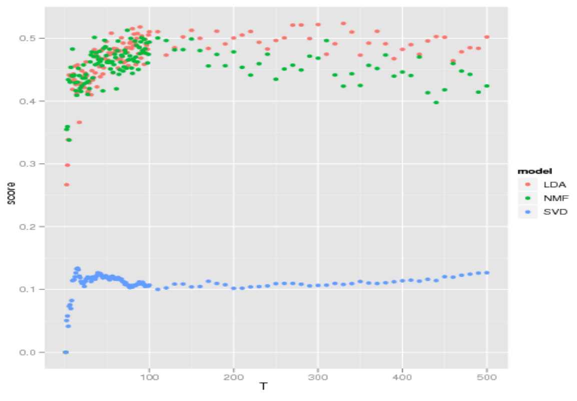


(b) UCI

Figure 26: Topic Uniqueness measures for each model



(a) Rubenstein & Goodenough



(b) Wordsim 353

Figure 27: Word Similarity Evaluations for each model