GnosisMiner: Reading Order Recommendations over Document Collections

Georgia Koutrika
HP Labs
Palo Alto, CA, USA

Alkis Simitsis
Hewlett Packard Labs
Palo Alto, CA, USA

Yannis Ioannidis
University of Athens and
"Athena" Research Center
Athens, Greece

ABSTRACT

Given a document collection, existing systems allow users to locate documents either using search keywords or by navigating through some predefined organization of the collection. Other approaches help the user understand a collection by generating summaries or clusters of the documents at hand. However, often users would like to understand how the documents may be related to each other and access them in some logical order. In this work, we present an interactive reading recommendation system, called *GnosisMiner*. Given a collection of documents and a theme, the system returns a partial order of documents relevant to that theme organized from more general to more specific. The recommended *reading order* resembles the human approach of learning as we typically start our path to knowledge from more general documents that help us understand the domain and then we proceed with more specific, more specialized documents to increase our knowledge of the matter.

1. INTRODUCTION

Given a document collection, existing systems allow users to locate documents either using search keywords or by navigating through some predefined organization of the collection. However, search engines hide document relationships, and navigational interfaces capture only fixed relationships that do not dynamically adapt to the users' specific needs. Therefore, while these systems work fine when users try to locate specific documents, they are insufficient when users would like to understand how the documents may be related to each other and access them in some logical order.

We advocate that given a document collection, it is very useful to recommend to a user a possible *reading order* over this collection so that she can access the documents in an organized, structured way. In the past, there have been efforts towards helping a user understand and access a corpus of documents in some meaningful way. These efforts include corpus summarization approaches, which try to generate a textual summary of the collection [9, 10], hierarchical document clustering methods, which segment the corpus [6, 7, 11], and document linking, which connect documents through specific types of links such as 'consequence of' or 'follow-up' [2, 8]. Google's advanced search interface [3] organizes search results into three reading levels: basic, intermediate, and advanced, offering a very coarse document ordering.

In this work, we present an interactive reading recommendation system, called *GnosisMiner*. Given a collection of documents and given a theme (i.e., a set of keywords), our system returns a partial order of documents relevant to that theme organized from more general to more specific. The recommended *reading order* resembles the human approach of learning as we typically start our path to knowledge from more general documents that help us understand the domain and then we proceed with more specific, more specialized documents to increase our knowledge of the matter.

GnosisMiner represents a reading order as a tree and users may select which path on the tree they would like to follow and, hence, which documents they would like to read in order. To help them further, the system shows the topics found in each document as well at each level of the tree. Users may modify the recommended reading orders through parameters that determine how fine-grained the ordering should be. Finally, the system provides several visualizations of the underlying collection aimed at the expert user who would like to gain insights into the similarities and topics of the documents and tune the recommendations accordingly.

Recommending reading orders is useful in many areas, such as (*a*) education, for organizing online educational material, (*b*) patent searching, for helping users (e.g. patent attorneys) to understand which patents have more general cover than others, (*c*) research, for organizing publications or news articles to help researchers study a topic, (*d*) publishing, for helping editors select and organize articles to publish on a web site, and so forth.

2. RECOMMENDING READING ORDERS

Given a collection of documents, a reading order is a partial order of the documents from general to more specific documents. In this partial order, there are two types of document relationships, equivalence and precedence, which can be informally described as follows. If two documents are about the same topics, then they are considered *equivalent*, denoted $a \leftrightarrow b$, and are grouped together. If they are about related topics but document b is more specific than a, then a *precedes* b in the order, denoted $a \to b$.

As an example, consider the following documents: a is an introduction to data mining, b is on classification methods, and c is another introductory document on data mining. Both a and c cover the same topic to a similar extent and hence they are considered equivalent. Consequently, one can choose to read any of them. However, b is more focused, hence it has precedence relationships to the other documents: $a \to b$ and $c \to b$.

We quantitatively define the equivalence and precedence relationships in a reading order using two metrics: *document generality* and *document overlap*. We will first describe the metrics and then show how the two relationships are defined with their help.

Series ISSN: 2367-2005

10.5441/002/edbt.2017.64

Given documents a and b, the document generality for a document a is captured by the *generality score*, $g(a)$, a real number such that higher values mean higher generality. In other words, a is more general than b iff $g(a) \geq g(b)$. The document overlap for the pair a and b is captured by the *overlap score*, $o(a, b)$, a real number typically in the range of $[0, 1]$, where 0 means no overlap between a and b, and 1 means maximum overlap.

We measure document overlap and generality based on the documents' topical relationships. To derive the topics describing the documents, we use topic modeling. Topic models [1] are based upon the idea that documents are mixtures of topics, where a topic is a probability distribution over words. A topic model aims at discovering the hidden thematic structure of a collection of documents by finding how topics are assigned to documents, and how topics are described by words in the documents. Representing a document using topics rather than document keywords is more effective because it allows capturing implicit relationships between documents, not just the explicit similarity of their common words.

Document generality. We compute the document generality as a measure of the document's entropy over the topics it covers. The basic intuition behind the entropy is that the higher a document's entropy is, the more topics it covers in less depth hence the more general it is. Given a collection $\mathcal{D}$ of n documents and s topics, we denote $F_{n \times s}$ the document-topic matrix that captures how the s topics are assigned to the n documents in $\mathcal{D}$. $F_{ij} \in [0, 1]$ with $i \leq n$ and $j \leq s$ describes how well topic t_j describes document a_i. Using the Shannon entropy, the *generality score* $g(a_i)$ of document a_i can be defined as follows:

$$g(a_i) = H(a_i) = \sum_j -F_{ij} \log(F_{ij}) \tag{1}$$

Document overlap. The topic overlap $o(a, b)$ of two documents a and b can be defined using the weighted Jaccard score [4]. The weighted Jaccard extends the classic Jaccard index, which is defined as the size of the intersection divided by the size of the union of the topic sets assigned to each document, by taking into account how well a topic represents a document. The topic overlap can be defined as follows:

$$o(a, b) = Jaccard(a, b) = \frac{F_a \cdot F_b}{|F_a|^2 + |F_b|^2 - F_a \cdot F_b} \tag{2}$$

where F_a (F_b) is the topic vector associated with a (b, resp.). Larger values indicate more common topics between two documents.

Note that other metrics for measuring document generality and overlap are possible. For example, instead of the Shannon entropy, we could use the residual entropy (entropy of non-common terms) or the distribution entropy (entropy of the location of common, non-common, or both types of terms throughout the document).

Now we can formally define the document equivalence and precedence relationships in a reading order as follows:

document equivalence: $a \leftrightarrow b$ iff $|g(a) - g(b)| \leq \kappa \wedge o(a, b) \geq \tau$ (3)

$$\textit{document precedence: } a \rightarrow b \text{ iff } g(a) > g(b) \wedge o(a, b) > 0 \\ \wedge \; (|g(a) - g(b)| > \kappa \;\vee\; o(a, b) < \tau) \tag{4}$$

τ defines the minimum topic overlap between two equivalent documents and κ defines the maximum difference of their generality scores.

Figure 1(a) shows an example reading order over six documents. A user can follow different reading paths following the document relationships, such as the example reading path: $d_1 \rightarrow d_4 \rightarrow d_6$, shown in the figure. Furthermore, there may be more than one reading orders for the same set of documents. For example, consider documents d_1, d_2 and d_3, which have some overlap and $d_1 \leftrightarrow d_2$ and $d_2 \leftrightarrow d_3$ but d_1 is not equivalent to d_3. Figure 1(b) shows two possible reading orders.

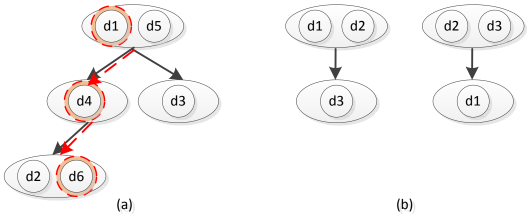


Figure 1: Example reading orders

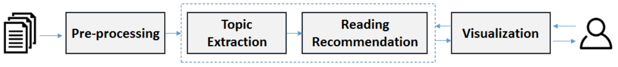


Figure 2: System architecture

3. SYSTEM OVERVIEW

We present GnosisMiner, a prototype system for recommending ordered readings over document collections. The system architecture is depicted in Figure 2. Its main components are: *preprocessing*, *topic extraction*, *reading recommendation*, and *visualization*. The users interact with the system through the visualization component to specify the document collection and the theme of their interest, interact with the recommended reading order and the documents, modify the recommendation parameters, and examine the various visualizations over the collection.

Next, we describe the main components of our system. For more details on the algorithms used for topic extraction and reading order recommendation we refer the interested reader to [5].

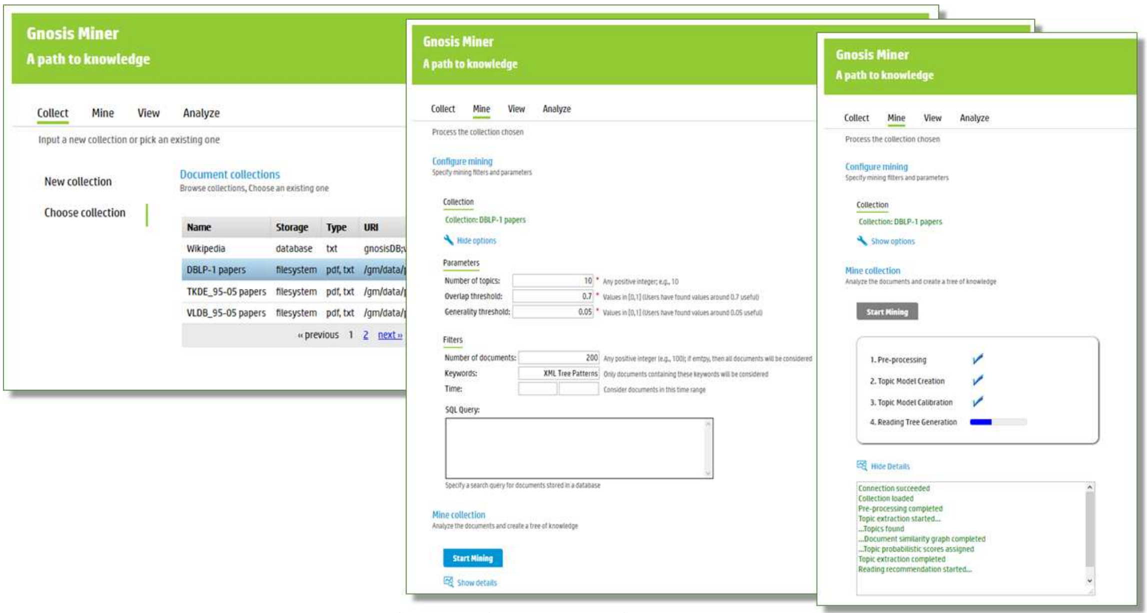
3.1 Pre-processing

Pre-processing removes noisy and stop words, performs stemming, and transforms each document to a term vector using a tf-idf weighting scheme. We perform this task incrementally; we skip documents already processed in a previous run and only work on documents never processed before.

3.2 Topic extraction

To measure the generality and overlap of the documents, and identify the topical relationships among them, we first derive the topics that describe the documents. The *topic extraction module* works in two phases. First, it extracts the topics that occur in the documents using the Latent Dirichlet Allocation (LDA) model with Gibbs sampling [1]. However, topic models often misassign or miss topics for documents. To reduce such errors, subsequently, the topic extraction uses a score propagation method that allows the topic scores of a document to be influenced by the topics of its most similar neighbors. This module leverages the content similarity of the documents by comparing their term representations and propagates document-topic scores between strongly similar documents on the basis that due to their similarity they likely have similar topics.

For this purpose, this module first builds the document similarity graph, where each node maps to a document, and each edge between two documents captures their similarity (i.e., the similarity of their term-based representations) in the edge weight. Then, the document-topic scores, returned by LDA during the first step of topic extraction, are propagated over the document similarity graph, so the potential topics of a document take into consideration the topic scores of their neighbors (which in turn, depend on the scores of their respective neighbors, and so on). The algorithm iteratively updates the topic scores of a node (document) based on the weighted average of the scores of its neighbors.



Input a new collection or pick an existing one

Document collections

Browse collections, Choose an existing one

Name	Storage	Type	URI
Wikipedia	database	txt	gnosisDB;
DBLP-1 papers	filesystem	pdf, txt	/gm/data/
TKDE_95-05 papers	filesystem	pdf, txt	/gm/data/
VLDB_95-05 papers	filesystem	pdf, txt	/gm/data/

« previous 1 2 next »

Process the collection chosen

Configure mining

Specify mining filters and parameters

Collection

Collection: DBLP-1 papers

Parameters

Number of topics: 10 * Any positive integer; e.g., 10

Overlap threshold: 0.7 * Values in [0,1] (Users have found values around 0.7 useful)

Generality threshold: 0.05 * Values in [0,1] (Users have found values around 0.05 useful)

Filters

Number of documents: 200 Any positive integer (e.g., 100); if emtpy, then all documents will be considered

Keywords: XML Tree Patterns Only documents containing these keywords will be considered

Time: Consider documents in this time range

SQL Query:

Specify a search query for documents stored in a database

Mine collection

Analyze the documents and create a tree of knowledge

1. Pre-processing ✓
2. Topic Model Creation ✓
3. Topic Model Calibration ✓
4. Reading Tree Generation

```
Connection succeeded
Collection loaded
Pre-processing completed
Topic extraction started...
...Topics found
...Document similarity graph completed
...Topic probabilistic scores assigned
Topic extraction completed
Reading recommendation started...
```

Figure 3: Collect and Mine components

3.3 Reading Recommendation

The *reading recommendation* uses the document-topic scores learnt from the topic extraction module to determine the equivalence and precedence relations among the documents and mine a reading order for them. This module takes as input a set of documents, the document-topic assignments, and the parameters τ, which defines the minimum topic overlap between two equivalent documents, and κ, which defines the maximum difference of their generality scores. The module uses an iterative method to build a tree that represents the recommended reading order.

In this tree, nodes correspond to the input documents and edges capture precedence relationships between the documents. In particular, a node maps a non-empty set of equivalent documents. An edge between nodes A and B signifies that documents belonging to the corresponding document set of A precede the documents belonging to the respective node B.

For the root of this tree, the method puts together the most general, equivalent documents (i.e, documents whose generality difference is small ($< \kappa$) and whose overlap is high ($> \tau$)). From the remaining documents, the method creates clusters of documents that can be grouped together because they overlap with each other and they also have some overlap ($0 <$ and $< \tau$) with the root of the tree. Each of these clusters will be used to grow a subtree that will be connected to the current node (let's call it the parent node of the cluster) in subsequent rounds. The reading recommendation module takes each of the clusters created in the previous cycle and selects the most general, equivalent documents. This set becomes a new node that is added under the parent node of the cluster. Then, the remaining documents are clustered. Note that in each clustering step, documents that were un-clustered before may get grouped now. This process repeats until no more tree growing is possible and there are no documents unprocessed.

3.4 Visualization

The *visualization component* allows the user to interact with the system. The user can specify the document collection they would like to explore and the theme of their interest (e.g., as a set of keywords), interact with the recommended reading order of the documents, modify the recommendation parameters, and examine various visualizations over the collection.

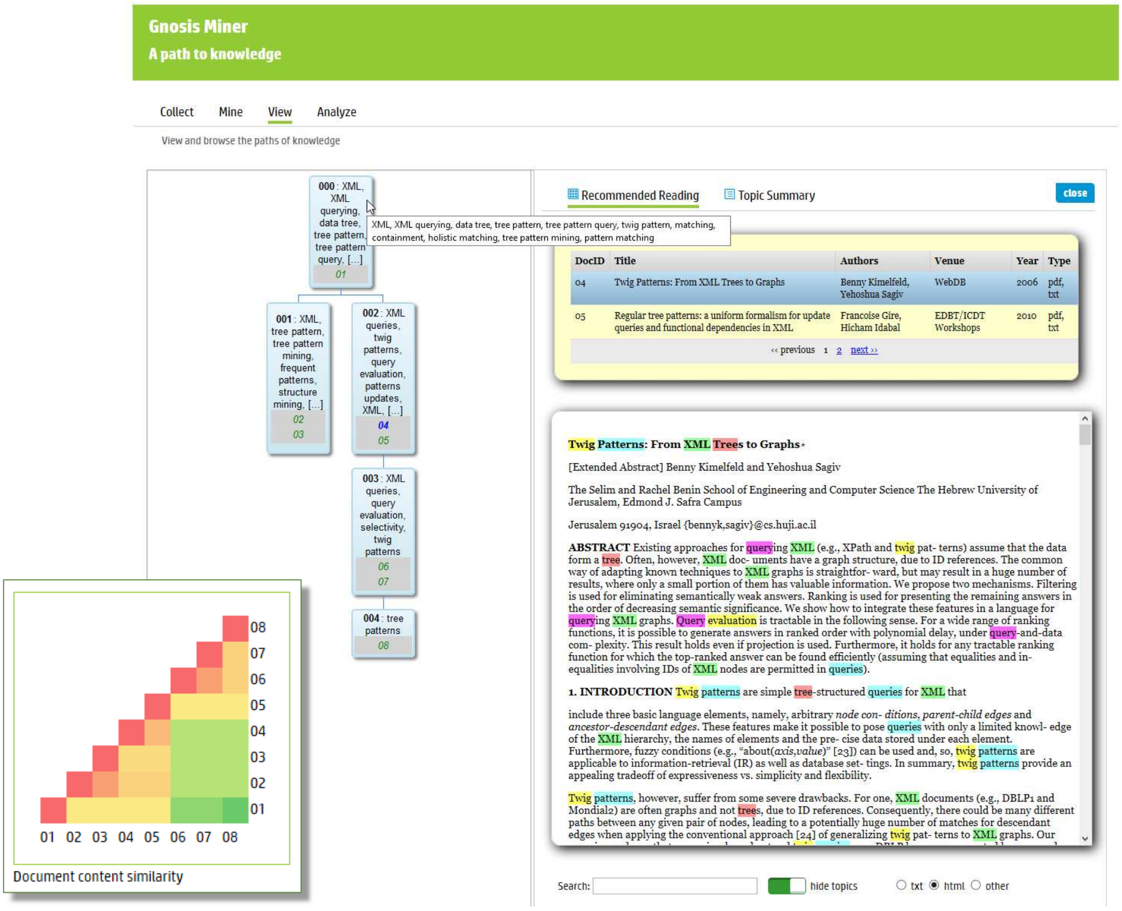
GnosisMiner visualizes a reading order as a tree, where each node corresponds to a set of equivalent documents. The system shows the topics found in each document as well as at each node of the tree. The user can interact with the tree in a table-of-contents manner, and choose which documents to read in the proposed order. The user can modify the recommended reading order through parameters that determine how fine-grained the ordering should be. These parameters include the number of topics to use for describing the documents of interest, the minimum topic overlap (τ) between equivalent documents, and the maximum difference (κ) of their generality scores.

Finally, the system provides several visualizations of the underlying collection that offer a look under the hood at the document relationships as well as at the operation and performance of the system. The user can visually examine the topics describing the selected set of documents, how these topics are assigned to documents, the document content similarities, and their generality scores. For instance, the document content similarities are visualized using a heatmap. The user can also review details regarding the operation of the various components of the system, such as execution times, number of iterations of the topic extraction, number of iterations of the reading recommendation, and so forth.

4. OUR PRESENTATION

Our presentation will demonstrate GnosisMiner's features using a collection of data management related papers as our corpus. Our demonstration script starts with a small number of representative examples. With these examples we will show how a user can choose a collection and specify which part of the collection she is interested in. For example, Figure 3 shows an example navigation and run of the system, using a collection of papers, example parameters for topic extraction, and an example filter limiting the search to 200 papers with a theme 'XML Tree Patterns'. When a reading order has been generated, it is shown in the View component.

Figure 4 shows a snapshot of an abridged result for this example. The left panel contains the tree representing the reading order chosen. Each node contains a set of topics along with links to papers related to those topics. Hovering over a node shows the complete


Gnosis Miner
A path to knowledge
Collect
Mine
View
Analyze
View and browse the paths of knowledge
000 : XML, XML querying, data tree, tree pattern, tree pattern query, [...]
01
001 : XML, tree pattern, tree pattern mining, frequent patterns, structure mining, [...]
02
03
002 : XML queries, twig patterns, query evaluation, patterns updates, XML, [...]
04
05
003 : XML queries, query evaluation, selectivity, twig patterns
06
07
004 : tree patterns
08
Recommended Reading
Topic Summary
close
‹‹ previous
1
2
next ››
Search:
hide topics
○ txt
● html
○ other
01
02
03
04
05
06
07
08
Document content similarity

XML, XML querying, data tree, tree pattern, tree pattern query, twig pattern, matching, containment, holistic matching, tree pattern mining, pattern matching

DocID	Title	Authors	Venue	Year	Type
04	Twig Patterns: From XML Trees to Graphs	Benny Kimelfeld, Yehoshua Sagiv	WebDB	2006	pdf, txt
05	Regular tree patterns: a uniform formalism for update queries and functional dependencies in XML	Francoise Gire, Hicham Idabal	EDBT/ICDT Workshops	2010	pdf, txt

Twig Patterns: From XML Trees to Graphs*

[Extended Abstract] Benny Kimelfeld and Yehoshua Sagiv

The Selim and Rachel Benin School of Engineering and Computer Science The Hebrew University of Jerusalem, Edmond J. Safra Campus

Jerusalem 91904, Israel {bennyk,sagiv}@cs.huji.ac.il

ABSTRACT Existing approaches for querying XML (e.g., XPath and twig pat- terns) assume that the data form a tree. Often, however, XML doc- uments have a graph structure, due to ID references. The common way of adapting known techniques to XML graphs is straightfor- ward, but may result in a huge number of results, where only a small portion of them has valuable information. We propose two mechanisms. Filtering is used for eliminating semantically weak answers. Ranking is used for presenting the remaining answers in the order of decreasing semantic significance. We show how to integrate these features in a language for querying XML graphs. Query evaluation is tractable in the following sense. For a wide range of ranking functions, it is possible to generate answers in ranked order with polynomial delay, under query-and-data com- plexity. This result holds even if projection is used. Furthermore, it holds for any tractable ranking function for which the top-ranked answer can be found efficiently (assuming that equalities and in- equalities involving IDs of XML nodes are permitted in queries).

1. INTRODUCTION Twig patterns are simple tree-structured queries for XML that

include three basic language elements, namely, arbitrary *node con- ditions*, *parent-child edges* and *ancestor-descendant edges*. These features make it possible to pose queries with only a limited knowl- edge of the XML hierarchy, the names of elements and the pre- cise data stored under each element. Furthermore, fuzzy conditions (e.g., "about(*axis,value*)" [23]) can be used and, so, twig patterns are applicable to information-retrieval (IR) as well as database set- tings. In summary, twig patterns provide an appealing tradeoff of expressiveness vs. simplicity and flexibility.

Twig patterns, however, suffer from some severe drawbacks. For one, XML documents (e.g., DBLP1 and Mondial2) are often graphs and not trees, due to ID references. Consequently, there could be many different paths between any given pair of nodes, leading to a potentially huge number of matches for descendant edges when applying the conventional approach [24] of generalizing twig pat- terns to XML graphs. Our

Figure 4: View component (main picture) and example analysis chart: similarity heatmap (bottom-left corner)

list of topics of the node. In the figure, the root node of the tree contains the paper entitled "*M. Hachicha, J. Darmont: A Survey of XML Tree Patterns. IEEE Trans. Knowl. Data Eng. 25(1): 29-46 (2013)*", which is a survey paper on the specified subject matter, and it covers several topics in XML patterns. We observe that under this node, nodes *001* and *002* cover more focused topics: the former related to tree pattern mining, frequent patterns, structure mining, and so forth, and the latter related to XML queries, twig patterns, query evaluation, etc.

Selecting a paper link in a node opens the right panel, which shows the recommended list of papers for the corresponding set of topics and a summary listing of these topics. Choosing a paper from the list, shows the text in a viewer. For instance, in the figure, the document entitled "*Twig Patterns: From XML Trees to Graphs*" is viewed. A user can read the paper, perform a text search, and so on. There is also a show/hide topics feature that highlights the relevant topics in the text (enabled in the figure).

The advanced user may use the Analyze component to examine analysis charts (e.g., document-topics assignments, document similarities, performance statistics) to get insights into the document collection and refine the mining process if needed. The bottom-left corner of Figure 4 shows an example snapshot of a similarity heatmap for the 8 documents shown in the reading tree of Figure 4. For instance, one can see how documents *02 - 05* are closer in similarity to *01*, which is the root of the tree in the figure, while *06 - 08* are more distant. One could also see that a slightly more flexible similarity threshold could group documents *06 - 08* together.

Finally, for off-script presentation and discussion, we will provide interactivity, where the participants can explore the data set themselves and experiment with GnosisMiner.

5. REFERENCES

[1] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[2] Ao Feng and James Allan. Incident threading for news passages. In CIKM, 2009.

[3] Google. Advanced search interface. Available at: http://www.google.ca/advancedsearch, 2016.

[4] G. Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, 1994.

[5] Georgia Koutrika, Lei Liu, Steven J. Simske. Generating reading orders over document collections. ICDE, 507-518, 2015.

[6] Qirong Ho, Jacob Eisenstein, and Eric P. Xing. Document hierarchies from text and links. In WWW, 2012.

[7] Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. Incremental hierarchical clustering of text documents. In CIKM, 2006.

[8] Dafna Shahaf and Carlos Guestrin. Connecting two (or less) dots: Discovering structure in news articles. *ACM Trans. Knowl. Discov. Data*, 5(4):24:1–24:31, February 2012.

[9] B. Shaparenko and T. Joachims. Information genealogy: Uncovering the flow of ideas in non-hyperlinked document databases. In KDD, 2007.

[10] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. Temporal corpus summarization using submodular word coverage. In CIKM, 2012.

[11] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data Min. Knowl. Discov.*, 10(2), March 2005.