Navigating the topical structure of academic search results via Wikipedia category network

ABSTRACT

Searching for scientific publications on the Web is a tedious task, especially when exploring an unfamiliar domain. Typical scholarly search engines produce lengthy unstructured result lists that are difficult to comprehend, interpret and browse. We propose a novel method of organizing the search results into concise and informative topic hierarchies. The method consists of two steps: extracting interrelated topics from the result set, and summarizing the topic graph. In the first step we map the search results to articles and categories of Wikipedia, obtaining a graph of relevant topics linked with hierarchical relations. In the second step we sequentially build nested summaries of the produced topic graph using a structured output prediction approach. Trained on a small number of examples, our method learns to construct informative summaries for unseen topic graphs, and outperforms unsupervised state-of-the-art Wikipedia-based clustering. We implement our method in an online tool working on top of the search API of an existing academic search service, enabling it with topic summary-based visualization and browsing.

1. INTRODUCTION

We search for publications on a daily basis – to keep up with our research fields, to expand our competences, to find related work. The number of papers being published is far beyond what a scientist can consume, so we have to be very selective in what we read or even look through. Scholarly digital libraries and search engines allow us to easily find papers provided we know, at least partially, their title or other bibliographic data, such as authors, publication venue and year. The way the search results are presented – the infamous 'ten blue links' – however, proves insufficient for more complex yet typical information seeking tasks. Whether reviewing related work or exploring a new research domain, we often want to perceive the returned results as a whole, identify the constituent topics, understand their span and interrelations. We would also like to find papers more efficiently without sifting through the overlong lists of mostly irrelevant items. Grouping the search results according to salient topics, were we be able to identify them, could help in this regard. Concise and structured visual representation of the topics would provide both the global picture and the means to focus on the relevant details.

Predefined taxonomies provide a natural way of grouping the scholarly search results in a given scientific discipline. Computer science publications, for instance, can be organized with respect to `ACM Computing Classification System (CSS)`, and biomedical articles – with respect to `Medical Subject Headings (MeSH)`. One of the benefits of such predefined taxonomies is that they provide hierarchical grouping of publications, which can be viewed at desired level of granularity. Another benefit is that the groups have well-defined semantics and meaningful labels corresponding to the subtopics of the discipline in question. Hierarchical relations between the subtopics provide additional useful information about the structure of the field in a visual way.

The main drawback of the predefined taxonomies is that they have to be manually created and maintained up to date, which requires significant effort. For example, ACM reports 120 computing specialists having worked on the new version of `Computing Classification System`[1]. Assigning new articles to existing categories manually is impractical at the scale of a search engine like `Google Scholar`, when the collection of publications is expanded in an automated way while crawling the Web. Categorizing the articles automatically, on the other hand, is a nontrivial task.

Another shortcoming of the predefined taxonomies is that they often represent a rather coarse-grained structure of the field. For instance, `ACM CSS` may tell that *Data mining* is a subfield of *Information systems applications* and includes *Association rules*, *Collaborative filtering* and *Clustering*. *Clustering* is, however, a leaf node in the taxonomy, and no information about its subfields is available.

Unsupervised learning methods, such as clustering, self-organizing maps or latent topic models, constitute an alternative approach to grouping and visualization of the scholarly search results. The advantage of this approach is that grouping is completely automatic, while the main drawback is that the results of the grouping are not easily interpretable. Firstly, some of the discovered groups may simply not correspond to distinct topics relevant to a human user.

[1] http://www.acm.org/news/featured/2012-acm-ccs

Secondly, most of the unsupervised learning techniques provide no meaningful labels for the groups; and although there have been research efforts towards generating labels for document collections [37, 31, 39], the results are not yet as expressive and meaningful as manually created topic names.

Another limitation inherent to most of the unsupervised grouping methods is the necessity of choosing the number of groups (clusters, topics, etc.), with hierarchical clustering being a notable exception. In order to discover meaningful topics in the result set, unsupervised models have to be previously built on a large corpus of data, a "universal dataset" [29], which is usually a computationally expensive task that cannot be performed on the fly. For most of the methods this implies choosing all the parameters, such as the number of topics, beforehand and once and for all further inputs. The chosen topic granularity has to remain fixed, even though it may not be optimal for specific users and queries.

Wikipedia is a large online encyclopedia that is collaboratively edited by Web users. In this work we propose using articles and categories of Wikipedia for grouping the results of academic search. Containing over 4 million articles in English alone, Wikipedia covers a broad range of subjects with considerable level of detail. For instance, considering our previous example, Wikipedia contains not only an article on *Clustering*, but also articles on a variety of specific clustering methods, such as *K-means*, *Spectral clustering*, *DBSCAN* and *BIRCH*, and different clustering criteria, such as *Rand index* and *Fowlkes-Mallows index*. Recently developed techniques allow annotating texts with links to Wikipedia articles [9, 26, 13, 25], which provides the basis for grouping texts according to article-based topics. Moreover, articles in Wikipedia are arranged into categories that form a subsumption hierarchy and can be viewed as higher-level topics. The network of categories provide for informative visual representation of topics, with possibility to focus on a desired level of generality. Because of the described properties, Wikipedia can be viewed as constantly collaboratively updated fine-grained taxonomy of topics with available mechanisms of assigning texts to its nodes. Using Wikipedia for representing academic search results thus combines most of the advantages and avoids most of the drawbacks of both predefined taxonomies and unsupervised learning methods.

In this paper we propose grouping academic search results, and document collections in general, according to concise hierarchical topic summaries derived from Wikipedia (see Figure 1 for an example). Given a document collection, we first build a large graph of relevant topics with the help of topic annotators, and then select the most informative subgraph thereof to serve as a topic summary. Selecting the summary that most informatively represents the documents is a challenging problem, first, due to its combinatorial nature, and second, due to the subjectivity and subtlety of the notion of informativeness. In this paper we propose a novel approach to building informative summaries through structured output prediction. We construct the summary subgraphs sequentially, 'growing' them from smaller subgraphs, which allows us to alleviate the complexity of the problem, while maintaining the collective contribution of topics into the quality of the summary. In order to capture the properties important for good topic summaries, we learn how to grow such summaries from given examples. We demonstrate that our method is able to learn from a small number of examples, and produce informative topic summaries for unseen document collections.

In the following Section 2 we review existing approaches to representing search results and finding topics in document collections. In Sections 3 and 4 we show how to extract topic hierarchies from Wikipedia and describe our method of topic graph summarization. In Section 5 we report on the evaluation of the proposed method. We present `S*******n` – a prototype web tool for browsing academic search results in Section 6, and conclude the paper in Section 7.

2. RELATED WORK

The benefits of the structured representation of search results have long been realized in the area of general Web search. Cutting et al. [10] first introduced a clustering method, Scatter/Gather, as a metaphor for exploring document collections, while Hearst and Pedersen evaluated this method in the context of Web search [18]. A variety of clustering algorithms have since been applied to Web search results. We refer the reader to [7] for an extensive survey on the subject. The problem of choosing the meaningful cluster labels has always remained crucial. Selecting the labels after and independently of the clustering phase is a difficult problem. Most of the approaches produce cluster labels as sets of frequent words that are not grammatically or otherwise connected [7]. Zamir and Etzioni [42] suggested using a suffix tree for discovering phrases to form initial cluster seeds and serve as cluster labels. Another influential work of Osiriski et al. [30] represents the class of methods built around the central goal of providing the meaningful cluster labels. This work is also an early example of using a dimensionality reduction technique (Singular Value Decomposition [12]) for discovering topics in search results. Other data mining techniques that have been applied to this problem include agglomerative clustering [24], k-means clustering [41], concept lattices [8], and probabilistic topic models [29].

An important distinction of our work is that it targets academic search rather then general Web search. Organizing the results of academic search has not been discussed widely in the literature. However, there has been a substantial amount of research into organizing publication collections, discovering and visualizing scientific topics and identifying research trends. Some of the developed methods can be applied generally for document datasets, while others exploit features specific to research papers, such as citations. Probabilistic topic models represent a class of methods that have been extensively employed in the context of scientific papers. Griffith and Steyvers [15] applied Latent Dirichlet Allocation (LDA) [4] – a general-purpose topic model – to a collection of abstracts from PNAS. A correlated topic model [3] developed by Blei and Lafferty improved upon LDA by introducing pairwise topic correlations. Pachinko allocation [23] allowed more complex and sparse topic correlations by modeling topic mixtures through directed acyclic graphs. It is also an example of a topic model with hierarchical relations between topics, another example being hLDA (Hierarchical Latent Dirichlet Allocation) [1]. Specific topic models have been developed to account for various aspects of scientific literature, such as explicit document authorship

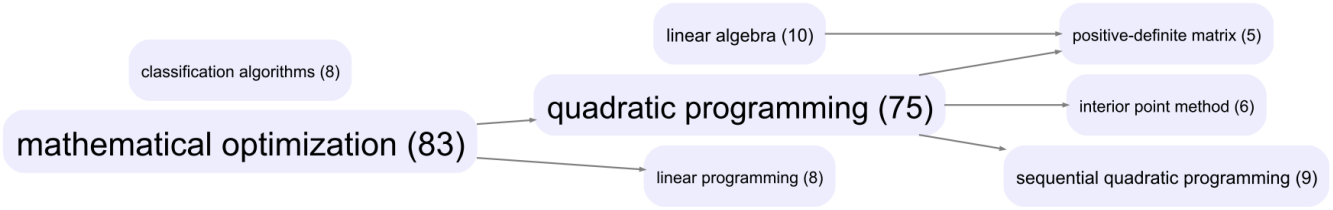

Figure 1: Example summary of 100 search results for the query "quadratic programming".

[32], author interests [22], publication venues [38], temporal ordering of the documents [5], topic evolution [2, 17] and citations [40, 28, 11, 17]. As we already mentioned in the introduction, the main drawback of topic models is that they do not address the problem of meaningful topic labeling.

Using external knowledge sources, such as Wikipedia, for organizing search results and documents in general has also been discussed in the literature. Gabrilovich and Markovitch [14] proposed representing texts as weighted combination of concepts based on Wikipedia articles for the purpose of computing semantic relatedness between texts. Jinarat et al. [20] mapped search results to the categories of the Open Directory Project in order to improve search result clustering. Jiang et al. [19] used WordNet categories to select the number of clusters and initial cluster centroids in k-means algorithm applied to search results. Sameh and Kadray [35] modified the frequent phrase extraction in the Lingo algorithm by generating phrase synonyms using WordNet.

Wikipedia has also been used in a number of works on search result clustering. Han and Zhao [16] proposed grouping the search results according to topics defined as communities in the graph of semantic relatedness between concepts. Săcărea et al. [34] exploited redirects and disambiguation pages of Wikipedia to improve the clustering algorithm based on the formal concept analysis. In the work of Calli et al. [6] semantic relations derived from Wikipedia were used to improve the performance of the Suffix Tree Clustering algorithm. Similarly to our work, Scaiella et al. [36] annotated the search result snippets with links to Wikipedia articles. The grouping of the results in their approach is performed based on the spectral clustering of the graph of snippets and topics. In contrast to these works, we use both articles and categories of Wikipedia to directly represent topics in the result set, and rely on their hierarchical relations for further grouping and visualization. More importantly, starting from a document-topic graph, we cast the further summarization problem into structured output prediction, which allows us to learn a scoring function combining multiple different features in a non-trivial way.

3. BUILDING THE TOPIC GRAPH

In this section we describe the procedure of extracting the hierarchy of topics – valid, useful and relevant for a collections of documents – from Wikipedia. As an input we assume to have a collection of documents $D = \{d_1, d_2, ..., d_N\}$. In the scenario of academic search, the documents may be publication abstracts retrieved by an academic search engine for some query. The result of this step is a topic graph $G(V, E)$, in which links $(v, v') \in E$ represent hierarchical parent-child relations between the topics, and a relation $R \subseteq V \times D$, which defines which documents are relevant to which topics. In order to present a valid hierarchy, the topic graph G must be acyclic.

3.1 Deriving the Topic Graph from Wikipedia

We treat both articles and categories of Wikipedia as topics, with topic v_1 being the *parent* of v_2 whenever v_1 is listed among the categories of v_2. The whole procedure of building the topics graph consists of the following steps: *a*) mapping documents to Wikipedia articles, *b*) retrieving the parent categories, *c*) merging duplicate topics, *d*) breaking the cycles in the topic graph and *e*) extending the main topic.

Mapping the documents to Wikipedia is performed using *wikification* procedure. In essence, *wikification* is augmenting arbitrary texts with links to Wikipedia articles, in much the same way as Wikipedia articles are linked to each other. Among the various approaches to *wikification* that can be used at this step, we select one described in [26], as it is implemented in an open source tool Wikipedia Miner [27]. Using machine learning and statistics derived from Wikipedia pages, Wikipedia Miner decides which phrases in the text should be linked to Wikipedia, and which articles they should be linked to. We concatenate the texts of the documents into a single string and submit it to Wikipedia Miner for *wikification*. The returned string augmented with links to Wikipedia articles is then split back into documents, and each document is associated with the set of topics corresponding to the articles to which its text has been linked:

$$R := \{(v, d) | \text{ the wikified text of } d \text{ contains a link to } v\},$$

$$V := \{v | \exists d, (v, d) \in R\}.$$

Retrieving the parent categories is a step that allows us to establish relations between the topics, which in turn provides the main tool for generalization and summarization. For every article v obtained at the previous step, we augment the discovered set of topics V with all its parent categories:

$$V := V \cup parent\_categories(v),$$

$$E := E \cup \{(c, v) | c \in parent\_categories(v)\}.$$

As we want to avoid topics that are too general or to abstract, we perform the above step only once, that is we do not explicitly retrieve further ancestors of the parent categories. However, the following step typically introduces more distant hierarchical relations in our topic graph.

Merging duplicate topics. Some of the topics have both an associated article and a category in Wikipedia. In order for our topic graph to contain no redundant nodes, we merge such duplicate topics into one. In addition, we merge near-duplicate topic whose titles coincide up to *lemmatization*, such as, for instance, the article on *Decision tree* and the category *Decision trees*. As from the pair of topics being merged one topic is typically an article, and another is a category, the resulting topic will have both parent and child relations. As a result, during this step the formerly bipartite graph G transforms into a general directed graph containing paths of various lengths. After this step we erase the distinction between articles and categories and from now on treat them uniformly as topics.

Breaking the cycles. The category graph of Wikipedia contains occasional cycles[2], and does not thus form a perfect hierarchy. For the purpose of our method we detect and break the cycles in the topic graph using a depth-first search starting from the 'root' topics (those without parent topics).

Extending the main topic. Having experimented with academic search results for various queries, we noticed that main topic of the query often has no children in the topic graph. The reason for that is that the queries we are interested in are usually quite specific: for instance, we will more likely query for *statistical relational learning* than for *machine leaning*. Wikipedia is often sufficiently fine-grained to contain articles on such specific topics, but not entire categories. This results is these topics being present only as leafs in our topic graph. The processing we perform at this step allows us to amend this situation to some extent.

The idea is quite simple: we can detect the main topic of our document collection and link it to candidate child topics. For example, when querying for *statistical relational learning*, we would like to see *Markov logic networks* as a child of the corresponding main topic. For the purpose of detecting the main topic, selecting the topic that has the most associated documents has proven a good heuristic. When extending the main topic v_{main} with a child topic v we require the following properties to hold: *a*) v should be already present in the topic graph ($v \in V$), *b*) Wikipedia article about v_{main} should contain a link to the article about v, *c*) v should not be an ancestor of v_{main} in the topic graph. Interestingly, the child nodes v introduced in this way are often not the proper subtopics of v_{main}, but can be viewed as such in the context of the query (think, for instance of *Regularized trees*, *Stepwise regression* and *LASSO* in the context of *Feature selection*). The described heuristic procedure thus usually transforms the topic graph in a useful way, providing the main topic with an informative sub-structure.

4. SUMMARIZING THE GRAPH

The topic graph G and topic-document relation R built at the previous step contain useful information about the distribution of topics in our document collection D. However, at this point the graph is too large to be an informative visual representation of the documents: for a hundred of publication abstracts, the typical number of topics in G exceeds two hundred. At this step we select a subgraph of G to be used as a visual summary of the document collection. Given a limit T on the number of topics in the summary, our goal is to select the subgraph G_T that represents the collection of documents in the most informative way.

We do not intend to grasp the exact notion of 'informativeness', which may not be objectively definable. Instead we define properties that are favourable for good topic summaries and learn their correct proportions from examples. We elaborate on the properties in Section 4.3, while in the following section we describe problem of learning and predicting the summaries.

4.1 The Learning Problem

Viewed as a standard structured prediction problem, our goal is to learn a scoring function

$$F_w(G, R, G_T) = \langle w, \Psi(G, R, G_T) \rangle \tag{1}$$

that is maximized by good topic summaries, and then construct summaries for new graphs G by maximizing this function over the set of all possible subgraphs:

$$\hat{G}_T = \underset{|G_T|=T}{\operatorname{argmax}} F_w(G, R, G_T). \tag{2}$$

Computing the argmax is generally prohibitively expensive, as it requires evaluating the scoring function over $\binom{|V|}{T}$ subgraphs. We alleviate this problem by imposing an additional constraint that is natural for our settings. Specifically, we require that for a given input graph G the optimal topic summaries of different sizes should be nested:

$$\hat{G}_1 \subset \hat{G}_2 \subset ... \subset \hat{G}_T.$$

In other words, bigger summaries can be obtained from smaller ones by only adding new topics:

$$\hat{G}_t(\hat{V}_t, \hat{E}_t), \hat{G}_{t+1}(\hat{V}_{t+1}, \hat{E}_{t+1}) \Rightarrow \hat{V}_{t+1} = \hat{V}_t \cup \{v_{t+1}\}.$$

This requirement is justified by the principle of least surprise: when moving from less to more detailed summaries, the user will likely not expect the topics to disappear. Considering this requirement, the problem can be reformulated as predicting the sequence of topics $\hat{v}_1, \hat{v}_2, ..., \hat{v}_T$ whose prefixes constitute the nodes of intermediate summary graphs. Assuming that we have 'ground truth' examples of the form $((G, R), (v_1, ..., v_T))$, we can view this as an imitation learning problem, in which we want to copy the expert's behaviour in selecting the topics $(v_1, ..., v_T)$.

DAgger (Dataset Aggregation) framework [33] allows us to cast this problem into the problem of training a local policy that predicts the best next action (topic v_{t+1}) given the current state (initial input (G, R) plus the current summary $\hat{G}_t$). In essence, DAgger ensures that such a policy behaves well when applied to its own-generated, rather then optimal, states. The way this is accomplished is by iteratively retraining the policy on a updated training set. At each iteration the training set is augmented with examples $((G, R, \hat{G}_t), v^{opt}_{t+1})$, in which inputs $(G, R, \hat{G}_t)$ are produced by the current policy, and outputs v^{opt}_{t+1} are optimal actions provided by the expert.

Applying DAgger requires two ingredients:

[2]According to Wikipedia, *Artificial intelligence* is both a parent and a subcategory of *Cognitive science*

1. a policy $\pi : (G, R, \hat{G}_t) \mapsto \hat{v}_{t+1}$ that can be trained on examples $((G, R, \hat{G}_t), v_{t+1})$, and

2. an 'expert' $\pi^* : (G, R, v_1, v_2, ..., v_T, \hat{G}_t) \mapsto v^{opt}_{t+1}$ that can produce optimal actions v^{opt}_{t+1} given the true optimal sequence $v_1, v_2, ..., v_T$ and a current (non-optimal) state $(G, R, \hat{G}_t)$.

Providing the policy. In order to build the policy π, we need a classifier that can learn how to map an intermediate topic graph $(G, R, \hat{G}_t)$ to the best next topic $\hat{v}_{t+1}$. We view this as structured prediction problem similar to the formulations (1, 2). Specifically, during training we would like to learn a linear function

$$F_w(G, R, \hat{G}_t, v_{t+1}) = \langle w, \Psi(G, R, \hat{G}_t, v_{t+1}) \rangle \quad (3)$$

that is maximized by optimal decisions v_{t+1}. The prediction is then performed by maximizing the learned function for a given input $(G, R, \hat{G}_t)$ over the possible set of topics:

$$\hat{v}_{t+1} = \underset{v_{t+1} \notin \hat{G}_t}{\operatorname{argmax}} F_w(G, R, \hat{G}_t, v_{t+1}). \quad (4)$$

The important distinction from the formulations (1, 2) is that argmax is computed over the set of topics (rather then subgraphs), which is feasible. We solve this prediction problem by using SVM^{rank} instantiation of the SVM^{struct} software [21]. In order to compute the partial ranking $r_{G,R,\hat{G}_t}$ of different solutions $\hat{v}_{t+1}$ for a given input $(G, R, \hat{G}_t)$ we define a loss function between the sequences

$$\ell_{G,R}((v_1, v_2, ..., v_{t+1}), (v'_1, v'_2, ..., v'_{t+1})), \quad (5)$$

and compute it with respect to the optimal decision:

$$r_{G,R,\hat{G}_t}(\hat{v}_{t+1}) = -\ell_{G,R}((\hat{v}_1, \hat{v}_2, ..., \hat{v}_{t+1}), (\hat{v}_1, \hat{v}_2, ..., v_{t+1})).$$

In our experiments we defined $\ell_{G,R}$ to be a 0/1 loss, which corresponds to specifying no preferences between non-optimal decisions, which in turn results in fewer constraints and an easier problem for SVM^{rank}.

Providing the expert actions. At each iteration of DAGGER we need to compute the optimal actions v^{opt}_{t+1} for all states $(G, R, \hat{G}_t)$ generated by our current policy. This is accomplished by minimizing the loss with respect to the true optimal sequence:

$$v^{opt}_{t+1} = \underset{\hat{v}_{t+1}}{\operatorname{argmin}} \ell'_{G,R}((\hat{v}_1, \hat{v}_2, ..., \hat{v}_{t+1}), (v_1, v_2, ..., v_{t+1})).$$

In these settings 0/1 loss is inappropriate, as it gives equal score to all non-optimal sequences, rendering the minimization problem meaningless in most cases. An obvious candidate for $\ell'_{G,R}$ is Jaccard distance function. However, it turns out that Jaccard distance does not take into account the similarity between the topics: it tends to add topics from the optimal sequence, even when the non-optimal partial sequence already contains similar topics. In other words, it encourages redundancy in the built topic summaries.

We designed a matching-based loss function $\ell'_{G,R}$ that does not suffer from this problem:

$$\ell'_{G,R}(..., ...) = 1 - match(..., ...) \quad (6)$$

The matching score greedily assigns best-scoring candidate topics to the topics from the optimal sequence, starting from the first optimal topic v_1. The score of the assignment (v, v') is computed as Jaccard distance between the sets of documents transitively associated with the topics v and v', plus a constant α if $v = v'$. The final matching score is the average of the assignment scores divided by $1 + \alpha$.

4.2 Connecting topics and documents

The learning procedure described above allows us to sequentially select the topics $v_1, v_2, ..., v_T$ to be included into the topic summary G_T. In order to completely define the summary graph, we need to decide how to connect the topics with links. On one hand, we want to maintain the hierarchical relations between the topics in the graph, but on the other hand we do not want to clutter the topic summary with unnecessary links. The way we solve this problem is by introducing the *minimum* possible number of links that still maintain the hierarchical structure of the original topic graph: for every $v_i, v_j \in V_T$ such that v_i is an ancestor of v_j in the original graph G, v_i must be the ancestor of v_j in the topic summary G_T. Technically this amounts to computing the transitive closure $G^+(V, E^+)$ of the original graph, selecting the subgraph of G^+ containing the nodes $v_1, v_2, ..., v_T$ and computing the transitive reduction of the result.

It is important to mention how documents are assigned to the nodes of the summary graph. After the summary graph is built, we compute a new transitive topic-document relation R^+_T. A document d is assigned to the topic v whenever it was assigned to any of the descendant topics of v in the original graph:

$$R^+_T = \{(v, d) | v \in V_T, \exists v' \in V, (v, v') \in E^+, (v', d) \in R\}.$$

4.3 Features

An important step of the algorithm is computing the joint feature representation $\Psi(G, R, \hat{G}_t, v_{t+1})$ for the problem (3, 4). Based on the features, the policy π should be able to learn how to add topics v_{t+1} to intermediate summary graphs $\hat{G}_t$. The features we use measure various properties of the graph $\hat{G}_{t+1}(\hat{V}_{t+1}, \hat{E}_{t+1})$ that results from adding the topic v_{t+1} to the summary $\hat{G}_{t+1}$. We compute properties that describe the structure and the look of the resulting summary, as well as the topic-document relations it induces.

The main set of features is related to frequency and diversity of the topics $\hat{v}_i \in \hat{V}_{t+1}$ in the summary:

1. *document coverage:* $\left|\{d \in D | \exists v \in \hat{V}_{t+1}, (v, d) \in R\}\right|$
2. *transitive document coverage:* $\left|\{d \in D | \exists v \in \hat{V}_{t+1}, (v, d) \in R^+\}\right|$
3. average and minimum *topic frequency*, where: $topic\_freq(v) = |\{d \in D | (v, d) \in R\}|$
4. average and minimum *transitive frequency*, where: $trans\_topic\_freq(v) = \left|\{d \in D | (v, d) \in R^+\}\right|$
5. average and maximum topic *overlap*, where: $overlap(v_i, v_j) = \frac{\left|\{d \in D | (v_i, d) \in R^+ \wedge (v_j, d) \in R^+\}\right|}{\left|\{d \in D | (v_i, d) \in R^+ \vee (v_j, d) \in R^+\}\right|}$

6. average pairwise *distance* between the topics, where *distance* is the length of the shortest path through a common ancestor in the original graph

7. *partition coefficient* (measures how crisp are the topics viewed as fuzzy clusters of documents)
$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\{v\in\hat{V}_{t+1}|\,(v,d_i)\in R^+\}|}$

Another feature measures the *unevenness* of the sizes (transitive frequencies) of sibling topics in the summary. Finally, one feature is dedicated to measuring the percent of topics in the summary graph that are subtopics of the main topic (see "Extending the main topic", Section 3).

5. EVALUATION

We carried out the evaluation of the proposed method on the search results obtained from Microsoft Academic Search (MAS)[3] for 10 distinct queries. For each query we collected one hundred top results from MAS, discovered the topics in their titles and abstracts, and built the topic graph as described in Section 3. The topic graphs were then annotated with 'ground truth' topic sequences of length 8, corresponding to nested summaries of the topic graphs. The summaries were selected so as to represent the search results and the discovered topics in the most informative way according to our judgement.

The method was evaluated on the task of predicting the topic sequences using leave-one-out cross-validation on the described dataset. Two different performance metrics were used: precision@n and match@n. Precision@n measures the percent of correctly predicted topics in the subsequence of length n, taking into account only exact matches. Similarly, match@n measures the match score (6) between the subsequences of length n, thus allowing for partial matches between similar topics. For the sake of comparison we implemented a baseline `greedy coverage` algorithm, and adapted the spectral clustering-based method of Scaiella et al. [36].

5.1 Baseline GreedyCov algorithm.

An implemented baseline algorithm `GreedyCov` selects topics by greedily optimizing the document coverage. That is, at each step it selects the topic that covers most of the documents that are not covered by previously selected topics. We should note that this is a reasonable baseline: it optimizes both the frequency and the diversity of the selected topics, both properties being important for a good summary. Accordingly, the feature responsible for document coverage received one of the largest weights in our learning algorithm. Moreover, the first topic selected by this greedy baseline coincides with the 'main topic' in the collection, as it is approximated in our approach (as described in "Extending the main topic", Section 3). As during the labeling we always selected the true main topic first, the baseline approach selected the first topic correctly, whenever our approximation was correct. Our method can also be seen as greedily optimizing a linear combination of features, the difference being that the feature weights are learned from the training data.

[3]http://academic.research.microsoft.com/

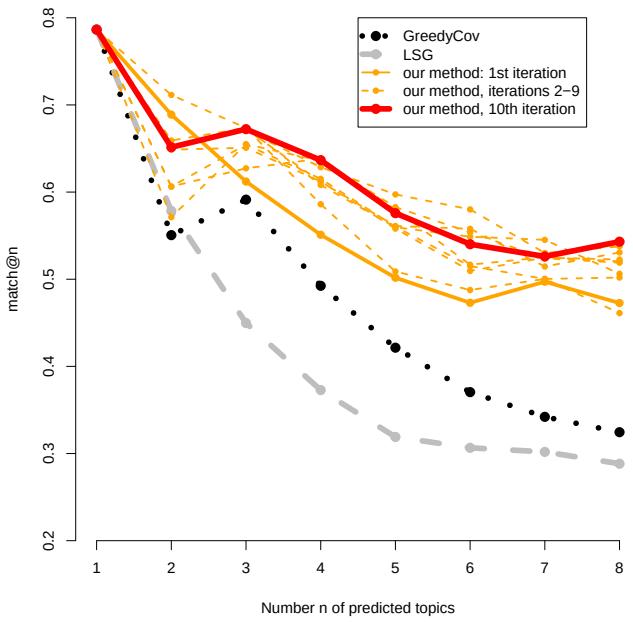

Figure 2: Match@n of predicted topics. Dotted black line corresponds to the baseline `GreedyCov` method, dashed grey line – to `LSC`. Solid orange and red lines correspond to our method at the 1st and the 10th iteration of DAgger respectively; thin dashed orange lines – to intermediate iterations.

5.2 Labeled spectral clustering.

Scaiella et al. [36] recently proposed a novel method for search result clustering based on Wikipedia. The method performs a particular form of spectral clustering on the graph of documents and topics, with subsequent selection of cluster labels. For convenience we will refer to this method as `LSC` (labeled spectral clustering). In the first step of `LSC` the documents (search result snippets) are annotated with links to Wikipedia articles using TAGME topic annotator [13]. For each document-topic link (d, v) TAGME provides an importance score $\rho(d, v)$, which is used as link weight in the resulting weighted bipartite document-topic graph. The graph is then augmented with between-topic links (v, v') that are weighted according to topic relatedness. The relatedness between topics $rel(v, v')$ is also computed by TAGME and is based on the incoming citations of the corresponding Wikipedia articles.

In the following graph pre-processing step the most significant topics are selected. The algorithm views topics as document sets, and selects significant topics by greedy solving a variation of the set cover problem that takes into account document-topic weights. In the classical set cover problem, sets fully "cover" their contained elements; at each step the greedy algorithm selects the set that covers the most elements not covered by the previously selected sets; and the algorithm proceeds until all elements are covered. In contrast, we consider topics as covering the documents only *partially*, with the degree of coverage being equal to the corresponding

topic-document link weights $\rho(d, v)$. In our implementation of LSC we maintain the set of documents $D = \{d\}$, each with the associated volume $vol(d)$ that remains to be covered. The volumes are initialized to one ($vol(d) := 1$); each selected topic v reduces the remained document volumes by the amount $min(\rho(d, v), vol(d))$; and the greedy algorithm selects topics that reduce the total volume the most until all documents are covered ($vol(d) = 0$). After this step the graph is restricted to only significant topics and documents associated with them.

In the original method of Scaiella et al., topics that cover more then fifty percent of the documents are removed from the graph prior to selecting significant topics. This step is justified in the task of plain clustering considered in [36], as overly frequent topics do not help discriminating between the documents. In our approach, the most frequent topics often corresponds to the main topics of the search query, and are arguably useful for our hierarchical topic representation. Therefore we omit this step in our implementation of LSC.

In the following step the topics are iteratively clustered based on the spectral properties of the graph of topics v and their relations $rel(v, v')$. At each iteration the algorithm selects one of the big clusters – those covering more than δ_{max} documents – and splits it in two. Informally, the algorithm ensures that the sparsest of the big clusters is selected, and that the split goes through the sparse region of the corresponding topic subgraph. As recommended in [36], in order to obtain T final clusters we build $T + m - 1$ clusters with the described algorithm and then merge m smallest clusters into one. Finally, each cluster is labeled with the topic that has is most strongly associated with the documents in the cluster, as measured by document-topic weights $\rho(d, v)$. We treat the produced cluster labels as a final output of LSC constituting the topical representation of the search results.

Overall there are the following main differences between LSC and our algorithm: *a*) LSC uses only Wikipedia articles to represent topics, while we use both articles and categories; *b*) LSC relies on similarity-based relatedness between topics, while we rely on hierarchical relations in the article-category network; *c*) in LSC topic aggregation is performed on the basis of clustering, while in our method – on the basis of topic `generalization` (based on the hierarchy); *d*) LSC selects topics through unsupervised procedure of labeled clustering, while we rely on supervised structured output prediction.

5.3 Details of the evaluation setup

For the ease of comparison, we used TAGME [13] as a topic annotator for all the three algorithms (thus running LSC, the method of Scaiella et al. [36], in the original settings). When evaluating the results produced by LSC, we first embedded them into the topic graph built as described in Section 3, in order to correctly match the results to 'ground truth' topics, in particular to capture the matches between *similar topics*.

The employed evaluation metrics precition@n and match@n assess the sequences of summary graphs of increasing sizes $t \in 1, 2, ..., T$, as they are produced by our method. In order to evaluate the method of Scaiella et al. on the same basis, we ran LSC varying the parameter values, and for each t selected the best average result across folds according to the metric in question. For simplicity, δ_{max} was fixed at the value of 3 which is arguably the smallest number of documents we would like to see in the cluster. We should note that changing the value of δ_{max} did not notably affect the results, which confirms the robustness of the method reported by Scaiella et al. The value of m – the number of the smallest clusters to be merged – was ranged from 1 to 5, which in our experiment corresponded to producing from 8 to 12 clusters prior to merging. The performance of our method was taken at the iterations of DAgger from one to ten.

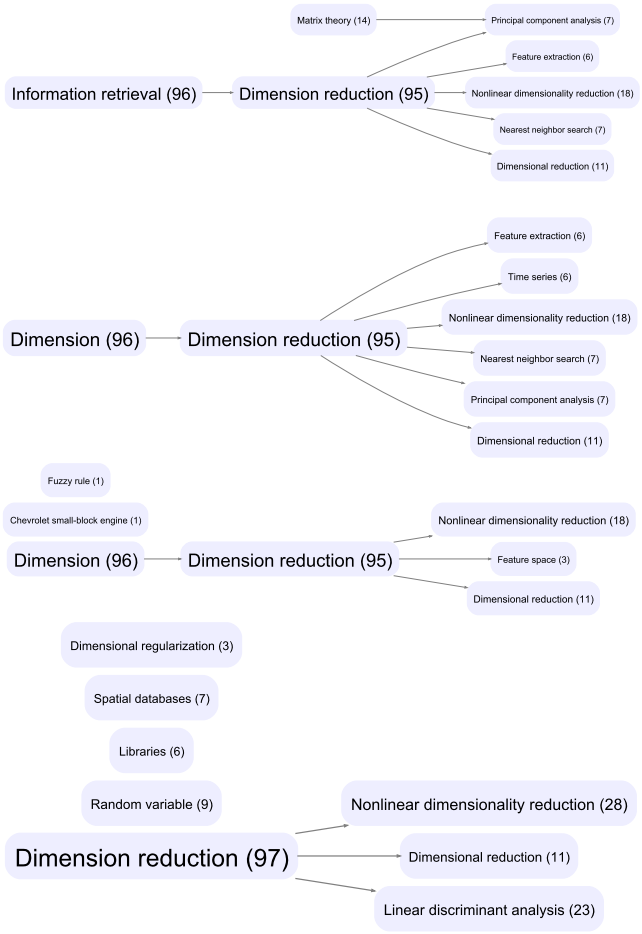

Figure 3: Topic summaries for search results on *dimensionality reduction* produced by (from the top): manual labeling, our method, GreedyCov and LSC.

5.4 Evaluation results

Figure 2 shows the performance of the three evaluated methods in terms of match@n. The figure for precision@n is omitted for space reasons (the metric exhibits similar behaviour). The first iteration is equivalent to not using DAgger, which corresponds to training only on the states encountered in the ground truth labeling. As we can see from the figure, at $n = 1$ the curves coincide, as for each of the 10 queries the methods happen to agree on the first predicted topic. As the number of topics increases, the curves begin to diverge, with growing advantage of our method over the other two. The advantage over LSC (measured as difference between the

scores) becomes statistically significant with p-value of 0.05 starting from $n = 3$, and over `GreedyCov` – starting from $n = 6$. The difference between the performance scores of `GreedyCov` and `LSC` is not significant for any n. We should note that the performance increase of our method after the first iteration justifies the procedures of dataset aggregation.

We can see that `GreedyCov` performs reasonably well for small n, which indicates that document coverage is an important characteristic for summary topic graphs of small sizes. As more topics are added to the graph, the performance of `GreedyCov` notably deteriorates. At the point when most of the documents are covered by previously selected topics, the greedy coverage strategy becomes suboptimal, as it starts to prefer "outlying" topic nodes. The spectral clustering-based `LSC` method encourages regular topic sizes both in terms of contained Wikipedia articles and documents. The drawback of `LSC` in the context of our task is that it is designed for plain rather than hierarchical clustering. In general, we can conclude that, being unsupervised methods, `LSC` and `GreedyCov` encode some of the useful properties of the topic summaries. However, as they are not specifically tailored for producing hierarchical summaries of various sizes, the captured properties are not sufficient for building the sequences of informative summary graphs. In these settings our supervised method has an advantage, as it is able to learn how to combine multiple properties in order to build high-quality summary sequences. Figure 3 shows an example of the "ground truth" topic summary for the query *dimensionality reduction* along with the summaries produced by the evaluated methods.

6. PROTOTYPE IMPLEMENTATION

We implemented our method of building the topic summaries into a prototype Web tool `S*******n` for academic search. The tool relies on the API[4] of `Microsoft Academic Search` for obtaining the query results and groups the results into topics according to our method. `S*******n` shows a familiar user interface for typing in queries and displaying results, and an additional browsing control – the topic summary (see Figure 4). The nodes of the topic summary serve as interactive filters restricting the displayed results to the selected node and its subtopics. The size of a node depends on the number of results it contains. The number of nodes in the displayed summary can be adjusted using a slider control. We deployed `S*******n` and made it publicly available at `http://s*******n.*************/`. The tool demonstrates the applicability of our method to online processing of academic search results and building browsing interfaces on top of existing scholarly services.

7. CONCLUSION

We introduced a method for grouping academic search results according to Wikipedia categories and article pages. The method produces concise and structured topical summaries useful for visualization and browsing. The topics in the summaries are fine-grained, meaningfully labeled and up to date due to the nature of Wikipedia. We developed a novel algorithm based on structured prediction that learns how to produce informative topic summaries from a small number of examples. The proposed method relies on the publicly available data, and can be implemented on top of existing academic search services.

[4]http://academic.research.microsoft.com/About/Help.htm#4

References

[1] D. Blei, T. Griffiths, and M. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57(2):7, 2010.

[2] D. Blei and J. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

[3] D. Blei and J. Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

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

[5] L. Bolelli, Ş. Ertekin, and C. Giles. Topic and trend detection in text collections using latent dirichlet allocation. *Advances in Information Retrieval*, pages 776–780, 2009.

[6] C. Calli, G. Ucoluk, and T. Sehitoglu. *Improving search result clustering by integrating semantic information from Wikipedia*. PhD thesis, MS Thesis, Middle East Technical University, Department of Computer Engineering, 2010.

[7] C. Carpineto, S. Osiński, G. Romano, and D. Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17, 2009.

[8] C. Carpineto and G. Romano. Exploiting the potential of concept lattices for information retrieval with credo. *Journal of universal computer science*, 10(8):985–1013, 2004.

[9] A. Csomai and R. Mihalcea. Linking documents to encyclopedic knowledge. *Intelligent Systems, IEEE*, 23(5):34–41, 2008.

[10] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992.

[11] L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *Proceedings of the 24th international conference on Machine learning*, pages 233–240. ACM, 2007.

[12] S. Dumais, G. Furnas, T. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285. ACM, 1988.

[13] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

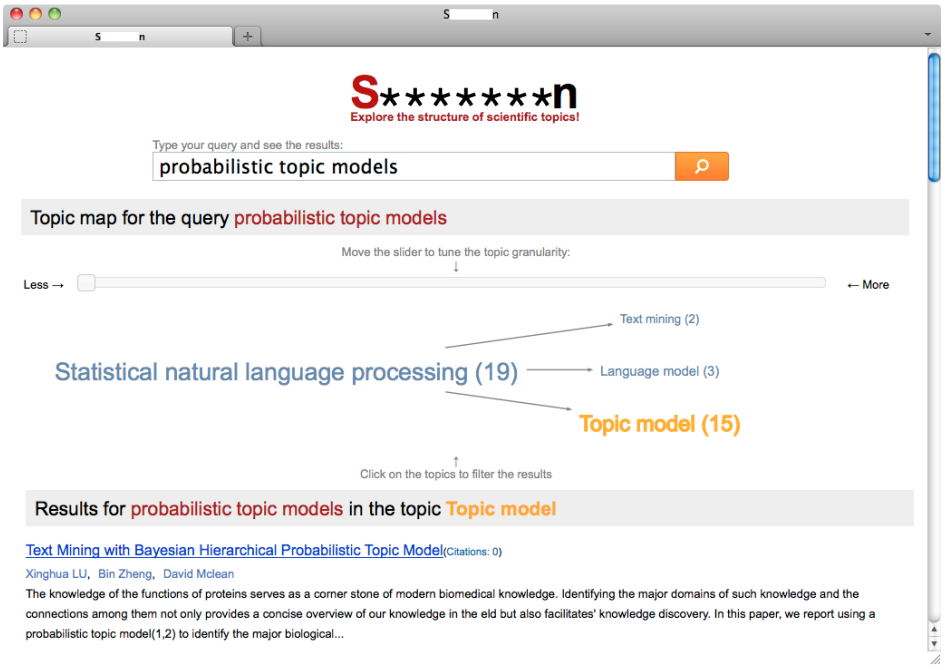

Figure 4: The interface of S*****n.**

[14] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1606–1611, 2007.

[15] T. 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.

[16] X. Han and J. Zhao. Topic-driven web search result organization by leveraging wikipedia semantic knowledge. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1749–1752. ACM, 2010.

[17] Q. He, B. Chen, J. Pei, B. Qiu, P. Mitra, and L. Giles. Detecting topic evolution in scientific literature: how can citations help? In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 957–966. ACM, 2009.

[18] M. Hearst and J. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84. ACM, 1996.

[19] P. Jiang, C. Zhang, G. Guo, Z. Niu, and D. Gao. A k-means approach based on concept hierarchical tree for search results clustering. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 1, pages 380–386. IEEE, 2009.

[20] S. Jinarat, C. Haruechaiyasak, and A. Rungsawang. Improving web search result categorization using knowledge from web taxonomy. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 2, pages 726–730. IEEE, 2009.

[21] T. Joachims. Training linear svms in linear time. In *SIGKDD*, pages 217–226. ACM, 2006.

[22] N. Kawamae. Author interest topic model. In *SIGIR*, pages 887–888. ACM, 2010.

[23] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[24] Y. Maarek, R. Fagin, I. Ben-Shaul, and D. Pelleg. Ephemeral document clustering for web applications. 2000.

[25] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8. ACM, 2011.

[26] D. Milne and I. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

[27] D. Milne and I. H. Witten. An open-source toolkit for mining wikipedia. *Artificial Intelligence*, 194(0):222 – 239, 2013.

[28] R. Nallapati, A. Ahmed, E. Xing, and W. Cohen. Joint latent topic models for text and citations. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550. ACM, 2008.

[29] C. Nguyen, X. Phan, S. Horiguchi, T. Nguyen, and Q. Ha. Web search clustering and labeling with hidden topics. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(3):12, 2009.

[30] S. Osiriski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent information processing and web mining: proceedings of the International IIS: IIPWMâĂŹ04 Conference held in Zakopane, Poland*, page 359, 2004.

[31] A. Popescul and L. Ungar. Automatic labeling of document clusters. *Unpublished manuscript, available at http://citeseer. nj. nec. com/popescul00automatic. html*, 2000.

[32] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press, 2004.

[33] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15:627–635, 2011.

[34] C. Sacarea, R. Meza, and M. Cimpoi. Improving conceptual search results reorganization using term-concept mappings retrieved from wikipedia. In *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, volume 3, pages 234–238. IEEE, 2008.

[35] A. Sameh and A. Kadray. Semantic web search results clustering using lingo and wordnet. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, 1(2), 2010.

[36] U. Scaiella, P. Ferragina, A. Marino, and M. Ciaramita. Topical clustering of search results. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 223–232. ACM, 2012.

[37] B. Stein and S. Zu Eissen. Topic identification: Framework and application. In *Proc. International Conference on Knowledge Management*, volume 400, pages 522–531, 2004.

[38] J. Tang, R. Jin, and J. Zhang. A topic modeling approach and its integration into the random walk framework for academic search. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 1055–1060. IEEE, 2008.

[39] P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *Proceedings of the 2006 international conference on Digital government research*, pages 167–176. Digital Government Society of North America, 2006.

[40] M. Wahabzada, Z. Xu, and K. Kersting. Topic models conditioned on relations. *Machine Learning and Knowledge Discovery in Databases*, pages 402–417, 2010.

[41] D. Weiss. *Descriptive clustering as a method for exploring text collections*. PhD thesis, Citeseer, 2006.

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