Learning to Grow Structured Visual Summaries for Document Collections

Abstract

In this paper we propose a method of summarizing collections of documents with concise topic hierarchies, and show how it can be applied to visualization and browsing of academic search results. The proposed method consists of two steps: building the graph of topics relevant to the collection, and selecting the optimal subgraph thereof. In the first step, we map documents to a universal topic hierarchy and extract a graph of relevant topics. In the second step, we sequentially build summaries of the extracted topic graph using a structured output prediction approach. We describe how to build topic graphs based on the network of articles and categories of Wikipedia, and show how graph summarization can in our settings be cast into sequential prediction problem using DAgger (Dataset Aggregation) framework. The initial experimental result suggest that our method is able to learn how to grow good topic summaries from a small number of examples.

1. Introduction

When searching or browsing a collection of documents, we often want to perceive it as a whole, identify the constituent topics, understand their span and interrelations. A familiar scenario is scanning through academic search results when exploring an new research domain. Whether to better understand the domain, focus on a relevant subset of the result, or refine our search queries – we would like to grasp the topical structure of the returned collection without having to look through every item. In this paper we propose summarizing academic search results (and document collections in general) with topic maps – concise graphs of topics connected by hierarchical relations. We pose a problem of building a topic map of given size that represents the collection in the most informative way.

The problem can be split into two subproblems: discovering the set of interrelated topics in the documents, and finding the most informative subset thereof. The first problem can be addressed by linking the documents to a predefined hierarchy of topics. In this work we rely on the article-category network of Wikipedia, which has considerable coverage and level of detail in a broad range of subjects. We map documents to Wikipedia topics (articles and categories) using Wikipedia Miner (Milne & Witten, 2013), building the graph of topics linked with hierarchical relations.

Selecting the topic subgraph that provides the most informative representation of 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. 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 of good (informative) topic summaries, we learn how to grow such summaries from given examples. We view this as a sequential prediction problem, and apply DAgger meta-algorithm (Ross et al., 2011) to ensure that the method behaves well given its own-generated intermediate summaries. We demonstrate that our method is able to learn from a small number of examples, and produce informative topic summaries for unseen document collections.

2. Related Work

There has been a vast amount of research dedicated to finding topics in document collections and document clustering. General-purpose clustering methods (Weiss, 2006), concept lattices (Carpineto & Romano, 2004), and probabilistic topic models (Nguyen et al., 2009) are among the methods that have been applied to documents in general, as well as in the context of Web search. The problem of choosing the meaningful cluster labels has always remained challenging.

Large-scale manually-built knowledge sources, such as

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Wikipedia, has provided us with vast amount of information that has been carefully produced, interlinked and labeled. Gabrilovich and Markovitch (Gabrilovich & Markovitch, 2007) proposed representing texts as weighted combination of concepts based on Wikipedia articles for the purpose of computing semantic relatedness between texts. Han and Zhao (Han & Zhao, 2010) proposed grouping the search results according to topics defined as communities in the graph of semantic relatedness between Wikipedia-derived concepts. Similarly to our work, Scaiella et al. (Scaiella et al., 2012) 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. The main distinction of our work is that, 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

As an input to our method 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.

The way we build a topic graph is by mapping documents to the nodes of an existing topic hierarchy. In this work we derive the topic hierarchy from Wikipedia, although we suggest that other sources, such as domain-specific categorization schemes, could be used as well. The important advantages of Wikipedia are its broad topic coverage and that it is being constantly updated. We will now describe the steps required to extract useful and valid topic graphs from Wikipedia.

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* we chose one described in (Milne & Witten, 2013) as it is implemented in an open source tool Wikipedia Miner. 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 os 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[1], 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 containing no 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 but not *detailed* 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 (Ross et al., 2011) 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

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

(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:

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

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

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:

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

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 (Joachims, 2006). 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

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

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:

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

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 the Jaccard distance between the sets of documents transitively associated with the topics v and v', plus a constant α if the topics are the same. 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 sum-

mary 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 $\hat{G}_{t+1}$.

The first 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: $trans\_topic\_freq(v) = |\{d \in D | (v,d) \in R^+\}|$
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 and maximum parent-child *overlap*, with *overlap* defined above
7. average pairwise *distance* between the topics, where *distance* is the length of the shortest path through a common ancestor in the original graph
8. *partition coefficient* (measures how crisp are the topics viewed as fuzzy clusters of documents) $\frac{1}{N}\sum_{i=1}^{N} \frac{1}{\left|\{v \in \hat{V}_{t+1} | (v,d_i) \in R^+\}\right|}$

Another set of features describes the properties of the topic summary as a graph. These features include *a*) the number of connected components, *b*) the number of links, *c*) the height of the graph, *d*) the average number of children (for topics having children), and *e*) the average and the maximum number of parents (for topics having parents).

Some features communicate some specific properties that do not fall into the described categories. Thus, we have a feature that measures the *unevenness* of the sizes (transitive frequencies) of sibling topics in the summary. Another feature is used to measure the 'subtopic coverage', that is the average ratio between the number of subtopics in the original graph and the number of subtopics in the summary (for topics that have subtopics in the summary graph). 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. Initial Evaluation

We carried out the initial evaluation of the proposed method on the search results obtained from Microsoft Academic Search (MAS)[2] 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. The second metric, match@n allows for partial matches between similar topics by measuring the match score (6) between the subsequences of length n.

As a baseline we implemented an algorithm that selects topics by greedily optimizing the document coverage. 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

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

our approximation was correct. Our method can also be seen as greedily optimizing the linear combination of features, the main difference being that the feature weights are learned from the training data.

Figures 1 and 2 show the performance of our method at different iterations. We can see that few iterations already give improvement over the baseline. This suggests that the method is able to capture some general properties that are important for good topic summaries. The first iteration is equivalent to not using DAGGER, which corresponds to training only on the states encountered in the ground truth labeling. Improvement after the first iteration justifies the procedures of dataset aggregation.

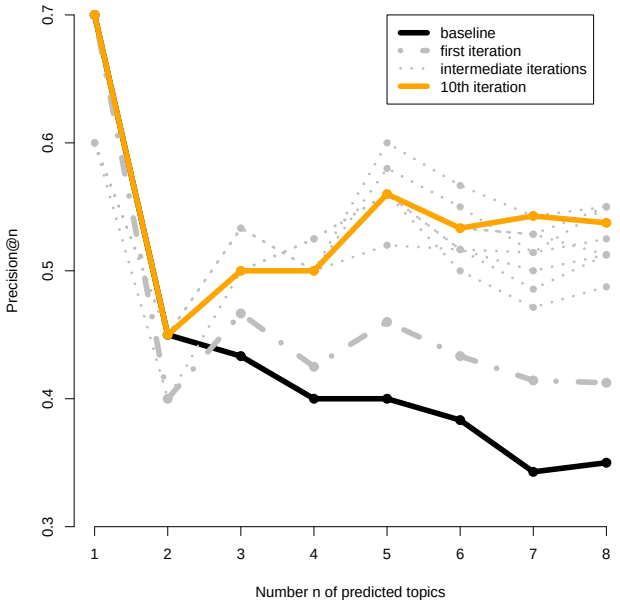

Figure 1. Precision@n of predicted topics. Solid black line corresponds to the baseline "greedy coverage" method, others – to our method at various iterations of DAGGER. Thick grey dash-dotted line and thick solid orange line correspond to the first and the last, tenth, iterations respectively.

References

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

Gabrilovich, Evgeniy and Markovitch, Shaul. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, volume 6, pp. 12, 2007.

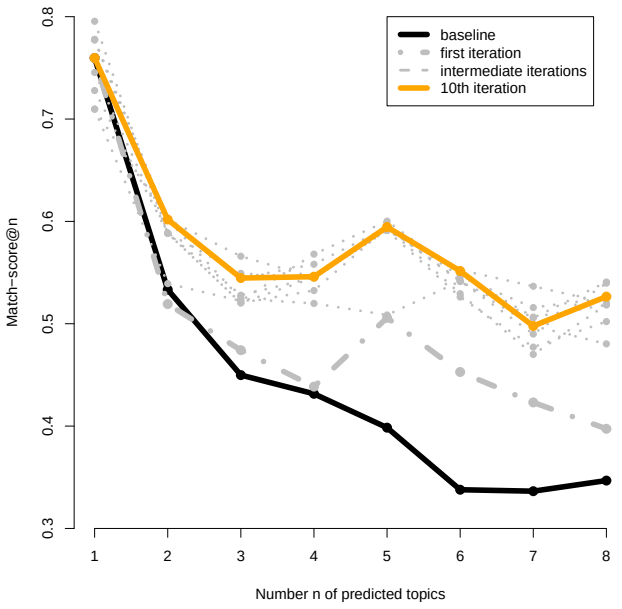

Figure 2. Match@n of predicted topics. Solid black line corresponds to the baseline "greedy coverage" method, others – to our method at various iterations of DAGGER. Thick grey dash-dotted line and thick solid orange line correspond to the first and the last, tenth, iterations respectively.

Han, Xianpei and Zhao, Jun. Topic-driven web search result organization by leveraging wikipedia semantic knowledge. In *CIKM*, pp. 1749–1752. ACM, 2010.

Joachims, Thorsten. Training linear svms in linear time. In *SIGKDD*, pp. 217–226. ACM, 2006.

Milne, David and Witten, Ian H. An open-source toolkit for mining wikipedia. *Artificial Intelligence*, 194(0):222 – 239, 2013.

Nguyen, Cam-Tu, Phan, Xuan-Hieu, Horiguchi, Susumu, Nguyen, Thu-Trang, and Ha, Quang-Thuy. Web search clustering and labeling with hidden topics. *TALIP*, 8(3):12, 2009.

Ross, Stéphane, Gordon, Geoffrey J., and Bagnell, Drew. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15:627–635, 2011.

Scaiella, Ugo, Ferragina, Paolo, Marino, Andrea, and Ciaramita, Massimiliano. Topical clustering of search results. In *WSDM*, pp. 223–232. ACM, 2012.

Weiss, Dawid. *Descriptive clustering as a method for exploring text collections*. PhD thesis, Citeseer, 2006.