To appear in 2011 International Joint Conference of Artificial Intelligence (IJCAI 2011), July 16-22, 2011.

A New Search Engine Integrating Hierarchical Browsing and Keyword Search

Da Kuang and **Xiao Li** and **Charles X. Ling***
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{dkuang, xli485, cling}@csd.uwo.ca

Abstract

The original Yahoo! search engine consists of manually organized topic hierarchy of webpages for easy browsing. Modern search engines (such as Google and Bing), on the other hand, return a flat list of webpages based on keywords. It would be ideal if hierarchical browsing and keyword search can be seamlessly combined. The main difficulty in doing so is to automatically (i.e., not manually) classify and rank a massive number of webpages into various hierarchies (such as topics, media types, regions of the world). In this paper we report our attempt towards building this integrated search engine, called SEE (**S**earch **E**ngine with hi**E**rarchy). We implement a hierarchical classification system based on Support Vector Machines, and embed it in SEE. We also design a novel user interface that allows users to dynamically adjust their desire for a higher accuracy vs. more results in any (sub)category of the hierarchy. Though our current search engine is still small (indexing about 1.2 million webpages), the results, including a small user study, have shown a great promise for integrating such techniques in the next-generation search engine.

1 Introduction

The original Yahoo! search engine consists of a manually organized topic hierarchy (directory) of webpages for easy browsing. For example, webpages related to Olympic Games are grouped under the topic hierarchy of Recreation→Sports→Olympic Games. People who want to find information about the 1996 Atlanta Olympic Games can easily browse the related webpages under this subcategory.

However, a web directory is rarely provided in modern search engines, such as Google and Bing. This is because manual classification can hardly keep pace with the explosive growth of webpages. The modern search engines usually return a flat list of ranked webpages, based on user keywords.[1] As queries are usually related to topics [Broder, 2002], simple keyword queries are often insufficient as there is no easy way to express such topics as keywords [Cronen-Townsend and Croft, 2002]. Some works have been done to organize the returned pages into clusters [Zeng *et al.*, 2004; Ferragina and Gulli, 2005], but the clusters formed may not be meaningful or consistent to users' intention [Brenes *et al.*, 2009]. In addition, the clustering is only based on the limited number of returned pages (that is, not on all indexed webpages).

It would be ideal if hierarchical browsing and keyword search can be seamlessly combined. This hybrid search engine allows users to search by keywords as in Google or Bing while choosing (i.e., drilling down) any (sub)category in the hierarchy to obtain the ranked results in that category. If users do not choose any subcategory, it would be the same as the current Google or Bing. On the other hand, users can also browse any (sub)category without keywords, and the search engine would return a list of the most popular pages (e.g., the pages with the highest PageRank values [Page *et al.*, 1998]) within that category. Following the same logic, the front page of the search engine (such as our search engine) will list globally the top ten highest ranked webpages. This helps people to find out what is the "most popular and interesting" at any time.

In fact, most e-commerce sites (such as Amazon, e-Bay) have already implemented such a function (often called hierarchical multi-faceted search [Hearst, 2006]). The main difficulty in doing so in a general search engine is to automatically classify and rank a massive number of webpages into various hierarchies (such as topics, media types, regions of the world). Previous works on hierarchical text classification have been done, such as hierarchical SVM classifiers [Sun and Lim, 2001] and HMC Tree [Vens *et al.*, 2008]. However, few previous works attempt to integrate the hierarchical predictions with the query relevancy and page importance into a hybrid search engine.

In this paper, we build a prototype of a hybrid search engine, called SEE (**S**earch **E**ngine with hi**E**rarchy, see Figure 1).[2] SEE seamlessly integrates hierarchical browsing with keyword search. We implement an efficient hierarchical classification system based on Support Vector Machines (SVMs). We also propose a new ranking strategy that integrates SVM

*The contact author.

[1] Very recently, both Google and Bing offer a simple, one-level category on the left-hand side of the ranked pages.

[2] http://kdd.csd.uwo.ca:88/see

predictions. In addition, we design a novel user interface that allows users to dynamically adjust their desire for a higher accuracy vs. more results in any (sub)category they choose. Though SEE is still small (indexing about 1.2 million webpages), its scalability is feasible. To confirm the usability of such a hybrid search engine, we conduct an anonymous survey from a small group of undergraduate students to compare the flat and hierarchical versions of SEE.[3] We receive positive feedback from most students.

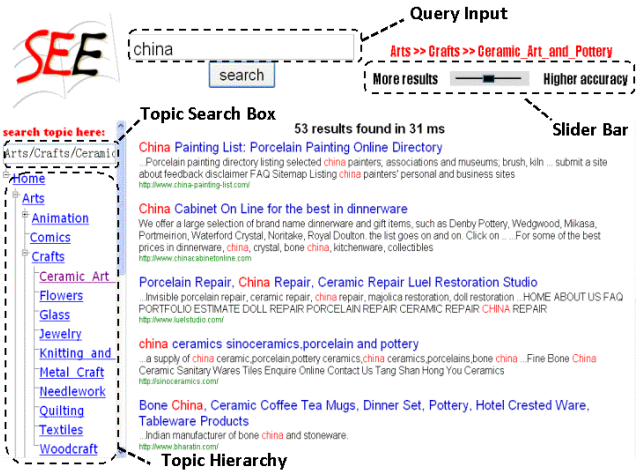

Figure 1: User interface of SEE. The topic search box on the left side and the slider bar on the top-right corner will be explained late.

There are at least two situations where our search engines can help users to find the desired results more easily than the flat one. The first common situation is when keywords are ambiguous. For example, if users want to find information about "china" (pieces of art), they could search keyword "china" in Google. Due to ambiguity of the word "china", most of the top ten returned pages are related to the country "China". If we combine more keywords (such as "ceramic"), although most of the top ten results become relevant, pages about china without the word "ceramic" will now be filtered out, and thus, users may miss some important webpages. However, if users select the category "ceramic" in the topic hierarchy while searching "china" as the keyword, all the results will be exclusively related to china (pieces of art) and the results without the word "ceramic" will still remain. Figure 1 shows the result of such search with SEE. We can see the top five results are all related to "china" and the top two results do not contain the word "ceramic".

The second situation is where users may not know what keywords to use, as they may not be familiar with the domain. In this case, they can select an appropriate category, and search with other keywords in that category. Section 4.3 shows an example of this case.

[3] As currently SEE only indexes 1.2 million webpages, it cannot be compared directly with Google and Bing. We compare the flat and hierarchical versions of SEE in Section 4.3.

2 Architecture of SEE

Figure 2 shows the architecture of SEE which consists of three loosely-coupled subsystems. The first subsystem, the webpage data server, collects the raw HTML pages from the web and archives the text data (i.e., removes HTML tags) for prediction. These text data will be indexed into the search engine's database. The second subsystem is the hierarchical classification system. It trains SVM classifiers on labeled text data, generates topic predictions and stores them into the topic prediction database. This is done off-line on the whole set of the indexed webpages with efficient hierarchical classification algorithms. The third subsystem, the hierarchical search engine, integrates ranking, the first two subsystems, and the user interface together. For each user query (with possible subcategory selection), the search engine combines the topic predictions, query relevancy and page importance, and returns the ranked pages within that subcategory. The user interface of SEE is shown in Figure 1. In the rest of this paper, we mainly focus on the second and third subsystems where we contribute the most.

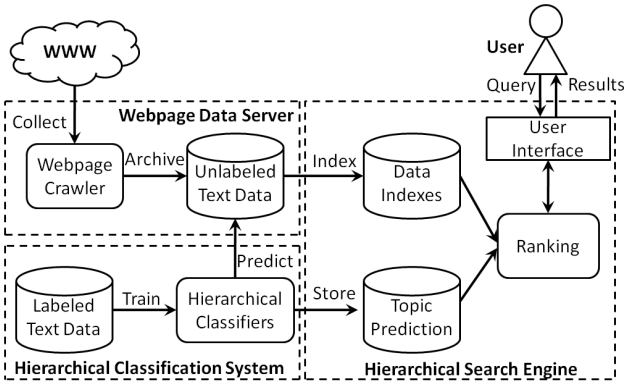

Figure 2: Architecture of SEE.

3 Hierarchical Classification

In this section, we describe our hierarchical classification method, experiment configuration and results.

3.1 Classification Method

The previous works of hierarchical classification can be categorized into the big-bang and the top-down approaches [Silla and Freitas, 2011]. In the big-bang approach, such as HMC Tree [Vens *et al.*, 2008], a single classifier is trained to predict the entire hierarchy. However, it is difficult to build a single classifier for a very large hierarchy [Xue *et al.*, 2008]. In the top-down approach, such as hierarchical SVM classifiers [Sun and Lim, 2001], multiple classifiers are trained to predict each subcategory separately. It was shown that the top-down approach is superior to the big-bang approach [Liu *et al.*, 2005; Xue *et al.*, 2008].

In this paper, we adopt the top-down approach for training. We build one binary classifier on each category of the predefined topic hierarchy. The positive and negative examples for each category are selected locally [Silla and Freitas,

2011]. Formally, for category c, let $\uparrow(c)$ denote the direct parent category of c, and $E(c)$ denote the examples belonging to c. Then the positive examples of c can be defined as $Tr_+(c) = E(c)$ and the negative training examples of c can be defined as $Tr_-(c) = Tr_+(\uparrow(c)) - Tr_+(c)$. As the negative examples are only selected from the positive examples of the parent category, the classifier would not suffer from serious imbalanced problem [Japkowicz and Stephen, 2002].

We use linear SVM as the base classifier. The LIBLINEAR [Fan *et al.*, 2008] package is used in our implementation. The Platt's calibration [Platt, 1999] is implemented to output calibrated probability. We use the BNS feature selection algorithm which has been shown to outperform most feature filtering methods [Forman, 2003]. To train SVM classifiers on large-scale datasets, we develop a parallel computing system with a small cluster of PCs.[4]

3.2 Experiments

Dataset

We use DMOZ (Open Directory Project) as our experimental data. DMOZ has a predefined topic hierarchy with a maximal depth of 14. As predicting the topic hierarchy is most challenging for hierarchical classification, we adopt the DMOZ topic hierarchy in our search engine. A meaningful, *4-level* topic hierarchy is extracted from the original hierarchy of DMOZ, since it is difficult to train highly accurate classifiers at the deep levels of the hierarchy [Liu *et al.*, 2005]. Besides, a very deep hierarchy is not convenient for users to browse. We select 11 out of the total 16 top categories in the DMOZ taxonomy. They are *Arts*, *Business*, *Computers*, *Health*, *Home*, *Recreation*, *Reference*, *Science*, *Shopping*, *Society* and *Sports*. Under these categories, we choose 651 meaningful subcategories within the top four levels as our topic hierarchy. After the data collection and cleaning, we obtain 1,204,244 webpages.[5]

[Xue *et al.*, 2008] reported that 93% documents in DMOZ belong to only one path (one deepest category) in the hierarchy. However, it does not consider the category alias. In fact, a lot of webpages in DMOZ belong to multiply paths [Qi and Davison, 2009]. Thus, we create a multi-path (multi-label) dataset according to the rules of category alias in DMOZ's RDF[6]. Here, we report the statistic information of our topic hierarchy at each level in Table 1.

We can see that the average number of documents and the average number of distinct words (features) vary dramatically at each level. In some previous works [Liu *et al.*, 2005; Xue *et al.*, 2008], a fixed and relatively small number (less than 5,000) of features was chosen for each category. As the number of words varies dramatically for different categories, for each category, we choose the top 25% features ranked by the BNS method [Forman, 2003] for SVM classification[7], as SVM learns more accurately with a large number of relevant features [Joachims, 1998].

Level	c	$\bar{r}$	$\bar{n}$	$\bar{f}$
1	11	0.2156	1,204,244	1,061,118
2	99	0.1137	31,989	198,184
3	499	0.1053	3,941	51,203
4	53	0.1030	2,650	53,581

Table 1: The statistic information of the topic hierarchy at each level. c is the total number of categories. $\bar{r}$ is the average positive class ratio. $\bar{n}$ is the average number of examples. $\bar{f}$ is the average number of distinct words.

Performance Metrics

Previous works in hierarchical classification often use Macro and Micro F1 measure [Yang, 1999] derived from binary classification measures where hard classification thresholds must be set. However, in SEE, it is difficult to find the optimal thresholds for all classifiers in the hierarchy for all users. Therefore, we allow users to dynamically set the classification threshold during search for the desired results (see Section 4.2). Thus, AUC [Bradley, 1997] (Area under the ROC curve) is a better choice to evaluate our classification results as it does not rely on classification thresholds.

To accurately reflect the probability of category prediction at a deep level, we adjust the probability prediction at each category as follows:

$$P_{adj}(c) = P(c) \times \prod_{c' \in \Uparrow c} P(c') \quad (1)$$

where $\Uparrow c$ denotes the set of all the ancestor categories of category c, and $P(c)$ is the likelihood probability produced by the classifier on category c. The adjusted probability prediction is not only based on the probability of the current category but also the probabilities of all its ancestor categories. Thus the hierarchical information is not lost. Using the adjusted probabilities, AUC can be calculated for each category in the hierarchy, which will be discussed in the next subsection.

Evaluation

We conduct a three-fold cross validation to evaluate our classification system. At each round, two folds are used for training and the rest is used for testing. The probability predictions of each testing fold will be used later in our search engine.

For the test fold, the average AUC scores at each level of the hierarchy are tabulated in Table 2. We can see that the average AUC at the first two levels is relatively high (close to 0.9). Although the lowest AUC at the fourth level is less than 0.5, the average AUC at the fourth level is still above 0.7 with a small variance. It means that our classification system can achieve a satisfactory performance.

The training of the hierarchical classifiers is quite efficient. The average training time at each round of cross validation is 1,189 seconds. The average testing time on one fold (401,415 examples) is 105 seconds.[8] The throughput (number of documents classified per second) of our classification system is

[4] It consists of four quad-core PCs each with four 2.4 GHz CPU cores and 3GB memory.

[5] Although we only choose four-level categories, the webpages in deep levels of the DMOZ hierarchy are popped up and collected in the categories of the fourth level.

[6] http://rdf.dmoz.org/

[7] After feature selection, the category with the fewest words still has 11,000 features.

[8] The time of disk I/O and network I/O is not counted.

Level	Average	Std. Dev.	Min.	Max.
1	0.889	0.033	0.843	0.954
2	0.889	0.056	0.728	0.972
3	0.860	0.076	0.550	0.998
4	0.723	0.084	0.490	0.866

Table 2: The AUC score at each level.

3,823 (401415/105), which is quite good considering only a small cluster of PCs is used. Since both the training and testing processes can be performed off-line, we believe that the scalability of the hierarchical classification for a large search engine is feasible.

4 Hierarchical Search Engine

In this section, we present how we integrate the probabilities of each webpage predicted by SVMs into our new ranking algorithm and user interface. We also compare the flat and hierarchical versions of SEE.

4.1 Ranking Algorithm

To our best knowledge, most flat search engines primarily rely on query relevancy and page importance (e.g., PageRank [Page *et al.*, 1998]) to rank the returned results. However, in our search engine, we also need to consider accuracy of category prediction when ranking the returned results within each category. Thus, a good ranking algorithm combining query relevancy, page importance, and category prediction, is crucial.

In SEE, we consider three different scores for the final ranking, which are described briefly below. The *query score* reflects how relevant a given webpage is to a user's query. In general, the more times the query terms appear in a webpage, the more relevant that webpage is to the query. The second score is the *importance score*, which determines the importance and the popularity of a webpage (e.g., PageRank). The last score is the *category score* which reflects how likely a given webpage belongs to a category. Here, we directly use the adjusted probability (see Formula 1 in Section 3.2) to represent the category score. The three scores are all important for the final ranking. It is reasonable that the final ranking score is defined as the product of the three scores:

$$score_R(q,w,c) = score_Q(q,w) \times score_I(w) \times score_C(c,w)$$

where $score_Q(q,w)$ is the query score of document w to query q, $score_I(w)$ is the importance score of document w, and $score_C(c,w)$ is the category score of document w on category c. If users simply browse a category without any keyword, $score_Q(q,w)$ is constant for all webpages. SEE will return all the indexed webpages predicted in this category with the most popular and most category-relevant ones at the top. In this case, SEE becomes a web directory, and people can browse the most popular webpages in each topic of the whole hierarchy with a good accuracy.

4.2 User Interface Design

In SEE, only the webpages with the likelihood probability greater than the classification threshold will be returned in the corresponding category for the ranking. However, using a fixed threshold to decide the category of webpages may not suit all users. Can we make the threshold flexible and customizable? To tackle this problem, we introduce a slider bar in the user interface (see Figure 1) for users to dynamically adjust their desire for more results or higher accuracy. When users drag the slider to the left side, it lowers the classification threshold, and some false negative webpages might become true positive and be displayed to the users. Users will see more (but perhaps less accurate) results. On the other hand, when the slider is dragged to the right, it increases the classification threshold, and some false positive webpages might become true negative and disappear from the returned list. Users will see more accurate (but perhaps fewer) results.

It is actually a fast and efficient operation to update the returned results dynamically when the slider is moved. When the users drag the slider to a new threshold value, the system just needs to compare the corresponding probability (already stored during the indexing phase) of each webpage with the threshold. Users can instantly see the updated results as they move the slider.

Besides the slider bar, we also introduce a topic search box on the left side of the user interface. When users are not sure about the location of the topic in the hierarchy, they can use the search box to find the topic first. We use the partial keyword matching for the topic search. After the user types in a few indicative characters of the topic name, the top ten matched topics will be shown and the user can select the desired topic from them. This functionality makes it more convenient for the new users of the search engine, since they may not know the structure of our topic hierarchy. However, we believe that over time a well-thought-out topic hierarchy can become familiar to users. New and popular topics can also be introduced into the hierarchy over time.

4.3 Evaluation

To evaluate our search engine, we still use the same DMOZ dataset as we used in Section 3.2. We index all of the 1.2 million webpages with the likelihood probabilities obtained in *the test set* of the three-fold cross validation mentioned in Section 3.2. That is, the categories and probabilities of each webpage in SEE are predicted by SVMs rather than given. This mimics the classification of new webpages whose categories must be predicted by our search engine.

In this section, we conduct a comparison between the flat and hierarchical versions of SEE. As currently SEE only indexes 1.2 million webpages, it cannot be compared directly with Google and Bing. We use one sample query on both the flat and hierarchical versions of SEE and compare their returned results. Then, a usability study is followed to further compare the two search engines.

A Test Case

Suppose we want to search information about the training programs in needlework. By using the hierarchical version of SEE, we can browse the topic hierarchy and select Arts→Crafts→Needlework, then search the keyword "training". In total, 15 results are returned with the slider in the middle (as the default threshold of 0.5). The top ten URLs

Results of the hierarchical version		Results of the flat version
Results with slider in the middle	Results with slider to the left	
http://www.carol-ann.co.uk/	**http://www.carol-ann.co.uk/**	http://www.serendipitytours.com/
http://www.lyndisfarne.com/	http://www.lyndisfarne.com/	http://orientcrafts.safewebshop.com/
http://egausa.org/	**http://www.crcsales.com/**	http://www.birdcrossstitch.com/
http://www.lequilters.com/	**http://egausa.org/**	http://www.purewhimsy.com/
http://www.classiccrossstitch.com/	**http://www.servicewatchsystems.com/**	**http://egausa.org/**
http://orientcrafts.safewebshop.com/	**http://www.sewingandembroiderywarehouse.com/**	http://www.millennia-designs.com/
http://www.planetembroidery.com/	**http://www.strawberrystitch.com/**	http://www.judy.com.au/
http://www.millennia-designs.com/	**http://www.fwebb.com/**	http://goddessrising.homestead.com/
http://www.embroideryabc.com/	**http://www.lequilters.com/**	http://www.goblenart.com/
http://www.purewhimsy.com/	**http://www.estherfaye.co.uk/**	http://www.tatting.co.uk/

Table 3: Top ten results returned by the hierarchical and flat versions of SEE (URLs in bold are the relevant URLs). The left column shows the results returned by the hierarchical version with threshold 0.5. The middle column shows the results returned by the hierarchical version with threshold 0.3. The right column shows the results returned by the flat version. (The results are retrieved by SEE on January 11, 2011)

are presented in the left column of Table 3. After carefully checking into those ten URLs, we find that four (in bold) are relevant to "training in needlework". If we wish to find more results, we can simply drag the slider to the left (threshold equals 0.3). In total, 41 results are returned and the top ten results are updated as shown in the middle column of Table 3. We can see that nine out of the top ten are relevant URLs. This verifies the functionality of the slider bar. When the slider goes left, more webpages that match the keywords will be returned and probably users can find their desired results.

Now, we search for the same information in the flat version of SEE with keywords "needlework training". The top ten of the total 88 results are shown in the right column of Table 3. We can see that there is only one URL (the 5th) relevant to the search purpose. Even by going further to the top 30 URLs, we still find only three relevant ones.

We further analyze the results returned by the hierarchical version and find that most of the top ranked webpages do not have the keyword "needlework"; instead, they contain related keywords such as "sewing", "cross stitch" and "embroidery". Often users of a search engine do not know exactly the right keywords to use, as they may not be familiar with the domain (such as "needlework"). If there is an appropriate category for them to choose, they can focus on the more important keywords (such as "training") in that category. From this simple example, the advantage of the hierarchical version of SEE over the flat one is obvious.

Usability Study

We choose a set of information-seeking tasks across different domains (e.g., find information about diseases of the crops) and set up a testing webpage on our website. When a user visits this testing webpage, he/she will be asked to do the same search task (randomly chosen from the task set) in both the flat and hierarchical versions of SEE, called search engine A and search engine B, as a blind test, on the test page. [Olston and Chi, 2003] also uses a pool of tasks to conduct their user evaluation in the similar way. After using both search engines, the user is asked to provide optional answers to two questions. They can also leave some comments if they wish. Each user can do multiple tasks and provide multiple feedback. It should be mentioned that, in our evaluation, no hints or guideline are given to participants on how to use the flat and hierarchical versions of the SEE; they simply figure things out themselves. This method was also employed in the usability study of [Käki, 2005].

Which search engine do you like better?	
A is much better than B	4%
A is slightly better than B	15%
A and B are roughly the same	11%
B is slightly better than A	37%
B is much better than A	33%
Did you use the slider bar?	
No	48%
Yes, but useless	19%
Yes, it helps a bit	26%
Yes, it helps a lot	7%

Table 4: Results of evaluation of the blind test. A and B represent the flat and hierarchical versions of SEE, respectively.

We asked 30 undergraduate students to participate the testing anonymously and voluntarily. Eventually, we get 27 feedback. The results of the feedback are given in Table 4. As we can see in the table, overall 70% of participants think that the hierarchical version of SEE is better than the flat version and only 19% have the opposite opinion. This result demonstrates that the hierarchical version of SEE is favorable to these users. However, in terms of the slider bar, only 52% of participants used the slider bar when they were searching, and only 33% of participants found it helpful. This result is within our expectation, since the usage of the slider bar is new to the users of search engines. Overall, the feedback is positive and encouraging.

5 Conclusion and Future Work

In this paper, we introduce a novel search engine, called SEE (**S**earch **E**ngine with hi**E**rarchy), with functionalities of both keyword search and hierarchical topic browsing. Machine learning and data mining techniques are integrated into our system to automatically classify the massive number of web-

pages into a predefined topic hierarchy. Besides, a new ranking algorithm and a novel user interface are embedded into SEE. According to a small usability study, SEE is favorable to the majority of users.

In future work, we will index more webpages to test the performance of SEE against other search engines. Since the labeled examples are limited and human annotation is costly, active learning may be a good direction to follow. Moreover, to better capture users' search intention, we will incorporate more hierarchies, such as media types and regions of the world, into SEE. Some existing speciality search, such as video, images, and music, can also be introduced. In this way users can then search across different hierarchies and different content.

References

[Bradley, 1997] A.P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[Brenes *et al.*, 2009] David J. Brenes, Daniel Gayo-Avello, and Kilian Pérez-González. Survey and evaluation of query intent detection methods. In *Proceedings of the 2009 workshop on Web Search Click Data*, WSCD '09, pages 1–7, New York, NY, USA, 2009. ACM.

[Broder, 2002] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36:3–10, September 2002.

[Cronen-Townsend and Croft, 2002] Steve Cronen-Townsend and W. Bruce Croft. Quantifying query ambiguity. In *Proceedings of the second international conference on Human Language Technology Research*, HLT '02, pages 104–109, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[Ferragina and Gulli, 2005] Paolo Ferragina and Antonio Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 801–810, 2005.

[Forman, 2003] George Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, March 2003.

[Hearst, 2006] M. Hearst. Design recommendations for hierarchical faceted search interfaces. In *ACM SIGIR Workshop on Faceted Search*, pages 1–5. Citeseer, 2006.

[Japkowicz and Stephen, 2002] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6:429–449, October 2002.

[Joachims, 1998] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. pages 137–142. Springer, 1998.

[Käki, 2005] Mika Käki. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 131–140, New York, NY, USA, 2005. ACM.

[Liu *et al.*, 2005] Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. In *ACM SIGKDD Explorations Newsletter - Natural language processing and text mining*, volume 7, pages 36–43, New York, NY, USA, June 2005. ACM.

[Olston and Chi, 2003] Christopher Olston and Ed H. Chi. Scenttrails: Integrating browsing and searching on the web. *ACM Trans. Comput.-Hum. Interact.*, 10:177–197, September 2003.

[Page *et al.*, 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1998.

[Platt, 1999] John C. Platt. Probabilistic outputs for support vector machines. *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[Qi and Davison, 2009] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41:12:1–12:31, February 2009.

[Silla and Freitas, 2011] Carlos Silla and Alex Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72, 2011.

[Sun and Lim, 2001] Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 521–528, Washington, DC, USA, 2001. IEEE Computer Society.

[Vens *et al.*, 2008] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73:185–214, November 2008.

[Xue *et al.*, 2008] Gui-Rong Xue, Dikan Xing, Qiang Yang, and Yong Yu. Deep classification in large-scale text hierarchies. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 619–626, New York, NY, USA, 2008. ACM.

[Yang, 1999] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, May 1999.

[Zeng *et al.*, 2004] Hua-Jun Zeng, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. Learning to cluster web search results. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 210–217, New York, NY, USA, 2004. ACM.