Citepack: An Autonomous Agent for Discovering and Integrating Research Sources

Christopher H. Brooks and **Yeh Fang** and **Ketaki Joshi** and **Papanii Okai** and **Xia Zhou**
Computer Science Department
University of San Francisco
2130 Fulton St.
San Francisco, CA 94117-1080
{cbrooks, yfang5, ksjoshi, nokai, xzhou2}@cs.usfca.edu

Abstract

This paper describes CitePack, an autonomous, agent-based system for automatically discovering resources based on learned user preferences. We discuss the need for a tool that can find information for a user without direct intervention, followed by the architecture of CitePack. We discuss the collection of related documents, and the construction of a user model based upon support vector machines, which allows us to both learn a user's tastes and also recommend similar users. We conclude by discussing potential future directions.

Introduction

The recent explosion in the availability of online, hyperlinked information sources, including both specialized sources such as CiteSeer (Bollacker, Lawrence, & Giles 1998) and the ACM Digital Library [1] and more generalized information sources, including the World Wide Web (WWW) itself, has created a need for tools that assist users in searching and organizing this information space and managing and sharing their own personal information. In particular, we are interested in developing *agents* that *autonomously* discover and filter information on a user's behalf.

Consider the following scenario: Jane Doe the researcher is reading an online article about information retrieval. This article contains a link to a second article (on, say, clustering) that Jane would also like to read. Jane has a few choices; she could stop reading the information retrieval article and switch context to the clustering article, she could bookmark or print the clustering article and remember to return to it later, or she could ignore the clustering article altogether. Most people would probably choose to bookmark or print the article, but this can be hard to manage.

We have developed an agent-based tool called CitePack to help address this problem. CitePack allows a user to easily indicate a paper of interest and autonomously fetches this paper, as well as other, similar papers, and related information about the subject of the paper, such as software, conferences, courses, and author home pages. This information is encapsulated in an RSS feed, which the user can then browse using either her own RSS reader or through the CitePack web site. CitePack allows the user to provide feedback about papers and resources she likes, thereby improving the search, and also allows users to form groups and to share their search results with other users.

This paper describes the architecture and design of CitePack. We begin with a more detailed use case, and then describe each of the components, starting with the collector, which finds resources, and the modeler, which constructs and refines a model of user preferences. We discuss the problem of parameter selection for the modeler, and end with a discussion of future directions.

There are many other systems that assist users in discovering information. For example , MarginNotes (Rhodes 2000) and Watson (Budzik & Hammond 1999) were systems that watched a user's working document and autonomously discovered related information. WebTop (Wolber *et al*. 2001) extended this idea to allow users to choose to share portions of their *personal web* with other users. On the search engine side, the idea of metasearch engines that aggregated data was explored by projects such as Inquirus (Lawrence & Giles 1998). There has also been work on creating portals specifically for locating computer science research papers, including Citeseer, BibFinder (Nie, Kambhampati, & Hernandez 2003) and Rexa (McCallum *et al*. 2000). We argue that CitePack's contribution is the combination of a metasearch engine with an autnomous agent that searches and filters on a user's behalf, learning their preferences.

Use Case

This section presents a detailed example to illustrate CitePack's functionality. Let's suppose that researcher Jane Doe is interested in informartion retrieval. While reading an online article about information retrieval, she notices a link to an paper on document clustering, which she's wanted to learn more about. She'd rather not interrupt the article she's currently reading, so she uses the CitePack Firefox plugin to *annotate* the clustering paper. (see figure 1.) She adds a few tags describing the clustering paper, and then continues reading her original article.

In the background, a collector agent is invoked on the CitePack server. Jane's tags are automatically submitted

[1] portal.acm.org

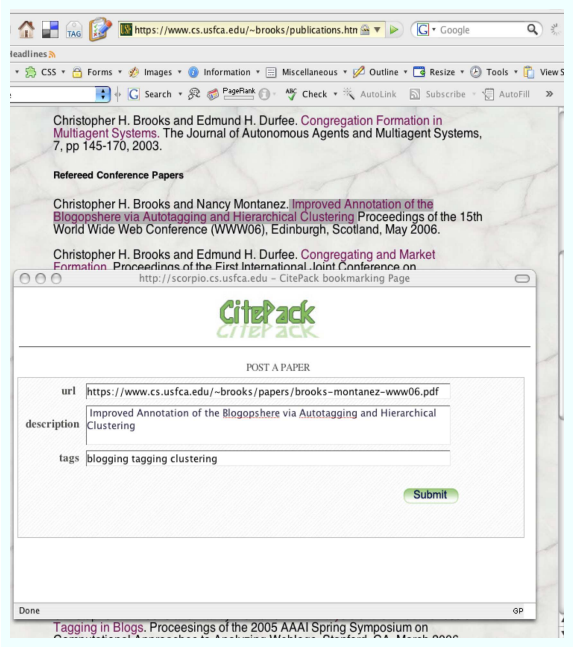

Figure 1: Tagging a URL for use with CitePack

to del.icio.us [2] for sharing, and also stored in the CitePack database. The collector then fetches references to the tagged paper from CiteSeer and the ACM digital library, and then crawls both of these sites to find other papers deemed to be similar, either because of similar content or else because of co-citation.

Once at least 50 candidate papes are discovered, this pool of candidates is filtered by comparing them to a user model. CitePack uses a support vector machine (SVM) (Joachims 2001) to construct a model of Jane's preferences. Each candidate paper is passed through the SVM, which classifies the paper as either "liked" or "not liked." "Liked" papers are retained, and "not liked" papers are discarded.

CitePack then tries to find other information on the Web related to this paper. Curently, we capture five types of information: related conferences, software, courses, books, and author home pages. Online resources in each of these categories are retrieved and then passed through the SVM for filtering.

Once all of this information has been gathered, an RSS feed is generated. At her convenience, Jane can then view this information, either through the RSS reader of her choice or else via the CitePack website. (see figure 2.) The website allows her to rate the resources Citepack has collected. This information is then folded back into the SVM for further training, thereby allowing it to make more accurate predictions in the future.

Jane can also create a group, allowing her to easily share these resources with her colleagues. Any resources discovered by one group member will also be added to the group's RSS feed. (see figure 3). CitePack will also autonomously

[2] http://del.icio.us

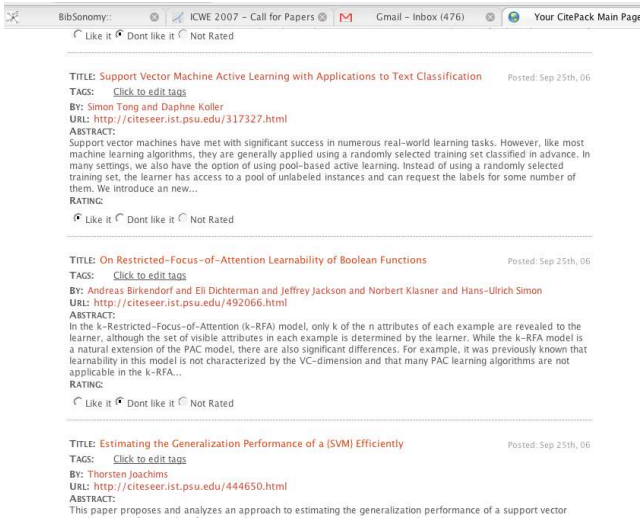

Figure 2: Related papers discovered by CitePack

discover other users with interests similar to Jane's, and recommend their RSS feeds to her, thereby allowing her to find out about resources and potential colleagues she might not have been aware of.

Collector

When Jane tags and submits an article to CitePack, the first component that is invoked is the *collector*. The collector is a specialized crawler implemented on top of the Slashpack framework (Brooks *et al.* 2006). The collector queries multiple information sources (currently Citeseer and the ACM Digital Library) for references to this paper, and then begins a breadth-first crawl on each site, following the "Find similar articles", "related by co-citation", and "related by similar text" links.

One challenge the collector faces is the presence of near-duplicates. Many times this approach will find several slightly different versions of a paper, such as a conference version and a journal version. We suspect most users would prefer only to see one version of the paper, so the others must be detected and removed. We compute an MD5 hash of each paper, which will detect exact duplicates, but not near-duplicates. To solve this problem, we use the well-known technique of *shingling* (Broder *et al.* 1997). Shingling treats each document as a string and computes a similarity based on the overlap of the two strings.

We compared shingling to Naive Bayes as a technique for detecting near-duplicates. Naive Bayes treats a document as a "bag of words", each of which has a conditional probability based on its frequency in a corpus of documents. We use this to compute the probability that both documents belong to the same class.

Naive Bayes is more efficient than shingling ($O(n)$ vs $O(nlog(n/m))$(Broder *et al.* 1997), but more prone to false positives, as it does not consider word ordering or proximity. For our problem, we found shingling to be more effective, as two versions of a paper's title or abstract often contained

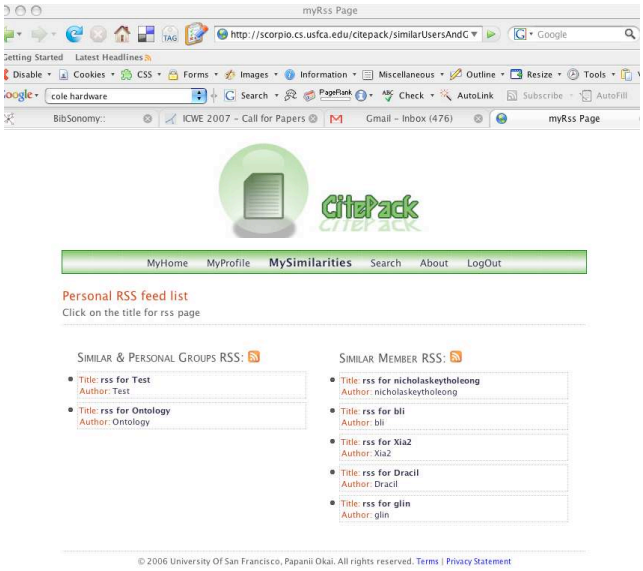

Figure 3: Related Groups and Users

identical substrings.

At this point, we keep the most recent version of the paper; future extensions would allow a user to specify a preference, such as journal articles over conference papers.

Once at least 50 pages have been collected, the collector then invokes the modeler to filter these papers based on the user's preferences.

User Modeling

Following "related" links is a great way to find other resources that are similar to Jane's original paper, but it leaves out some crucial information: all of the other papers Jane has tagged in the past. This can potentially give us a lot of other data about Jane's interests, and help to eliminate papers not of interest to her.

We model a user's preferences using a support vector machine (SVM). A detailed discussion of the workings of SVMs and their application to text classification is beyond the scope of this paper, but can be found in Joachms, et al (Joachims 2001). In short, an SVM treats a document as a weighted vector, and constructs a hyperplane that separates papers indicated as "liked" from papers indicated as "not liked." This hyperplane can then be used to easily classify unseen papers based on their position with respect to the hyperplane. We used Joachims' svm^{light} library (Joachims 1999) for this task.

Applying SVMs to this task of document classification is not without its challenges. The typical paper has many features, of which we considered four: author, abstract, title, and user-assigned tags. svm^{light} requires the user to indicate the relative importance of these features, which meant that we needed to perform a separate set of experiments to determine the weight that should be assigned to each of these features. We did this by having a set of users each rate at least 50 papers and then exhaustively searching the parameter space to find the combination of parameters that performed best using n-fold cross-validation.

One question that came up along the way was whether to use one set of parameters for all users, or whether to learn a separate set of parameters for each user. Having separate parameters for each user was determined to produced a 9% improvement in accuracy. Examining the results, we found that some users focused primarily on the author and title when selecting papers, whereas other users focused primarily on the abstract.

An advantage of using an SVM to learn user preferences its ability to learn incrementally and incorporte new evidence. Each time a user tags a paper in CitePack, she is presented with a list of recommendations and the opportunity to provide feedback for each of those recommendations. Every time we receive 20 new recommendations, the SVM is reconstructed, thereby improving the user model.

Discovering similar users

Once we have a user model, we can use this to discover other users with similar preferences. We do this by constructing a *virtual document* for each user, consisting of all papers that user has rated positively. We then construct a vector model (Baeza-Yates & Ribiero-Neto 1999) for this virtual document using the standard TFIDF weighting, in which words that occur frequently in the virtual document but rarely in all documents are given the most weight.

We then compare the vector models of each user with both cosine similarity and the Jaccard measure (Baeza-Yates & Ribiero-Neto 1999); all users within a system-defined threshold are considered to be "similar enough" and recommended to Jane.

To date, this module has only been tested with synthetic data, but has been able to accurately cluster users into four groups based on the papers they've rated.

External Resources

Finally, the collector is used to also discover other resources that Jane might be interested in. Currently, we consider five types of resources: books, software, webpages for courses, author home pages, and webpages for conferences. In all five cases, we take the approach of automatically constructing a search query for use with an existing Web Service provided by either Google or Amazon.

The interesting challenge is in forming this query. Our first approach was to use the relevant words in the title of the paper. However, these are often far too specific to yield meaningful results. For example, a user who has tagged the paper " A Statistical Learning Model of Text Classification with Support Vector Machines" might be interested in conferences on information retrieval, data mining, or machine learning, even though none of those phrases are in the title.

We chose to solve this problem by constructing a taxonomic classifier derived from the ACM Computing Classification system. This system subdivides computer science into a number of areas and subareas, with specialized keywords at the "leaves" in the taxonomy. Once we determine

the appropriate leaves for the title in question, we then replace them with the more general category terms extracted from the taxonomy. We consider all one, two, and three-word phrases within the title in doing so.

For example, for the paper "A Statistical Learning Model of Text Classification with Support Vector Machines", we would use the taxonomic classifier to predict categories for every one-word, two-word, and three-word substring of the title. This would produce matches for "Learning", "Text Classification", and "Support Vector Machines", which would be mapped to "Learning," "Information Storage and Retrieval," and "Pattern Recognition."

These new, more general terms are then used as the basis for queries that are sent to Google and Amazon. Each query is also augmented with additional keywords that help identify the particular type of information (for example, "Call for Papers" for conferences, "home page" for home pages, or "sourceforge" for software). The resulting URLs are then passed off to the SVM for filtering, just as with papers.

Future Work

Currently, we feel that CitePack is in the alpha stage. One thing that is needed from a software development standpoint is a scaling up of the number of users. Currently, it's been vetted primarily by students at USF, which has limited the generality, as well as the number of users. A larger user study will allow us to evaluate CitePack "in the wild" and determine whether it is truly a useful tool.

We are also interested in integrating CitePack with other resources that allow "people tracking", such as Peoplicious [3] and linkedIn [4]. This will allow us to more easily relate authors to papers, and discover other resources that might be of interest to a user, such as other people at the same institution, or advisors, or emloyees.

We would also like to add other data sources to CitePack. Currently, the biggest limitation to this has been the fact that many services, such as Google Scholar [5], explicitly forbid software agents from accessing their services. (While there are several applications that use Google Scholar results through screen-scraping, this would appear to be a violation of their terms of use.)

Conclusions

This paper has described the design and implementation of Citepack, a system allows users to easily annotate papers of interest and then discovers other resources realted to these papers. CitePack incrementally learns a model of user preferences using a support vector machine and uses this model to filter potentially interesting papers and match users with others who have similar interests. Useful references are compiled into an RSS feed, allowing the user to browse them at her convenience.

[3] www.peoplicious.com

[4] www.linkedin.com

[5] scholar.google.com

References

Baeza-Yates, R., and Ribiero-Neto, B. 1999. *Modern Information Retrieval*. Addison-Wesley.

Bollacker, K.; Lawrence, S.; and Giles, C. L. 1998. CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications. In Sycara, K. P., and Wooldridge, M., eds., *Proceedings of the Second International Conference on Autonomous Agents*, 116–123. New York: ACM Press.

Broder, A. Z.; Glassman, S. C.; Manasse, M. S.; and Zweig, G. 1997. Syntactic Clustering of the Web. Technical Report 1997-015, Systems Research Center, Palo Alto, CA. Available at http://gatekeeper.research.compaq.com/pub/DEC/SRC/technical-notes/SRC-1997-015-html/.

Brooks, C. H.; Agarwal, M.; Endo, J.; King, R.; Montanez, N.; and Stevens, R. 2006. Slashpack: An Integrated Toolkit for Gathering and Filterng Hypertext Data. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*.

Budzik, J., and Hammond, K. 1999. Watson: Anticipating and Contextualizing Information Needs. In *62nd Annual Meeting of the American Society for Information Science*.

Joachims, T. 1999. Making large-Scale SVM Learning Practical. In Schlkopf, B.; Burges, C.; and Smola, A., eds., *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Joachims, T. 2001. A Statistical Learning Model of Text Classification with Support Vector Machines. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*. ACM Press.

Lawrence, S., and Giles, C. L. 1998. Context and Page Analysis for Improved Web Search. *IEEE Internet Computing* 2(4):38–46.

McCallum, A.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the Contruction of Internet Portals with Machine Learning. *Information Retrieval Journal* 3:127–163.

Nie, Z.; Kambhampati, S.; and Hernandez, T. 2003. Bibfinder/statminer: Effectively mining and using coverage and overlap statistics in data integration. In *Proceedings of VLDB 2003*.

Rhodes, B. 2000. Margin Notes: Building a Contextually Aware Associative Memory. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, 219–224. ACM.

Wolber, D.; Witchell, J.; Kepe, M.; and Ranitovic, I. 2001. Exposing Document Context in the Personal Web. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI), 2001*.