12-2011

WebTailor: Internet Service for Salient and Automatic User Interest Profiles

John Anderson
University of Arkansas, Fayetteville

WEBTAILOR: INTERNET SERVICE FOR SALIENT AND AUTOMATIC USER INTEREST PROFILES

WEBTAILOR: INTERNET SERVICE FOR SALIENT AND AUTOMATIC USER INTEREST PROFILES

A thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

By

J. Patrick Anderson

December 2011
University of Arkansas

This thesis is approved.

Thesis Director:

_____

Dr. Susan Gauch

Thesis Committee:

_____

Dr. Russell Deaton

_____

Dr. James Parkerson

ACKNOWLEDGEMENTS

I would like to thank my advisor and committee chair, Dr. Susan Gauch, for her inspiration, guidance, and patience.

I would also like to thank Dr. Russell Deaton and Dr. James P. Parkerson for being on my defense committee.

Finally, I would like to thank both my brother and my fiancé for proof reading this thesis, and for humoring my seemingly endless talk about grandiose ideas.

ABSTRACT

Website personalization systems seek to give users unique, tailored content and experiences on the Internet. A key feature of these systems is a user profile that represents each user in a way that distinguishes them from others. In current personalization systems, the data used to create these profiles is extremely limited, which leads to a host of problems and ineffectual personalization.

The main goal of this thesis is to improve these personalization systems by addressing their lack of data and its poor quality, breadth, and depth. This is accomplished by analyzing and classifying the content of each user's Internet browsing activity, rather than just their activity on a single website, to autonomously build persistent, ontology-based user profiles. Furthermore, these profiles are built and stored on a remote server, which allows them to be easily made available to approved websites in the interest of providing the data to enable accurate, relevant, and up-to-date personalization.

TABLE OF CONTENTS

List of Figures iv

I. INTRODUCTION 1

A. Motivation 1

B. Current Problems 2

C. Goals and Contributions 3

D. Overview 4

II. RELATED WORK 7

A. Autonomous Profile Creation 7

B. Ontology-based Systems 9

III. BUILDING A USER PROFILE 11

A. Overview of Profile Creation and Maintenance 11

B. System Architecture 12

C. Introduction to Ontologies 15

D. Training the Classifier 17

E. Collecting Data 20

F. Classifying Webpages and Updating a Profile 20

4. EVALUATION 23

5. CONCLUSIONS AND FUTURE WORK 27

A. Conclusions 27

B. Future Work 27

BIBLIOGRAPHY 31

List of Figures

Figure 3.1: The process of building a user profile ……. 12

Figure 3.2: ODP hierarchy example ……..…. 17

Figure 3.3: Example dictionary file ……..……. 19

Figure 3.4: Example postings file ……..…. 19

Figure 4.1: Example profile showing categories' names and weights ..……. 23

Figure 4.2: Disabling the browser extension ……. 26

I. INTRODUCTION

A. Motivation

The Internet contains an unfathomable amount of information on practically any given subject and enables a wide range of activities such as e-commerce, media consumption, project collaboration, topic discussion, political coordination, and social networking. This abundance of information, content, and near-instantaneous communication across great distances has spawned a host of new technologies and benefitted mankind immensely, but there is still much more that could be improved in the way the Internet works.

When searching for information or participating in one of the aforementioned activities, Internet users are largely anonymous and treated in a homogeneous manner no matter the differences in their interests, habits, or personalities. Internet personalization systems seek to improve on this "one-size-fits-all" approach to user/system interaction by giving each user unique, tailored content and experiences on the Internet rather than the generic ones provided by default. Personalization systems can be used to enhance a wide range of Internet activities, such as tailoring search results to each user or recommending content that the user is likely to find interesting or enjoyable.

A key component of Internet personalization systems – the data structure upon which the systems fundamentally depend – is the user profile that represents each user in a way that distinguishes them from others. The data contained in a profile can be as simple as basic demographic information or can be more complex to include information about the user's tastes, habits, or browsing history.

B. Current Problems

The majority of website personalization efforts are extremely limited and range in effectiveness from mediocre to almost completely ineffectual. The root of the problem lies in the quantity and quality of the data that websites are collecting for their personalization systems, namely that the data is collected from only one source: the user's activity and/or feedback on the website doing the personalization. This leads to several issues with the personalization data collected.

First, the range and scope of personalization data are extremely limited. As an example, Amazon's[1] product recommendations system is completely unaware of how each user rates certain movies on Netflix[2]. This makes the system's recommendations less relevant and less accurate because their narrow focus ignores the full breadth of the user's potential profile.

As a corollary of the first issue, a single user's profile can vary drastically across personalized websites; because site X does not have access to site Y's personalization data, the two sites can have very different profiles (both in terms of structure and in terms of content) for the same user on which to base their respective personalization. This leads to inconsistencies in user profiles between personalization systems, creating a partial and inaccurate conception of the user in each system.

Thirdly, such limited personalization data makes it difficult to keep the user profiles on these sites current with each user's fluctuating tastes and interests. If a user visits a personalized website infrequently while their preferences change fairly regularly,

[1] http://www.amazon.com
[2] http://www.netflix.com

the user's profile in the website's personalization system can easily become out-of-date and lead to ineffectual personalization.

Finally, privacy is a large concern for many Internet users. They would like control over what personal information is collected and how it is used, but current implementations of personalization systems do almost nothing to address these concerns.

C. Goals and Contributions

Many Internet applications – search engines, product and content recommendation services, news aggregators, *et cetera* – could greatly benefit from more accurate, current, and comprehensive personalization data. This thesis and its corresponding project aim to meet those needs by creating an Internet service that will automatically generate, maintain, and make available profiles of individual users that accurately reflect the breadth and depth of their many interests and allow those users more control over their profile than is given by other personalization systems. It will accomplish this by:

- using a browser extension to collect data on users' browsing activities,
- utilizing a server-side system for classifying the webpages the user visits and for creating weighted categorical user profiles,
- creating a website for users to view their profiles, and
- enabling the future creation of a web API which will disseminate user profiles to user-approved third parties for use in their website personalization systems.

This thesis focuses on autonomously building and maintaining long-term, ontology-based user profiles. It is based off of work done by the Ontology Based Web Navigation Agent (OBIWAN) project [Pretschner99, Zhu99] and the KeyConcept project

[Gauch2003]. It aims to extend these projects by enabling the user to build his/her profile regardless of location and to allow for the construction of an API permitting approved third-parties to access users' profiles in the interest of improving website personalization systems. This thesis' corresponding system, called "WebTailor," will accomplish these goals by implementing the creation and maintenance of profiles as an online "cloud computing" service, rather than being tied to a specific personal computer or local network.

D. Overview

This thesis relates to the construction of user profiles in the interest of web personalization, which can be defined as the process of delivering the most relevant information to each individual user "in the most adequate way at the most appropriate time" [Mizzaro2002]. In this field, user profiles can generally be classified as either long-term or short-term. In systems utilizing long-term profiles, the user profile represents the (condensed) sum total of data collected since the user began using the personalization system. These profiles are built over time by analyzing the user's search, browsing, communication, and other Internet activities. Conversely, short-term profiles are constructed by considering only the most present actions of the user and are generally destroyed after a single session.

For some use cases, such as suggesting purchases for a user who is utilizing the Internet for holiday shopping, short-term profiles may be ideal. However, for the more general use case of providing relevant personalization data for virtually any Internet activity or website, a short-term profile is inadequate because it does not provide enough quantity, breadth, or depth of information about the user. Therefore, this thesis focuses

on enabling personalization with long-term, persistent profiles that are continuously updated in order to ensure their completeness and relevance.

Personalization systems construct long-term user profiles in one of two ways: autonomously or with a method dependent on user input (such as rating content or saving bookmarks). This thesis focuses on constructing its profiles autonomously, without direct input from the user as to what data is included in their profile. This method of profile creation and maintenance was chosen for two reasons. First, non-autonomous systems are by design more demanding on the user, requiring active participation in the creation and maintenance of his/her profile, whereas autonomous systems do not require any major change in the user's activities. They "just work" somewhat invisibly to the user. Secondly, autonomous systems produce a more complete, accurate profile by eliminating the influence of any biases the users may hold about themselves.

Several different data structures may be used to represent a user profile. In past work, profiles have been represented as a list of bookmarks [Lieberman95], as a collection of vectors in the N-dimensional document vector space [Chen97], and as a weighted hierarchical ontology of concepts or categories [Trajkova2003]. The last approach was chosen for this thesis because it provides a simple yet robust method of representing a user profile as both the general and the more specific categories that a user is interested in, rather than just as the (often ambiguous) words that semantically indicate a concept or as the documents that exemplify it.

This thesis presents information on previous related work in Chapter II. Chapter III describes the process for generating and maintaining user profiles, and Chapter IV

analyzes and evaluates the results of this process. Chapter V summarizes this thesis, explores its implications, and suggests ideas for improvements to the system and for future research.

II. RELATED WORK

A. Autonomous Profile Creation

Autonomous profile creation systems are those systems that construct user profiles by collecting data without the direct input of the user as to what is included in their profile. These systems are very easy to use and undemanding of the user, who simply continues with his/her normal browsing or searching activities while the autonomous system builds their profile invisibly in the background.

One of the first systems to take an autonomous approach to building user profiles was ***Letizia*** [Lieberman95]. It is a content recommendation system – suggesting links and documents to its users – which builds its profiles by analyzing the links the user follows or stores as bookmarks. Letizia recommends content by performing a breadth-first search of any hyperlinks on the current webpage, representing the contents of those documents as keyword vectors, and comparing those vectors to the user's profile to suggest which link(s) the user should follow for the most appropriate information.

Personal WebWatcher [Mladenic98] is a content-recommendation system similar to Letizia in function, but significantly different in implementation. Personal WebWatcher utilizes a proxy server that analyzes the contents of the user's browsing activities and represents each webpage as a keyword vector, with the sum total of the individual vectors comprising the user's profile. To recommend the next link to follow, Personal WebWatcher analyzes the content of each link on the user's current page, and compares it them to the user's profile using a naïve Bayes classifier. By considering which links the user follows and which he/she does not, the user profile is extended to

include classes of documents determined "interesting" and others "not interesting," to further guide content recommendations.

Another autonomous system that uses a proxy server to autonomously build user profiles is ***WebMate*** [Chen97]. In addition to suggesting content, WebMate also features functionality for extracting keywords from a document, in order to further refine search engine results. It accomplishes these two functionalities by representing each user profile as a predefined number of vectors; when the user visits a new webpage of interest, it is represented as a vector of keywords and added the user's profile if the profile has not reached the predefined limit of concepts. If the limit has been reached, the WebMate system compares each vector in the user profile to every other one, and combines the most similar into new concepts that are labeled with the word that has the highest weight in the new vector.

An interesting hybrid system that builds user profiles autonomously but also accepts relevance feedback from the user is called ***Quickstep*** [Middleton2001]. Though limited to analyzing only documents in PDF or PostScript formats, the system utilizes a proxy server for data collection and represents the content of each document the user views as a keyword vector and then utilizes both machine learning algorithms and the user's relevance feedback to construct a profile. The profile is represented as a list of topics and their related weights, which correspond to how interesting the user is likely to find that particular topic. When recommending new papers to the user, Quickstep calculates the correlation between the paper and the topics currently in the user's profile, and if the correlation score is high the system only recommends the document if the user has not seen it before.

B. Ontology-based Systems

Ontology-based systems are those that represent the user profile as a weighted hierarchy of concepts or categories, as opposed to a collection of keyword vectors, saved bookmarks, list of topics, or other means of representation. Ontology-based systems can be used to improve search and browsing by matching the user's interests to a predefined ontology, and then comparing the contents of the search or browsing to the user's profile and tailoring the results or recommending content accordingly. Another way to explicitly use ontology-based systems is to include concept tags in a document's metadata, which allows intelligent retrieval agents to retrieve documents that belong in the same category rather than just ones that have similar keywords.

The personalized search system of OBIWAN [Pretschner99] is an autonomous, ontology-based system that represents user profiles as weighted categories from the online Magellan Directory[3]. The documents from each category in the ontology are merged and each category is represented as a keyword vector. Rather than making use of a proxy server to collect user browsing information, the system utilizes the browser's cache; each webpage the user visits is also represented as a keyword vector and is then matched with the vectors of Magellan's predefined concept hierarchy and the most similar category's weight is increased in the user profile. OBIWAN uses these adaptive profiles to tailor search results for the user by comparing the results of a search to the user's profile and re-ranking the results accordingly.

[3] http://www.magellan.cc/

The Simple HTML Ontology Extension (***SHOE)*** is a system that utilizes ontologies to allow authors to add semantic data to their documents, enabling their retrieval by intelligent search agents [Luke97, Heflin99]. This semantic labeling is done using HTML tags in a document's metadata; the tags can be generated by using or extending existing online ontologies.

Other ontology-based systems include ***SmartPush*** [Kurki99] and ***Persona*** [Tanudjaja2001]. However, these systems require significant user interaction in order to build their profiles. Because of their non-autonomous nature, they are not examined in more depth in this thesis.

WebTailor combines aspects of several different systems detailed in this chapter to autonomously create user profiles relevant to the broader goal of enabling overall Internet personalization. Following the work done by Personal WebWatcher [Mladenic98] and WebMate [Chen97], WebTailor represents documents as keyword vectors. However, WebTailor calculates the similarity between documents by using a cosine similarity metric in the vector space model rather than using a naïve Bayes model. Similar to OBIWAN [Pretschner99, Zhu99], user profiles are represented as a weighted hierarchy of concepts, however a different ontology and training collection is used instead of Magellan.

III. BUILDING A USER PROFILE

A. Overview of Profile Creation and Maintenance

Before generating weighted categorical user profiles, an ontology describing the permitted categories and their relationships to one another was obtained. Next, a collection of training documents that had been manually classified into categories from the ontology were indexed, creating a dictionary and postings for the purposes of training a document classifier.

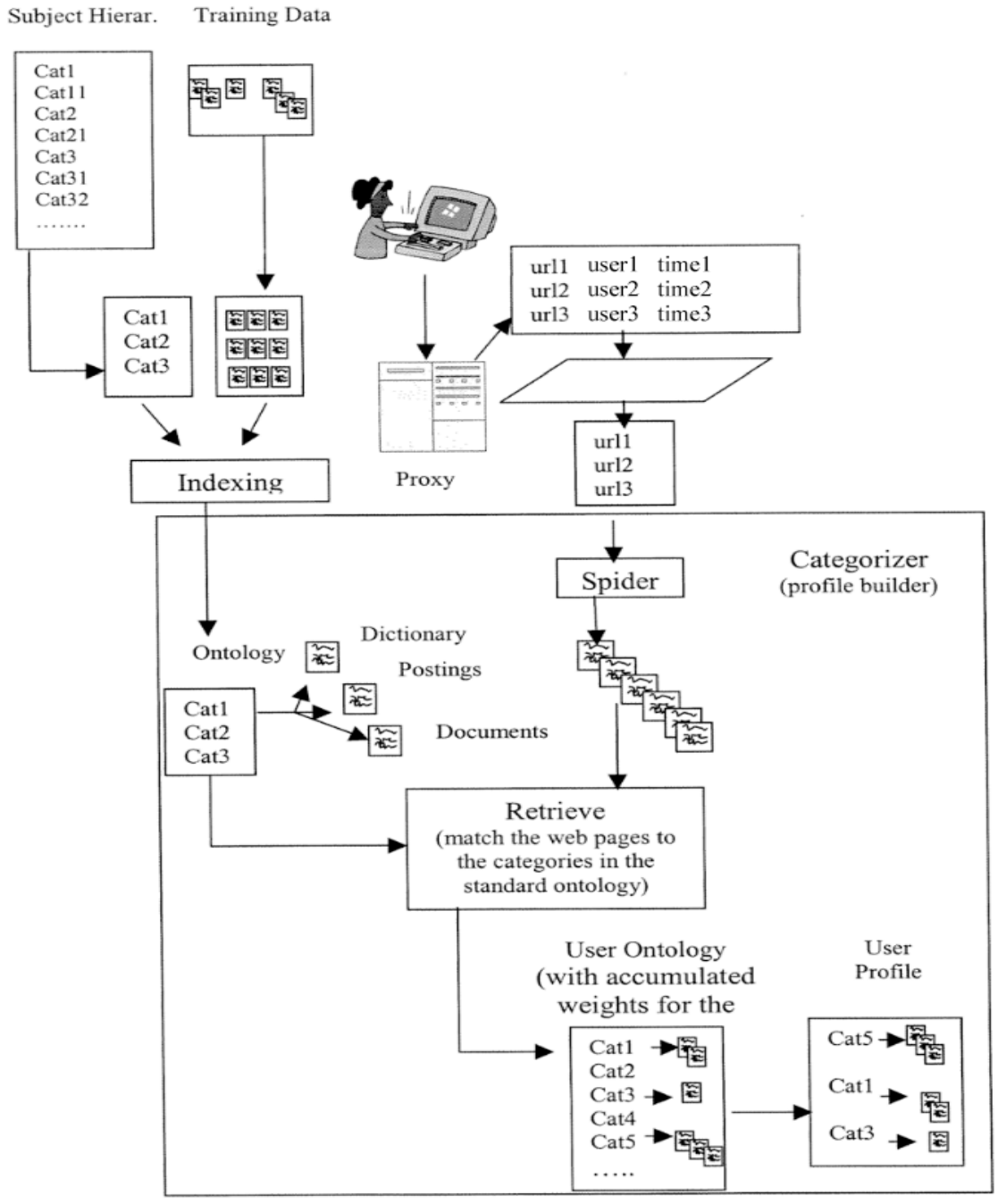
Profile creation and maintenance begins after the user installs a very simple extension to their web browser. For every webpage the user visits after it is installed, the browser extension relays the webpage's URL and the user's unique identification number to a script residing on a remote server. The server script then inserts that information into a database; another program running on the server periodically checks the database for new entries. When a new entry is discovered, the program uses the open-source Wget spider from the GNU Foundation[4] to retrieve the webpage document. The document undergoes preprocessing to remove non-content information like HTML tags and then lexical analysis and indexing to represent the document as a vector in the training data index's N-dimensional space. All documents in the training data index that are similar to the new document are retrieved and ranked by an angle similarity metric, and a "k Nearest Neighbors" (kNN) algorithm uses those vectors to classify the new document and assign it a weight in its given category. Finally, this information is added to the user's profile, which is stored in a database on the remote server.

[4] http://www.gnu.org/s/wget/

B. System Architecture

Figure 3.1 [Trajkova2003] below illustrates the process of obtaining an ontology, indexing a collection of training documents which have been classified into that ontology, and building user profiles out of classified webpages.

Figure 3.1: The process of building a user profile



With the exception of the browser extension, the entire system resides on a remote server. The browser extension is written using in Javascript and utilizes Google Chrome's extension APIs to access the user's browsing activity and to communicate that information to the server. The server-side system is composed of a collection of training documents sorted into a file structure mirroring the hierarchical organization of the chosen ontology, an inverted index of those documents, a database to store user profiles and unclassified webpage URLs, the Wget spider program for retrieving webpages, and a webpage classifier program.

The program which generates the inverted index of the training documents is written in C++ because of the language's inherent runtime efficiency, object-oriented paradigm, extensive code libraries, and online support community. The indexer utilizes the Flex[5] lexical analysis library from the GNU Foundation for all preprocessing and lexical analysis computations. Flex enables the indexer to specify regular expressions for each type of token it is interested in processing and to process each kind of token separately, which is how the indexer discards HTML tags, removes stop words, and stems the remaining tokens.

Because the dictionary and postings files generated by the indexer are relatively small (approximately 10GB total), even for the millions of training documents necessary to train the document classifier, they are stored as simple text files on the server. This allows them to be easily and efficiently read by the classifier program – the dictionary file is written in hashed order (including blank lines), so retrieval of the dictionary entry for any given token can be accomplished in nearly constant time by hashing the token

[5] http://flex.sourceforge.net/

and retrieving the corresponding line in the dictionary – and to be easily transferrable and functional between systems if necessary.

The system only requires two database tables to function fully. The first is a very simple table which stores the user ID numbers and webpage URLs sent to it by the browser extension. The other is the table for storing user profiles, which is also a relatively simple table consisting of fields for the user IDs, category IDs, and category weights. The user profile table is indexed by the user ID field to speed up retrieval and insertion operations. MySQL was the database management system chosen to store the databases because of its ease-of-use, cross-platform portability, and the online support community that almost always accompanies an open-source project of such size.

The webpage classifier program is also written in C++ and utilizes the Flex library to perform the same preprocessing and lexical analysis as the indexer. The classifier takes an HTML document and an inverted index as input, and uses a kNN algorithm to output a classification for the document. This process is examined in more detail in section F of this chapter.

Two trivial PHP scripts tie the disparate modules together and enable them to act as a unified system. One script is responsible for receiving user IDs and webpage URLs and for inserting them into the database of unclassified webpages. Another is responsible for checking that database for new entries, invoking Wget to retrieve and temporarily store any new webpages, invoking the classifier program on those stored webpages, and updating the user profile database according to the output of the classifier.

C. Introduction to Ontologies

An ontology is defined as "a specification of a conceptualization" which defines "content-specific agreements" on vocabulary, sharing, and reuse of knowledge [Gruber93_1, Gruber 93_2]. In lay terms, an ontology is a collection of categories for a given field of study; this collection specifies how concepts in that field can be grouped, related to each other via a hierarchy, and subdivided according to their similarities and differences.

Ontologies are used in philosophical pursuits, such as the study of metaphysics and epistemology, and are also used to facilitate precise communication between data systems in fields of study that make use of similar ambiguous terms, such as biology and medicine. On the Internet, ontologies have been used to improve both search and browsing. Websites that present collections of documents, such as the *Yahoo!* Directory[6], use hierarchical ontologies to allow users to quickly and intuitively browse for categories containing documents for the subject(s) in which they are interested. Specialized search engines – such as the CiteSeerX engine which searches computer science research documents that have been classified with the Association of Computing Machinery's Computing Classification System (CCS)[7] – use ontology tags in document metadata to further improve search results by searching based not just on the content of the document, but by also considering the concept category to which the document belongs.

Autonomous profile generation systems classify webpages using various machine learning algorithms to identify the user's interests. These systems encounter the so-called "cold-start problem" [Middleton2002]; namely, they require an initial training/learning

[6] http://dir.yahoo.com/
[7] http://www.acm.org/about/class/

phase to collect data on the user's behavior and to build the user's profile with that data. During this phase, these systems perform poorly due to the lack of data and its incomplete nature. Ontologies can be used to remedy this issue by *a priori* defining categories and their relationships to one another and by providing examples of documents that have been classified into each category for use in training a classifier.

While many ontologies like the CCS exist for specialized fields, it is (understandably) much more difficult to obtain one that specifies a category hierarchy and provides training documents for all categories of content on the Internet. There are only a few well-maintained, comprehensive Internet directories and even fewer that allow the free licensing of their ontology and documents. For this thesis, the ontology and documents from the Open Directory Project[8] (ODP) were used. The ODP, which bills itself as the "definitive catalog of the web,"[9] is a human-edited directory in which a category hierarchy is manually defined and documents for each category are submitted and approved by human editors. It is generally better maintained than similar projects and can be freely licensed to anyone.

The ODP subject hierarchy contains approximately 24,000 categories and millions of documents that have been manually sorted into the ODP's ontology hierarchy [Trajkova2003]. Each category is identified by a unique ID number, a locator number that specifies the location of the category in the hierarchy, and a string that gives the name of the category and its parents in a manner similar to a file path.

[8] http://www.dmoz.org
[9] http://www.dmoz.org/docs/en/about.html

Figure 3.2: ODP hierarchy example

```
1 000000000000000000 1 Top
2 001000000000000000 2 Top/Arts
27 001001000000000000 27 Top/Arts/Music
1033 001001001000000000 1033 Top/Arts/Music/Styles
1081 001001001001000000 1081 Top/Arts/Music/Styles/Country
133269 001001001002000000 133269 Top/Arts/Music/Styles/Holiday
…
 69065 010004055000000000 69065 Top/Regional/Africa/Regions
69159 010004055001000000 69159 Top/Regional/Africa/Regions/Central_Africa
69160 010004055002000000 69160 Top/Regional/Africa/Regions/East_Africa
69161 010004055003000000 69161 Top/Regional/Africa/Regions/Horn_of_Africa
69162 010004055004000000 69162 Top/Regional/Africa/Regions/Southern_Africa
80559 010004055005000000 80559 Top/Regional/Africa/Regions/African_Islands
328827 010004055006000000 328827 Top/Regional/Africa/Regions/West_Africa
328828 010004055007000000 328828 Top/Regional/Africa/Regions/North_Africa
…
```

D. Training the Classifier

Once a file describing the ODP's category hierarchy was obtained and the documents assigned to each category were downloaded and stored in a directory structure mirroring the ODP's hierarchy, training the webpage classifier program was relatively straightforward. Because the kNN algorithm used to assign categories to new webpages operates in the same N-dimensional vector space that a basic search engine for the ODP would use, creating a standard inverted index for the ODP documents was the only process necessary to train the classifier. The process of creating an inverted index, which is composed of a dictionary of unique terms in the collection and their relative weights in the collection, and a file of postings that map a term to one or more documents and its relative weight in each document, is as follows:

A program that traverses directory structures and records information about them was written. This program traversed the directory hierarchy containing the ODP files and

recorded in a temporary text file the paths to every ODP document of a predefined size in every directory containing a predefined number of documents – for this thesis, the file size requirement was one kilobyte and the directory size requirement was five documents – and assigned each document an ID number. This text file was then passed to a program for generating the inverted index.

Each document on the list was processed and an index was created using a hybrid in-memory/file-based indexing algorithm, in the following way:

In the preprocessing step, extraneous text in each document (such as HTML tags and metadata) was removed; stop words such as "the" were deleted because they are extremely common and therefore provide no information to distinguish one document from another; and the remaining text was stemmed to remove suffixes such as "-ed" or "-ing." The remaining content was then lexically analyzed, and each unique token (word) was added to a hash table that recorded the number of occurrences for each word. Once every term in the document had been added to the local hash table, a temporary postings file listing each unique token in the document, its relative frequency, and the current document's ID was generated. The document hash table was then merged with a collection hash table that recorded each unique token in the collection and the number of documents in which it occurred.

After every document was processed in this manner, a postings file for the entire collection was generated by concatenating the postings file for each document and sorting the resulting file by alphabetically by token using the highly-efficient Linux *sort* command. Because it was programmatically ensured that each line in the individual postings files was the same length, an index number representing the start of each token's

postings in the postings file could be assigned to each token in the collection hash table, and the dictionary could be written to a text file in hashed order, including blank lines for empty buckets in the hash table. This made the dictionary file much larger than it would have been were the blank lines excluded, but allowed for tokens to be retrieved in nearly constant time by hashing them and retrieving the corresponding line in the dictionary file.

Figure 3.3: Example dictionary file

```
…
annulled                    4     3503795

soundcast                   8     62107519

powerkite                   14    50493456

annuler                     6     3503799
…
```

Figure 3.4: Example postings file

```
…
278572  71.794
243482  12.742
243493  14.807
84653   4.8438
84653   4.8438
101114  9.2789
…
```

E. Collecting Data

After the ODP documents were indexed, in effect training the classifier, the next step in generating user profiles was to collect data about the interests of each user. Working under the general assumption that users view webpages on topics they are interested in more often than they view webpages on topics that they are not interested in, the users' real-time Internet browsing activity was used as the data for generating each user's profile.

This data is obtained via a very simple browser extension; currently, only an extension for Google's Chrome browser has been written, but extensions for other browsers are similarly easy to program. Once downloaded and installed, the extension runs in the background. Every time the user visits a webpage, the extension retrieves the URL of the webpage and calls a PHP script residing on a remote server. This script inserts a record containing the user's ID number and the URL of the visited webpage into a database and leaves the retrieval and classification of the webpage, as well as updating the user's profile with the new data, to other programs running on the server. This very simple process is all that is needed for the system to autonomously collect the data necessary to create a user profile.

F. Classifying Webpages and Updating a Profile

The final step in generating a user profile is to classify the webpages the user visits, and to accumulate that information over time. When the webpage classifier program is given a document to classify, it is treated as a query. It undergoes the same preprocessing, stop word removal, stemming, and lexical analysis as was performed on the ODP documents to generate the training data index. Next, the dictionary and postings

entries for every remaining token from the query document is retrieved from the training data index, and an accumulator keeps track of the weight of each query term in each ODP document in which it appears. This weight, denoted w_{ij}, is calculated according to the following formula:

$$w_{ij} = tf_{ij} \times idf_{ij}$$

where tf_{ij} is the normalized frequency of term t_i in document d_j,

$$tf_{ij} = \frac{\#\, occurences\ of\ \ t_i\ in\ d_j}{\#\ of\ \ terms\ in\ d_j},$$

and idf_i is the inverse document frequency of t_i in the entire collection,

$$idf_i = \log\left(\frac{\#\, of\ documents\ in\ \ the\ collection}{\#\ of\ \ documents\ containing\, t_i}\right).$$

Thus each document in the ODP is represented as a vector of token weights in the N-dimensional vector space, where N is the number of unique tokens in the query document. Tokens that appear in the query but do not appear in a given document are (implicitly) represented by a zero in the document's vector, and training documents that do not share any tokens with the query are ignored altogether.

Once every w_{ij} has been calculated, the unit-normal weight of each term in each document is calculated according to the formula:

$$W_{ij} = \frac{w_{ij}}{length\ of\ d_j} = \frac{w_{ij}}{\sqrt{\sum_{t_k \in d_j} w_{kj}}}.$$

This is done to increase computational efficiency when computing the similarity score between a given document and the query, which is simply measured as the cosine of the angle between the document vector and the query vector, Θ, as follows:

$$similarity(d_j,\ q) = \cos\Theta = \frac{d_j \bullet q}{\| d_j \| \times \| q \|} = d_j \bullet q = \sum_{i=1}^{N} W_{ij} \times Q_i ,$$

where q is the N-dimensional query vector and Q_i is the normalized weight of t_i in q,

$$Q_i = \frac{\#\ occurences\ of\ \ t_i\ in\ query}{\#\ terms\ in\ query} .$$

After the postings for each file are retrieved and the similarity score between the query document and each document in the ODP is computed, a simple kNN algorithm assigns a category to the query document. The kNN algorithm was chosen over others such as a Rocchio or naïve Bayes algorithm because it is easiest to implement on top of existing code for generating inverted indexes, because it is computationally efficient after the training phase since it is a non-probabilistic algorithm, and because it is best for classifying new documents in a document vector space in which the boundaries between categories may be non-linear or poorly defined or in which categories may be irregularly shaped in the space's N dimensions [Manning2008]. The kNN algorithm classifies documents by having the k (for some predefined positive integer k) documents with the highest similarity scores "vote" on what the category of the query document should be; each vote is weighted according to its corresponding document's similarity score, and the category with the highest vote score is assigned to the query document.

Finally, the weight value for the winning category is increased by the category's vote score in the user's profile, which is stored in a simple MySQL database. At this point, the classifier program has successfully updated the user's profile according to new data.

IV. EVALUATION

Figure 4.1: Example profile showing categories' names and weights

Category	Weight
Top/World	8
Top/Business/Chemicals/Catalysts_and_Adsorbents	146776
Top/Arts/Design	96696
Top/Arts/Music/Bands_and_Artists/H	177890
Top/Shopping/Crafts/Nature	18130.8
Top/Shopping/Clothing/Footwear	110532
Top/Shopping/Photography/Used_Equipment	121854
Top/Arts/Illustration/Illustrator_Portfolios/K	41793.3
Top/Arts/Music/Music_Videos/Streaming	45857.4
Top/Reference/Knowledge_Management/Information_Overload/Relevance_Filtering	407.09
Top/Recreation/Birding/Mailing_Lists	25280.4
Top/Arts/Comics/Comic_Strips_and_Panels/V	13106.7
Top/Science/Math/Education/Homework_Help	9553.38
Top/World/Telugu/Vinodam	11744.5
Top/Society/Gay,_Lesbian,_and_Bisexual/Arts_and_Entertainment/Comics	158307
Top/Society/Religion_and_Spirituality/Fictional/Jediism	180947
Top/Reference/Education/Instructional_Technology	5835.6
Top/Business/Electronics_and_Electrical/Electromechanical	3276.72
Top/Regional/Middle_East/Business_and_Economy/Financial_Services	7413.15
Top/Shopping/Ethnic_and_Regional/Caribbean	60078
Top/Science/Environment/Sustainability	164214
Top/Arts/Writers_Resources/Poetry	20686.8
Top/Regional/Asia/Hong_Kong/Recreation_and_Sports	2728.6
Top/Society/Law/Services/Law_Practice_Support/Marketing	1213.1
Top/Sports/Cycling/BMX	2762
Top/Business/Business_Services/Customer_Management/Data_Management	18781.2
Top/Society/Ethnicity/African/African-British	9865.1

Overall, the implemented system worked well. An inverted index of the entire ODP training data collection was generated in a matter of hours on a personal computer and transferred via SFTP to the University of Arkansas Computer Science and Computer

Engineering Department's server, named Turing [10]. This server functioned as the nexus for the system; the database housing user profiles and unclassified webpage URLs resided on Turing, the browser extension communicated with scripts on Turing, and webpage document retrieval (using Wget) and classification were performed on the server. This effectively offloaded all computations and data storage to the "cloud," which in future versions of WebTailor will greatly increase the accessibility and usability of the users' profiles for use in the personalization systems of third-party websites.

The webpage classifier worked generally well, in some cases shockingly so. This resulted in fairly accurate, relevant user interest profiles. Figure 4.1 shows a small example profile generated by the system. Even though it was built from only a few dozen webpage views, this profile had a wide range of categories, some of which were four or five levels deep, indicating a very specific classification. For instance, when pages from the website for the online shoe retailer Zappos[11] were visited the system appropriately classified them into the "Top/Shopping/Clothing/Footwear" category, and that category achieved a high weight relative to other categories in the profile. Another interesting example of a classification representing the interests of the user was the classification of a page from ThinkGeek[12] showing a Star Wars themed bathrobe into the category "Top/Society/Religion_and_Sprituality/Fictional/Jediism." On the surface this might seem to be an odd classification, since ThinkGeek is an online retailer and the webpage was for a product, but this category likely reflects the users' interests better than a classification of something like "Top/Shopping/Clothes." Someone who is willing to

[10] http://turing.csce.uark.edu
[11] http://www.zappos.com
[12] http://www.thinkgeek.com/tshirts-apparel/miscellaneous/de79/

consider paying nearly $100 for a terrycloth Jedi Knight bathrobe is likely doing so because they are interested in Star Wars, not because they like bathrobes.

However, there was also a significant number of highly questionable webpage classifications. For instance, the homepage of CNN[13] was classified with a high weight as "Top/Society/Gay_Lesbian_and_Bisexual/Arts_and_Entertainment/Comics." A few likely hypotheses may explain this behavior. First is that during the training phase, the indexer program inadvertently ignored valid content in some documents due to malformed HTML, and thus the N-dimensional vectors representing those documents in the index were inaccurate and lead to mistaken classifications of new webpages. The other hypothesis is similar to the first: the indexer may have inadvertently indexed non-content data in some documents, such as Javascript, PHP, or other programmatic text, adding non-zero weighted entries to those documents' vectors and detrimentally affecting the classification of new documents.

Such aberrations, though undesirable, do not significantly affect the performance of the system as a whole because it is the aggregate performance of the system over time that builds the user profiles. Though there may be several entries in any given user's profile that are the results of misclassification, over time they will become vastly outweighed by the other entries in the user's profile and therefore become virtually irrelevant.

From the user's perspective, WebTailor works well and invisibly. Installing the system's browser extension is a simple process requiring only a few clicks. After that, the user is completely free to go about their normal browsing habits because the system

[13] http://www.cnn.com

no longer requires their active input. Their privacy concerns are also easily addressed; disabling the extension is as simple as un-checking a box in the browser's preferences (as shown in Figure 4.2), and by default the extension is not enabled and does not collect data when the user is browsing using so-called "private" or "incognito" mode. User functionality is currently limited to viewing their profile in a simple table listing of categories and weights, but future improvements could easily display this information in a more presentable form, like an expandable tree, and provide other functionality like profile pruning. There is a small lag between the user viewing a webpage and having their profile updated to reflect the classification of that webpage, but it is small enough for its effect on the overall user experience to be negligible.

Figure 4.2: Disabling the browser extension

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

This thesis focuses on building accurate, long-term user profiles of weighted categories from a predefined concept ontology and pre-classified training documents. The resulting system shows that such profiles can be built autonomously by analyzing the contents of the users' browsing activity, that those profile can be stored and all calculations necessary to create and maintain them can be performed on a remote "cloud" server, enabling them to travel with their respective users and providing the means for profiles to be accessed by approved third parties who wish to improve their own website personalization systems.

Though generally accurate, the webpage classifier still has a significant rate of mistaken classifications, and so the user profiles generated by the system still contain incorrect or irrelevant concepts. So, significant improvements can be made to the existing system, and additional features are also necessary before the system can become truly usable in the pursuit of enabling widespread website personalization that is accurate, comprehensive, relevant, and up-to-date.

B. Future Work

Significant improvement can be made to the system in two areas: data and functionality. The data that the system uses falls into the two categories of training data and user data, and the functionality improvements fall into the general categories of user interactivity and profile accessibility.

Because human editors compile the ODP training data collection manually, it is small and incomplete compared to the ideal theoretical collection that has ample data for

every category. Although the collection is growing over time due to the efforts of its curators, it is doing so at the relatively slow pace necessitated by manual, non-automatic classification. One possible approach for growing the collection much more rapidly would be to couple an Internet crawler program with a document classifier similar to the one used in the system presented in this thesis and have the ODP project's human editors approve or reject the program's automatic classifications. The crawler program would be continuously sending the system new webpages to classify at a rate much higher than is accomplishable by the current manual submission and classification procedure, and the accuracy of those classifications would be ensured by the human editors. In this way, the ODP collection would become much more complete, at least for use in webpage classification systems (although this is not the chief aim of the ODP project).

The programs that the WebTailor system uses to generate user data can also be improved. Possible incorrectness in the preprocessing and lexical analysis phases of the inverted index generation program – discarding relevant content or analyzing non-content data – as well as a low tolerance for malformed HTML lead to the classifying of some documents into highly questionable categories. These are technical challenges that can be overcome by more thoroughly analyzing the output of the indexer program for individual documents and correcting the preprocessing and lexical analysis code accordingly. The indexer program's code was mostly written during an undergraduate course on information retrieval, so perhaps it would even be prudent to completely discard the current indexer and replace it with a much more mature one in future versions of WebTailor.

User data can also be improved by modifying it to consider the results of previous research done in the field of long-term, ontology-based user profiles. For instance, research conducted by J. Trajkova indicates that user interests are better represented by profiles in which the categories are ranked by the number of documents the user has viewed of any given category, rather than the accumulative weights of the categories [Trajkova2003]. Trajkova's research also indicates that modifying the depth of categories represented in the user profiles can improve their accuracy; specific categories with low relative weights should be aggregated into their more general parents, while general categories with high weights should be broken up into their child categories.

Another way to improve the user profiles for use in website personalization would be to include a short-term component in the profiles. Some personalization use cases, such as the previously referenced case of a user shopping for holiday gifts, could benefit greatly from short-term data on the user's browsing activity. Including a separate, short-term profile with the user's persistent, long-term profile would enable more relevant personalization in systems that require a time-sensitive picture of the user's interests.

Before third-party personalization systems can use the profiles generated by WebTailor, two new functionalities must be added to the system. First, a web API must be written which allows third party websites to request access to a specific user's profile, and that profile data must be returned in a manner that is easily understandable and parseable. Secondly, users must be able to easily approve or deny third parties access to their profile data. This should be done in order to alleviate privacy concerns; users would be unlikely to use a profile system that made their data available to anyone. For this and other practical, ethical, and legal reasons, permission should be explicitly granted by the

user for every third party seeking access to a their profile prior to the system granting that access.

Other possible improvements to the functionality of the system include optional relevance feedback from users on each category in their respective profiles or possibly allowing the user to manually prune their profile of unwanted categories. In order to attract users and to demonstrate the benefits of using the WebTailor system to third parties, a personalized search engine, content recommendation service, social network, or other Internet service that utilizes the profiles generated by WebTailor to personalize itself could also be created in future work.

BIBLIOGRAPHY

1. [Chen97] Chen L., Sycara K. WebMate: A Personal Agent for Browsing and Searching. *In Proceedings of the 2nd International Conference on Autonomous Agents. AGENTS '98, ACM*, p. 132 – 139, 1998

2. [Gauch2003] Gauch S., Madrid J., Induri S., Ravindran D., Chadlavada. S. KeyConcept: A Conceptual Search Engine. *Information and Telecommunication Technology Center, Technical Report: ITTC-FY2004-TR-8646-37*, University of Kansas, Lawrence, KS, USA, 2003

3. [Gruber93_1] Gruber T. A translation approach to portable ontologies. *Knowledge Acquistion*, 5(2):199-220, 1993

4. [Gruber 93_2] Gruber T. Toward principles for the design of ontologies used for knowledge sharing. *Presented at the Padaua workshop on Formal Ontology,* March 1993

5. [Heflin99] Heflin J., Hendler J, Luke S. SHOE: A Knowledge Representation Language for Internet Applications. *Technical Report CS-TR-4078 (UMIACS TR-99-71),* Dept. of Computer Science, University of Maryland at College Park, 1999

6. [Kurki99] Kurki T., Jokela S., Sulonen R., Turpeinen M. Agents in delivering personalized content based on semantic metadata. *In Proceedings of the 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cybersapce,* Stanford, USA, 1999, 1999, p. 84-93. As cited in [Pretschner99]

7. [Lieberman95] Lieberman H. Letizia: An Agent That Assists Web Browsing. *In Proceedings of the International Joint Conference on Artificial Intelligence,* Montreal, CA, 1995

8. [Luke97] Luke S., Spector L., Rager D., Hendler J. Ontology-Based Web Agents. *In Proceedings of the First International Conference on Autonomous Agents, W.L. Johnson (ed.), New York Association for Computing Machinery,* p. 5966, 1997

9. [Manning2008] Manning C., Raghavan P., Shütze H. *Introduction to*

Information Retrieval. Cambridge University Press, New York City, New York, 2008

10. [Middleton2001] Middleton S, De Roure D., Shadbolt N. Capturing knowledge of user preferences: ontologies in recommender systems. *In Proceedings of the 1st International Conference on Knowledge Capture (K-Cap2001),* Victoria, BC, Canada, 2001

11. [Middleton2002] Middleton S., Alani H., Shadbolt N, De Roure D. Exploiting Synergy Between Ontologies and Recommender Systems. *ACM, Semantic Web Workshop 2002 At the Eleventh International World Wide Web Conference* Hawaii, USA, 2002

12. [Mizzaro2002] Mizzaro S., Tasso C. Ephemeral and persistent personalization in adaptive information access to scholarly publications on the web. *Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference,* 2002

13. [Mladenic98] Mladenic D. Personal WebWatcher: Design and Implementation. *Techincal Report IJS-DP-7472, J. Stefan Institute, Department for Intelligent Systems,* Ljubljana, 1998

14. [Pretschner99] Pretschner A. Ontology based personalizd search. *Master's thesis,* University of Kansas, Lawrence, KS, USA, 1999

15. [Tanudjaja2001] Tanudjaja F., Mui L. Persona: A contextualized and personalized web search. *In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02),* Volume 3 p. 53, Big Island, Hawaii, 2002

16. [Trajkova2003] Trajkova J.. *Improving Ontology-Based User Profiles*. *Masters Thesis,* University of Kansas, Lawrence, KS, USA, 2003

17. [Zhu99] Zhu X., Gauch S., Gerhard L., Kral N., Pretschner A. Ontology Base Web Site Mapping for Information Exploration. *In Proceedings of the Eighth International Conference on Information Knowledge Management.* Kansas City, MO, USA. p. 188-194, 1999