A SOFTWARE AGENT FOR CONTENT BASED RECOMMENDATIONS FOR THE WWW

Gulden Uchyigit
University of Brighton, Brighton, U.K.

Keywords: Recommender systems, web browsing, user modelling.

Abstract: The evolution of the WWW has led to an explosion of information and consequentially a significant increase on usage. This avalanche effect has resulted in such uncertain environment in which we find it difficult to clarify what we want, or to find what we need. In this paper we introduce RecSys which aims to confront the problem by developing a software agent which intelligently learns users interests, and hence makes recommendations of resources on WWW based on the user's profile. The system employs multiple TFIDF vectors to represent various domains of user's interests. It continuously and progressively learns users profile from both implicit and explicit feedback. This is achieved by extraction and refinement of featured keywords within the learning examples. Several heuristics were also adapted to improve the overall performance of the system.

1 INTRODUCTION

The evolution of the WWW has led us an explosion of information and consequentially a significant increase in usage. This avalanche effect has resulted in such an uncertain environment in which we find it difficult to clarify what we want, or to find what we need. What's more, the information sources have high noise, i.e contains irrelevant content to a particular user's interest, hence one could never predict what he/she gets is exactly what he/she expected. There is therefore a need to find a way to improve such a situation, so that a user can always, to some extent, be able to guarantee that the next recommendation is the one which is right for him/her.

Recent research shows that there has been a great deal of work on how artificial intelligence can help solve this problem. Depending on the different focuses of interest, these ideas exist in a great variety of forms, these include personalised search engines and intelligent recommender systems. The latter can be further divided into different categories, namely collaborative recommender systems, content-based recommender systems, and a hybrid recommender systems. In this paper we focus on the area of content-based recommender systems. In particular we present a software agent system which is able to provide the user recommendations whilst the user is browsing the WWW.

2 BACKGROUND

Content-based recommendations and collaborative based recommendations are two most popular paradigms in recommending items. The former extracts information from the documents that have already been evaluated by the user in order to obtain new related items, while the latter recommends items to the user based on the evaluation by users within a similar category. With content-based recommendation systems recommendations are made for a single user based solely on a profile set up by analysing the content of documents as the user has rated in the past. Table 1 shows a comparison of different content-based systems. The main disadvantage of these types of systems is that they over specialise to a certain area and only recommend items which the user has seen in the past. Collaborative based systems recommend items to its user's by basing its recommendations on how other user's have rated similar items. One of the main disadvantages of these systems is the so called early rater problem, where new items can not be

Uchyigit G. (2008).
A SOFTWARE AGENT FOR CONTENT BASED RECOMMENDATIONS FOR THE WWW.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - SAIC*, pages 178-183
DOI: 10.5220/0001720501780183

recommended if they have not been previously rated by other users.

Profiles are important part to represent user interests. There are several methods which can be used to represent the user profiles. One well known method is the vector space model. *Vector space representation* (Baeza-Yates and Ribeiro-Neto, 1999) method has been successfully employed in several content-based systems (Chen and Sycara, 1998), (Mladenic, 1996), (Lang, 1995), (Moukas, 1996), (Liberman, 1995), (Armstrong et al., 1995)). Each element of the vector is composed of a word/concept and an associated weight. A well known technique for determining weights is a technique known as TF-IDF. In this approach documents are converted into a vector space by representing them as a set of scores.

Consider a collection of documents C:

$$\mathbf{C}= \begin{bmatrix} & T_1 & T_2 & \ldots\ldots\ldots & T_t \\ D_1 & w_{11} & w_{21} & \ldots\ldots\ldots & w_{t1} \\ D_2 & w_{12} & w_{22} & \ldots\ldots\ldots & w_{t2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ D_n & w_{1n} & w_{2n} & \ldots\ldots\ldots & w_{tn} \end{bmatrix}$$

This term document matrix contains a set of documents $D_{1..n}$ each contains a set of words $t_{1..n}$. Each entry in the matrix, w_{td}, corresponds to the associated document d. Terms which appear in lots of different documents are less indicative of the topic of the document, these are the so called stop-words and are words such as “the”, “and” etc. The TF-IDF factor assigns weights to words depending on their relevance to the identification of the topic. The TF-IDF is evaluated using the formula below:

$$w_{ij} = tf_{ij} idf_i = tf_{ij}(\frac{N}{df_i})$$

where, tf_{ij} is the term frequency across the entire corpus, it is normalised value of f_{ij}, frequency of term i in document j. N is the total number of documents with in the collection C. idf_i is the inverse of df_i document frequency of term T_i. This definition implies that a term occurring frequently in a single document is given high weight.

Content based recommender systems can make recommendations using direct comparison between the user’s profile and candidate items. Hence it is worth discussing some data matching techniques here. Typical techniques include standard keyword matching, nearest neighbours, classification and cosine similarity.

Cosine similarity is the normalized form of inner product, which can be used as measure of similarity between two TF-IDF vectors:

$$sim(d_j, q) = d_j \bullet q = \sum_{i=1}^{t} w_{ij} w_{iq}$$

Here d_j is the vector representation of document D_j and q is a vector to be compared.

User feedback is an important factor with respect to recommender systems. As the user continues to use the system and provide feedback the system improves in the recommendations it provides. Feedback can be provided explicitly by the user providing direct ratings to the items recommended by the system. It can also take the form of implicit feedback by the system observing the browsing behaviour of the user and making an assumption as to which items the user is interested in. The advantage of explicit feedback is that the feedback comes directly from the user and is therefore considered to be more accurate. With implicit feedback there is a degree of uncertainty that the feedback is not very accurate. The advantage of implicit feedback is that the reduced burden on the user to rate every single item the user has rated. Of course a system can combine both forms of feedback to have hybrid feedback.

Table 1: Comparison of different content-based systems.

Agent	*Goal*	*Profile Rep.*	*Prof. Match*	*Feedback*
PWW (Armstrong et al., 1995)	Web brows.	Prob. Feat. Vect	NB	Implicit
SIFT (Lang, 1995)	News filt.	Bool. Feat. Vect	Cos. Sim.	Explicit
WebMate (Chen and Sycara, 1998)	Web brows.	Prob. Feat. Vect.	NB	Explicit
Letizia (Liberman, 1995)	Web brows.	Prob. Feat. Vect.	Cos. Sim	Explicit
Lira (Balabanovic1998)	Web brows.	Prob. Feat. Vect.	Cos. Sim	Explicit
Syskill Webert	Web brows.	Prob. Feat. Vect.	NB	Explicit

3 SYSTEM OVERVIEW

The implemented system consists of four main components the User Interface, the

Recommendation Manager, and the Learning Manager (See Figure 1).
On start of the application, the top level GUI component a HTTP Proxy starts which constantly listens to a user defined port on a local host. The learning manager initialises to supervise the learning process. The recommender manager is initialised to make recommendations based on the learning managers outcome.

Whenever a positive example is identified, Http session hands over the relevant information to a session analyser, in which positive learning example is constructed, and hence the example will be thrown into the system's example pool.

The learning manager constantly checks the status of the example pool. It picks up any example from the pool one by one. On receiving of a positive learning example, the learning manager extracts the features of the example, learns it and hence generates a new profile record, which will be then used to update user's main profile. Meanwhile the recommendation manager extracts the information from user's main profile and hence makes recommendations, which will then be shown on the GUI.

Profile Representation. Among a variety of representation techniques, Vector space model was chosen due to its simplicity and efficiency in representing multiple text documents. The user's profile is represented as multiple feature vectors. Each feature vector, corresponds to a single domain of interests, consists of a set of objects in the form of (keyword(stemmed), keyword(original), weight)

Learning. The user's profiles are progressively learned and updated from both implicit feedback and explicit feedback. The system collects implicit feedback from user's browsing activities. Such information, typically is in the form of a web page which is then extracted and stored as a positive learning example and these examples will hence be thrown into the system's example pool, waiting for Learning managers collection.

Upon receiving of every single learning example, the learning manager will perform the following tasks:

Text Processing. Text processing is used to extract features from and example. In particular it will:

1. Fetch pure text content of the current page
2. Fetch pure text content of the pages linked by current page's referrer page.
3. Process text obtained in step 1 by :
    a. Pre-process text
    b. Remove stop words
    c. Stem the words
    d. Assign weights to words using TFIDF
4. Repeat procedures stated in step 3, for every single page obtained in step 2.

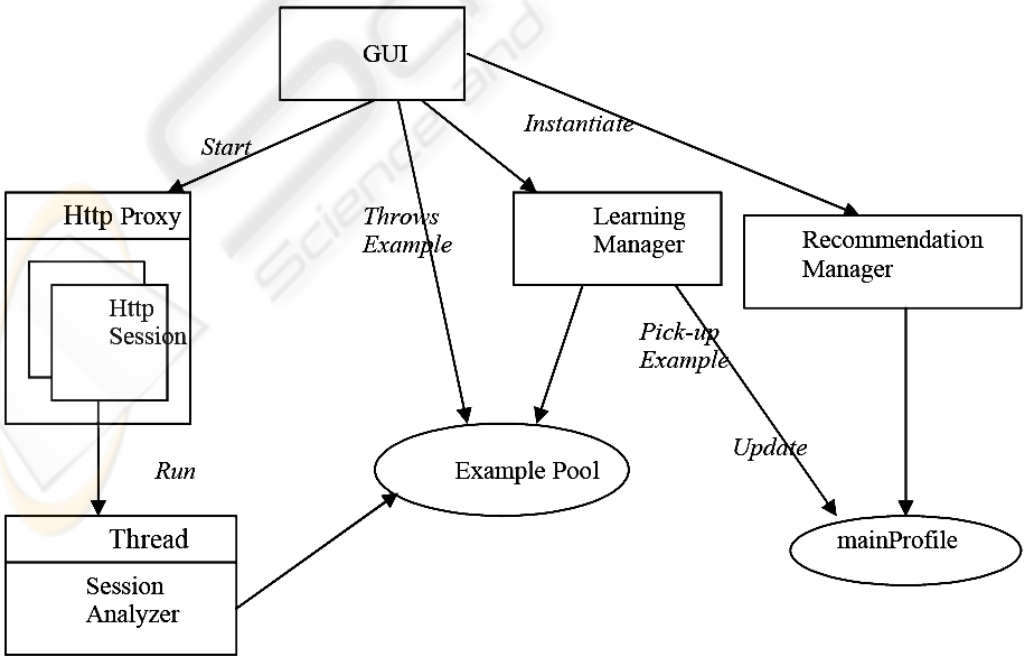

Figure 1: Overview of system Architecture.

Profile Updating. User profiles are incrementally updated from the contents of the pages which the user has seen and found interesting.

Explicit Feedback. The system collects explicit feedback on the user's opinion, for the web pages that have been already recommended to them. Such information, again typically in the form of a web page will be extracted and stored as positive rating example and these examples will be also be placed in the example pool until retrieved by the learning manager.

HTTP is the main protocol of the WWW. In essence, it defines a set of rules describing how data should be formatted and exchanged between servers and browsers. Being an intermediate between the http server and the client side, it is the system's responsibility to recognize the different types of HTTP request the client may send, as well as the various HTTP responses that may be returned by the server. In essence, in order to generate a positive learning example. The system is versatile enough to cope with a wide variety of HTTP headers. It is able to parse the HTTP headers sent between the client and server and extract the relevant information such as url's of the pages of interest and extract the contents of the page which the url is pointing to.

Graphical User Interface. The GUI plays an important role in displaying recommended pages to the user. The user interface is split into two parts: The headline panel and the abstraction panel (see Figure 2).

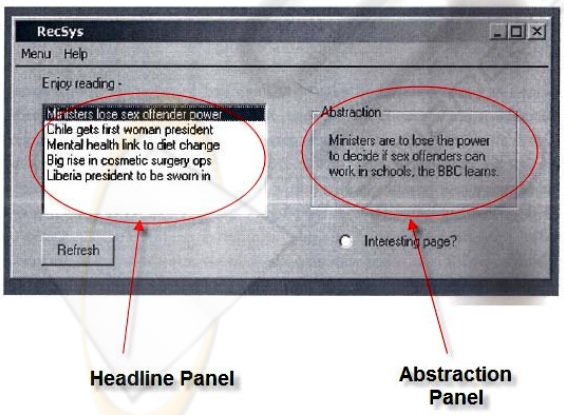

Figure 2: The User Interface.

The headline panel is the component related directly to the underlying behaviour of the system, displaying the recommendations made by the recommending part. The titles of the pages are displayed in the headline panel. To promote interactivity, each recommended item has to respond to certain selection events. On double clicking the page title, the system should start a browser, displaying the content of the selected page. The possibility to refresh the page so that up-to-date recommendations, is possible.

The abstraction panel serves to give more details of the page recommended by the system and also allows a user to give explicit feedback, i.e rate the selected page as interesting, based on the title of the page, the abstraction of the page or even full content of the page. It has been designed that on selection of a recommended item in headline panel, should also correspondingly display the abstraction of the selected page. On clicking of the button "an interesting page?" a window pops up informing the user that his/her opinion has been taken into consideration.

Two other tabs at the top left hand corner of the main page "Menu" and "Help" On clicking Menu the following options can be selected: Settings, save, exit.

The following options may be selected from the settings menu. Profile path allow the user to define the location of his/her profile is located. Configuration path, allows the user to define the port the application should listen to, in performing as a proxy server. Browser allows the user to choose the browser they would like to use to browse the recommendations. Information source, allows the user to specify the information source of recommendations.

The Recommendation Process

System recommends web pages upon user's request, a simple outline of recommendation process is seen below:

1. Query user's profile
2. Pick up a profile record, extract the top N words form the user's profile and send these words to Google.
3. Extract the top M results returned by Google, put them in a collection of web pages denoted by as C.
4. Generate a TF-IDF for every single page within C.
5. Calculate *cosine similarity* between every vector gained in step 4 and the profile, return the top K recommended objects to GUI. K is chosen based on the priority of current profile record.
6. Repeat step 2 to 5 for every single profile record in user's main profile.

Alternatively, with a user specified information source, the recommender system:

1. Extract all pages that could be directed from the information source.
2. Put hem into a collection of web pages, donated as C.
3. Generate TF-IDF vector for every single page within C.
4. For every profile record calculate the cosine similarity between every vector gained in step 3 and profile record and return the one that has the biggest similarity value compared with the profile.

4 TESTING AND EVALUATION

In this section we discuss our testing strategies for the essential components of the system, followed by the evaluation in both qualitative and quantitative ways.

Usability. Usability evaluation is very subjective and as such hard to quantify. As part of the evaluation process, we conducted a questionnaire to gauge strength and weakness in usability, with carefully selected questions. The questions must be quick and easy for a user to complete and also any bias need to be eliminated in designing of questions.

Based on the considerations above we decided to incorporate heuristic evaluation methods as part of our usability evaluations.

Heuristic evaluation is a discount usability engineering method for quick, cheap and easy evaluation of a user interface design. It involves having small set of evaluators examine the interface and judge its compliance with recognized usability principles.

Each user was given a short demonstration following a 15 minute familiarisation session. A questionnaire was then supplied with the following questions.

1. *Visibility of system status*: do you agree that the system keeps you informed of what's going on?
2. *Match between system and the real world*: Do you agree that the RecSys speaks the language familiar to you, rather than system oriented terms?
3. *Consistency and standards*: Do you agree that in accessing RecSys, there is no confusion between different words, situations, or actions means the same thing, all follow platform conventions?
4. *Help and documentation*: Do you find help documentation useful?
5. *Look and Feel*: Do you agree that the application is visually appealing?
6. *Reasonable utilization of the system resource*: Do you agree that you browsing activities are not affected by RecSys?

The users responded to the questions by providing a rating for each question between 1(strongly disagree)-5(strongly agree).

In total there were 30 participants which took part in the experiments. Table below shows the results of our study.

	Question	Score(max 25)
1	Visibility of system status	120
2	Match between system and system and the real world	132
3	Consistency and standards	126
4	Help and documentation	126
5	Look and Feel	90
6	Reasonable utilization of the system resource	120

Testing of the Domain Classification. The accuracy of the algorithms were measured by implementation of a test harness as a replacement for a graphical user interface, to drive the underlying components of the system for test purposes. We allow test cases to be defined in text files for ease of test case generation. We also format the outputs of tests to a result file where it can be post-analysed.

We fed a list of paired urls (url1,url2) to the test harness. The test harness automatically extracts and creates a profile record based on url1 and inserts this record into the user's profile. The test harness calls the underlying methods that deal with similarity calculations and profile classification. It finally outputs a file with statistical results of the test, in which accuracy is analyzed as well as Precision and Recall.

In our results we have found that the system gave an overall accuracy of 80%, with recall 66.67% and 100% precision.

5 CONCLUSIONS AND FUTURE WORK

In this paper we presented an implementation of a software agent, which able to present the user with content-based recommendations whilst the user is browsing the WWW. The system uses both implicit and explicit feedback to learn the user's profile.

The accuracy of the system was evaluated by performing usability testing using a set of users. The learning and classification algorithms were evaluated by implementing a test harness where several positive and negative examples were used to test the system.

In our future work we plan to improve:

- the GUI as suggestions from our user focused study;
- Resource usage awareness: In the current implementation, we deploy only one learning manager to deal with all learning activities. A more sophisticated implementation might detect available system resources and autonomously decide the number of threads it could dispatch.
- Alternative to collect user's implicit feedback: Apart from the current heuristics we use, in making judgement based on http headers it would be better to come up with more ideas such as the time spent on a particular pages, user's activity such as "add to my favourite" are all good examples of implicit implications on user's interests.

REFERENCES

Chen, L. and Sycara, K. (1998). *Webmate: A personal agent for browsing and searching*. In 2nd International Conference on Autonomous Agents, Minneapolis MN USA.

Mladenic, D. (1996). *Personal WebWatcher: design and implementation*. Technical report, Department for Intelligent Systems, J. Stefan Institute, Jamova 39, 11000 Ljiubjana, Slovenia.

Lang, K. (1995). *Newsweeder: Learning to filter netnews*. In 12th International Conference on Machine Learning.

Moukas, A. (1996). *Amalthaea: Information discovery and filtering using a multi-agent evolving ecosystem.* In Proc. 1st Intl. Conf. on the Practical Application of Intelligent Agents and Multi Agent Technology, London.

Liberman, H. (1995). *Letzia: An agent that assists in web browsing.* In Proceedings of the 1995 International Joint Conference on Artificial Intelligence, Montreal, Canada.

Armstrong, R., Freitag, D., Joachims, T., and Mitchel, T. Webwatcher: *A learning apprentice for the world wide web.* In AAAI Spring Synopsium on Information Gathering from Heterogenous, Distributed Environments. (1995)

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Adsison Wesley.

Balabanovic, M. (1998). *Learning to Surf: Multi-agent Systems for Adaptive Web Page Recommendation.* PhD thesis, Department of Computer Science, Stanford University.

Pazzani, M. and Billsus, D. (1997). *Learning and revising user profiles: The identification of interesting web sites*. Machine Learning, 27:313-331.