Exploiting User Context and Preferences for Intelligent Web Search

Carlos I. Chesñevar[†] **Carlos M. Lorenzetti**[‡] **Ana G. Maguitman**[‡]
Fernando M. Sagui[‡] **Guillermo R. Simari**[‡]

[†]Artificial Intelligence Research Group – Department of Computer Science
Universitat de Lleida – C/Jaume II, 69 – E-25001 Lleida, SPAIN – Email: cic@eps.udl.es
[‡] Departamento de Cs. e Ing. de la Computación – Universidad Nacional del Sur
Alem 1253, (8000) Bahía Blanca, ARGENTINA – Email: {cml,agm,fms,grs}@cs.uns.edu.ar

ABSTRACT

Seeking information relevant to a topic of interest has become a common task in our daily activities. However, searching the Web using current technologies still presents many limitations. One of the main limitations is that existing tools for searching the Web restrict user queries to a small number of terms. As a result, a single query may not reflect the user information needs at a sufficient level of detail. In addition, even if longer queries were allowed, the user may not find the right terms to supply appropriate queries, or may not be willing to put the effort required to explicitly describe his or her information needs. Another limitation of today's search tools is that they are not capable of performing qualitative inference on the suggestions they offer. For certain domains, such as news or scientific articles, a good amount of structural information can be usefully exploited to extract meaningful content. This can help sort out the material returned by a search engine and to perform a qualitative analysis to warrant some of the search results. This paper shows how to enhance current search engines capabilities by (1) taking advantage of the user context, and (2) ranking search results based on preferential criteria provided by the user. We describe ongoing research on the use of context-specific terms to refine Web search and on the use of a defeasible argumentation framework to prioritize search results.

Keywords: web search, context, argumentation, user preferences, intelligent aides.

1 INTRODUCTION

The World Wide Web is a huge source of information about basically any topic. An important skill for any user is to know how to find out about a topic of special interest, focusing the search on material that is relevant to the current task. This search activity could be done more effectively if "intelligent mechanisms" for information access and delivery were included as part of the system search tools. In order to reduce the user cognitive overload, task-specific Web search tools need to be adapted to deliver few but highly relevant resources.

To support effective Web search we are developing a family of intelligent aides that monitor the user and search the Web for material related to the user current task and preferences. An intelligent aide that makes relevant information easily accessible can "make the user smarter" or at least more competent in completing many of his or her daily activities. Unfortunately, many support tools have the potential to obstruct the tasks they are designed to support, by providing inadequate information or doing it at the wrong time. As a consequence an important requirement for our tools is to provide relevant material, doing it at the right time, and without causing undue or excessive distraction.

Two elements that can be exploited to enhance Web search are *user context* and *user preferences*. User context reflects the task in which the user is immersed (e.g., [1, 7]). The context may consist of an electronic document the user is editing, Web pages the user has recently visited, etc. User preferences reflect the way in which a user would prioritize search results. The user preferences could be entered explicitly by the user or could be inferred by the system (e.g., by monitoring the user's behavior).

This paper presents techniques for taking advantage of user context and preferences during Web search. Section 2 presents methods for context-based information search based on the dynamic extraction of topic descriptors and discriminators. Section 3 describes how the search process can be further enhanced by applying a defeasible argumentation framework to prioritize search results. Section 4 describes how the pieces are combined into an end-to-end system for context-based search. Finally, in Section 5 we present our conclusions.

2 SUPPORTING CONTEXT-BASED INFORMATION SEARCH

For many computer-mediated tasks, the user context provides a rich set of terms that can be exploited to enhance

Web search. A context-based search tool can be embedded in different kinds of computer utilities, such us email systems, browsers and text editors. In order to reflect the user context, incremental search approaches are needed.[1]

The first query terms generated for a Web search may not provide the definitive results. However, comparing the set of search results to the user task can help to automatically refine subsequent queries. As a first approximation, we assume that documents that are similar to the user context are relevant to the user task (although a different scheme could be adopted [2, 11, 8]). Once the relevance of the retrieved material is estimated, we can proceed to assess the importance of the terms found in the set of search results. This requires a framework for weighting terms based on context.

Substantial experimental evidence supports the effectiveness of using weights to reflect relative term importance for traditional information retrieval (IR) [10]. The main purpose of a term weighting system is the enhancement of retrieval effectiveness. The IR community has investigated the roles of terms as descriptors and discriminators for several decades. The combination of descriptors and discriminators gives rise to schemes for measuring term relevance such as the familiar *term frequency inverse document frequency* (TF-IDF) weighting model [10]. The TF-IDF scheme is a reasonable measure of term importance but is insufficient for the task domain for our research. Searching the Web to support context-based retrieval presents new challenges for formulation of descriptors and discriminators.

Our basic approach is to use descriptors and discriminators automatically extracted from the user current context to guide querying a Web search engine for relevant information. Differently from conventional approaches for querying the Web, search requests are not treated in isolation but in the context of task-specific descriptors and discriminators.

In previous work [9] we have tested the following two hypotheses:

- Good topic descriptors can be found by looking for terms that occur *often* in documents similar to the given topic.
- Good topic discriminators can be found by looking for terms that occur *only* in documents similar to the given topic.

In our proposal, similar documents on the Web play a crucial role in deciding "what terms best describe the topic at hand, and which ones best discriminate it". Previous experiments show that human assessments of term importance in a topic are in good correspondence with the proposed notion of term descriptive power. In addition, we have shown empirically that queries constructed with terms dynamically selected in light of the above notion of term discriminating power result in better precision than the one achieved by the traditional TF-IDF weighting scheme (see [9] for details on these results). The next section summarizes our framework for the dynamic extraction of topic descriptors and discriminators.

[1] Search engines restrict queries to a small number of terms (e.g., the 10-term limit for Google). As a result, a single query cannot reflect extensive contextual information.

2.1 A Framework for the Dynamic Extraction of Topic Descriptors and Discriminators

Given a collection of m documents and n terms we can build a $m \times n$ matrix $\mathbf{H}$, such that $\mathbf{H}[i,j] = k$, where k is the number of occurrences of term t_j in document d_i. We *define discriminating power of a term in a document* as a function $\gamma : \{t_0, \ldots, t_{n-1}\} \times \{d_0, \ldots, d_{m-1}\} \rightarrow [0,1]$:

$$\gamma(t_i, d_j) = \frac{sign(\mathbf{H}[j,i])}{\sqrt{\sum_{k=0}^{m-1} sign(\mathbf{H}[k,i])}}.$$

Analogously, we define *descriptive power of a term in a document* as a function $\lambda : \{d_0, \ldots, d_{m-1}\} \times \{t_0, \ldots, t_{n-1}\} \rightarrow [0,1]$:

$$\lambda(d_i, t_j) = \frac{\mathbf{H}[i,j]}{\sqrt{\sum_{k=0}^{n-1} (\mathbf{H}[i,k])^2}}.$$

These simple notions of document descriptors and discriminators share some insight with standard IR proposals [10]. Another recurrent notion in IR is document similarity. Let $\sigma(d_i, d_j)$ stand for the *similarity measure between documents* d_i and d_j. This measure can be computed in terms of term descriptive power as follows:

$$\sigma(d_i, d_j) = \sum_{k=0}^{n-1} [\lambda(d_i, t_k) \cdot \lambda(d_j, t_k)].$$

We are interested in identifying good topic discriminators to form queries that will result in high precision. Function γ allows discovering terms that are good discriminators of a document, as opposed to good discriminators of the topic of a document. Because our goal is to refine queries to best reflect the topic of the user task, we propose a topic-dependant definition of topic discriminators based on the notion of similarity between documents. As we informally formulated earlier, a term is a good discriminator of a topic if it tends to occur *only* in documents associated with that topic. We define the *discriminating power of a term in the topic of a document* as a function $\Gamma : \{t_0, \ldots, t_{n-1}\} \times \{d_0, \ldots, d_{m-1}\} \rightarrow [0,1]$ calculated as follows:

$$\Gamma(t_i, d_j) = \sum_{\substack{k=0 \\ k \neq j}}^{m-1} [[\gamma(t_i, d_k)]^2 \cdot \sigma(d_k, d_j)].$$

Thus the discriminating power of term t_i in the topic of document d_j is an average of the similarity of d_j to other documents discriminated by t_i. The notion of topic descriptors was informally defined earlier as terms that occur *often* in the context of a topic. We measure *term descriptive power in the topic of a document* as a function $\Lambda : \{d_0, \ldots, d_{m-1}\} \times \{t_0, \ldots, t_{n-1}\} \rightarrow [0, 1]$. If $\sum_{\substack{k=0 \\ k \neq i}}^{m-1} \sigma(d_i, d_k) = 0$ then we set $\Lambda(d_i, t_j) = 0$. Otherwise we compute $\Lambda(d_i, t_j)$ as follows:

$$\Lambda(d_i, t_j) = \frac{\sum_{\substack{k=0 \\ k \neq i}}^{m-1} [\sigma(d_i, d_k) \cdot [\lambda(d_k, t_j)]^2]}{\sum_{\substack{k=0 \\ k \neq i}}^{m-1} \sigma(d_i, d_k)}$$

Descriptive power of a term t_j in the topic of a document d_i is a measure of the quality of t_j as a descriptor of documents similar to d_i.

Guided by the notions of topic descriptors and discriminators, it is possible to reinforce the weights of existing and novel terms. This results in a better representation of the user search context, facilitating query refinement and context-based filtering.

3 EXPLOITING USER PREFERENCES

The previous sections describe methods for context-based Web search, which have the potential to help the user focus on highly relevant material. However, user context may not be sufficient to identify definitive results. For human-generated queries, users frequently decide, based on initial results, to refine their queries based on their own preferences.

Context-based retrieval methods as the ones described above are insensitive to user preference criteria. These methods perform search based on the user task, but do not reflect the user's preferences. Only the terms that appear in the user search context (either explicitly or inferred by the system) are used to describe the user's information needs. In addition, information sources that the user considers reliable cannot be prioritized over those considered unreliable.

For an increasing number of search situations, the key to success is access to high-quality relevant information guided by a simple specification of the information needs and some preference criteria, without excessive distraction. Consider, for example, the case of a journalist investigating certain events and searching for relevant information. As the journalist browses the results returned by a conventional search engine, she will apply some preference criteria to manually select the most valuable results (e.g., those articles published during a specific date range will be preferred over others). Much of the process of selecting such material according to some preference criteria could be effectively automatized. However, a full-spectrum analysis such as the one described requires some form of *qualitative* analysis of the search results based on the user preferences.

As part of our current research we are developing methods that evaluate and ranks search results based on the user's declared preference criteria. User preferences are captured as a set of rules and facts, which can be made explicit in a more intuitive manner than by the use of query special commands. Such set of rules and facts will provide a knowledge base upon which a qualitative analysis of the results returned by a context-based search tool will be performed.

The next section presents an overview of DeLP, a general-purpose argumentation formalism based on logic programming that we have adopted for ranking search results based on user preferences.

3.1 Defeasible Logic Programming: Overview

Defeasible logic programming (DeLP) [5] is a general-purpose defeasible argumentation formalism based on logic programming, intended to model inconsistent and potentially contradictory knowledge.[2] A defeasible logic program is a set $\mathcal{P} = (\Pi, \Delta)$ of Horn-like clauses, where Π and Δ stand for sets of *strict* and *defeasible* knowledge, resp. The set Π of strict knowledge involves *strict rules* of the form $P \leftarrow Q_1, \ldots, Q_k$ and *facts* (strict rules with empty body), and it is assumed to be *non-contradictory*. [3] The set Δ of defeasible knowledge involves *defeasible rules* of the form $P \prec Q_1, \ldots, Q_k$, which stands for "$Q_1, \ldots Q_k$ *provide a tentative reason to believe* P." Strict and defeasible rules in DeLP are defined in terms of *literals* P, Q_1, Q_2, ... A literal is an atom or the strict negation ($\sim$) of an atom.

Deriving literals in DeLP results in the construction of *arguments*. An argument $\mathcal{A}$ for a literal Q (denoted $\langle \mathcal{A}, Q \rangle$) is a (possibly empty) set of ground defeasible rules that together with the set Π provide a SLD proof for a given literal Q, satisfying the additional requirements of *non-contradiction* (*i. e.*, an argument should not involve contradictory information) and *minimality* (*i. e.*, the set of defeasible information used should be minimal). Note that arguments are obtained by the usual query-driven SLD derivation from logic programming, performed by backward chaining on *both* strict and defeasible rules; in this context a negated literal $\sim P$ is treated just as a new predicate name no_P. As a program $\mathcal{P}$ represents incomplete and tentative information, *conflicting* arguments may arise. An argument $\langle \mathcal{B}, R \rangle$ is a *counterargument* for another argument $\langle \mathcal{A}, Q \rangle$ if $\Pi \cup \mathcal{A} \cup \mathcal{B}'$ is a contradictory set, where $\mathcal{B}' \subseteq \mathcal{B}$. Intuitively, this means that both arguments cannot be accepted simultaneously as they their joint acceptance leads to contradictory conclusions. A preference criterion among arguments " $\succeq$ " is used to determine when

[2] For space reasons, we will restrict ourselves to a basic set of definitions and concepts which make this paper self-contained. For more details, see [5, 3].

[3] Contradiction stands for deriving two complementary literals wrt strict negation (P and $\sim P$) or default negation (P and not P).

an argument is a *defeater* for another argument. An argument $\langle \mathcal{B}, R\rangle$ *defeats* another argument $\langle \mathcal{A}, Q\rangle$ if $\langle \mathcal{B}, R\rangle$ is a counterargument for $\langle \mathcal{A}, Q\rangle$ (*i. e.*, $\Pi \cup \mathcal{A} \cup \mathcal{B}'$ is a contradictory set, $\mathcal{B}' \subseteq \mathcal{B}$) and $\mathcal{B}' \succeq \mathcal{A}$.

However, as defeaters are arguments, they may on its turn be defeated by other arguments, which could on their turn be defeated by other arguments, and so on. This prompts a recursive *dialectical* process rooted in a given argument $\langle \mathcal{A}_0, Q_0\rangle$, considering all their defeaters, defeaters for such defeaters, and so on. The process can be characterized in a tree-like structure called *dialectical tree* $\mathcal{T}_{\langle \mathcal{A}_0, Q_0\rangle}$, in which nodes are arguments, the root node is the original argument at issue, and every children node defeats its parent node. Every path in a dialectical tree is a sequence $[\langle \mathcal{A}_0, Q_0\rangle, \langle \mathcal{A}_1, Q_1\rangle, \langle \mathcal{A}_2, Q_2\rangle, \ldots, \langle \mathcal{A}_n, Q_n\rangle \ \ldots]$ that can be thought of as an exchange of arguments between two parties, a *proponent* (evenly-indexed arguments) and an *opponent* (oddly-indexed arguments). Each $\langle \mathcal{A}_i, Q_i\rangle$ is a defeater for the previous argument $\langle \mathcal{A}_{i-1}, Q_{i-1}\rangle$ in the sequence, $i > 0$. A path is *won* by the proponent if its length is odd (*i. e.*, the last argument in the path was given by the proponent, and no defeater followed it); otherwise the path is *lost*. An argument $\langle \mathcal{A}_0, Q_0\rangle$ is *warranted* iff every path in $\mathcal{T}_{\langle \mathcal{A}_0, Q_0\rangle}$ is won. Given a DeLP program $\mathcal{P} = (\Pi, \Delta)$, a query Q_0 wrt $\mathcal{P}$ is solved by computing the preceding tree-like structure. Three answers are distinguished: YES (there is at least one warranted argument $\mathcal{A}_0$ for Q_0); NO (there is at least one warranted argument $\mathcal{A}_0$ for $\sim Q_0$); UNDECIDED (none of the previous cases hold).

3.2 Applying DeLP to Prioritize Search Results

Our proposal is to apply DeLP to prioritize the search results returned by a context-based search system. In this setting, users preferences and background knowledge can be codified as facts and rules in a DeLP program $\mathcal{P}$. These facts and rules can come from different sources. For example, user's preferences could be entered explicitly by the user or could be inferred by the system (e.g., by monitoring the user's behavior). Additional facts and rules could be obtained from other repositories of structured (e.g., databases) and semi-structured data (e.g., the Web).

A context-based search system will provide a list of search results L =[s_1, s_2, s_3, s_4], where s_i is a unique name characterizing a piece of information $info(s_i)$. We assume that a number of features (meta-tags, filename, URL, etc.) can be identified and extracted from $info(s_i)$ by some specialized tool, as suggested by Hunter [6] in his approach to dealing with structured news reports. Such features will be encoded as a set $\mathcal{P}_{search}$ of new DeLP facts, extending thus the original program $\mathcal{P}$ into a new program $\mathcal{P}'$. A special operator **Revise** deals with possible inconsistencies found in $\mathcal{P}_{search}$ with respect to $\mathcal{P}'$, ensuring $\mathcal{P} \cup \mathcal{P}_{search}$ is not contradictory.[4] For the sake of simplicity, we will assume in our analysis that potential search results will be DeLP terms associated with a distinguished predicate name rel (which stands for *relevant* or *acceptable in light of the user preferences*). For each potential search result s_i, a new query $rel(s_i)$ will be analyzed in light of the new program $\mathcal{P}'$. Elements in the original list L of context-based search results will be classified into three sets, namely: (a) S^w (warranted search results): those results s_i for which there exists at least one warranted argument supporting $rel(s_i)$ based on $\mathcal{P}$'; (b) S^u (undecided search results): those results s_i for which there is no warranted argument for $rel(s_i)$, neither there is a warranted argument for $\sim rel(s_i)$ on the basis of $\mathcal{P}$', and (c) S^d (defeated search results): those results s_i such that there is a warranted argument supporting $\sim rel(s_i)$ on the basis of $\mathcal{P}$. Finally, the output presented to the user will be a sorted list L' in which the elements of L are classified according to their epistemic status with respect to $\mathcal{P}'$. Note that the resulting classification has a direct correspondence with the doxastic attitudes associated with answers to DeLP queries. For further discussion and examples on how to rank results using DeLP see [4].

4 COMBINING THE APPROACHES

Our proposal is to combine quantitative and qualitative analyzes to enhance Web search. Context-based search applies quantitative methods for the identification of context-specific terms and documents. On the other hand, preference-based search makes use of qualitative methods to rank results according to preference criteria provided by the user.

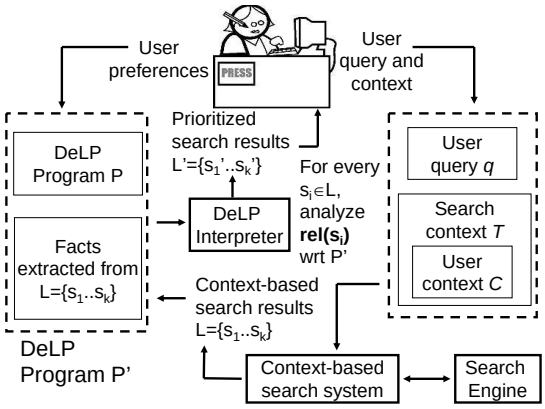

Figure 1: Architecture for Web search based on user context and preferences.

Figure 1 and the following algorithm summarize how the set of methods are combined in order to exploit user context and preferences for intelligent Web search:

[4]For example, contradictory facts may be found on the web. A simple belief revision criterion is to prefer the facts with a newer timestamp over the older ones.

ALGORITHM
INPUT:
Query q,
Set $C = \{t_1, \ldots, t_m\}$ describing the user context,
DeLP program $\mathcal{P}$ with user preferences,
Number h of iterations for incremental search.
OUTPUT:
List L_{new} of search results based on user context and preferences.
BEGIN
$T[0] = \{C\}$
FOR (i=0; i < h; i++)
DO
$T[i+1] = \emptyset$.
FOR EVERY set of terms $S \in T[i]$
DO
Submit queries to a search engine based on q and S.
Use C to filter and weight results and add them to L.
Compare search results to C to identify best descriptors and discriminators.
Use best descriptors and discriminators to expand C.
Use C to generate a set N of overlapping term clusters.
T[i+1]= T[i+1] $\cup$ N.
Let $L = [s_1, s_2, \ldots s_k]$ be the set of context-based results
$\mathcal{P}_{search}$ = {facts encoding $info(s_i)$, $i = 1 \ldots k$ }
{$info(s_i)$ *stands for features associated with result* s_i }
$\mathcal{P}'$:= **Revise** ($\mathcal{P} \cup \mathcal{P}_{search}$).
Initialize S^w, S^u, and S^d as empty sets.
{S^w, S^u, *and* S^d *stand for the set of results* s_i*'s which are warranted as relevant, undecided and warranted as non-relevant, respectively* }
FOR EVERY $s_i \in L$
DO
Solve query $rel(s_i)$ using DeLP program $\mathcal{P}'$
IF $rel(s_i)$ is warranted **THEN** add s_i to S^w
ELSE
IF $\sim rel(s_i)$ is warranted **THEN** add s_i to S^d
ELSE add s_i to S^d
Return L_{new} = $[s^w_1, \ldots, s^w_{j1}, s^u_1, \ldots, s^u_{j2}, s^d_1, \ldots, s^d_{j3}]$
END

5 CONCLUSIONS

This paper has described ongoing research on exploiting the information in the user context to refine Web search queries. In addition it proposes a novel approach for enhancing Web search technologies through the use of qualitative, argument-based analysis. In particular, we have shown that DeLP is a suitable tool for carrying on such analysis in a specific real-world application, providing thus a tool for higher abstraction when dealing with users' information needs.

Current research trends show that the combination of quantitative and qualitative analysis of user context and preferences will play a major role in Web search technology. We have proposed a set of methods to help carry out this goal.

References

[1] BUDZIK, J., HAMMOND, K. J., AND BIRNBAUM, L. Information access in context. *Knowledge based systems 14*, 1–2 (2001), 37–53.

[2] BUDZIK, J., HAMMOND, K. J., BIRNBAUM, L., AND KREMA, M. Beyond similarity. In *Proceedings of the 2000 Workshop on Artificial Intelligence and Web Search* (2000), AAAI Press.

[3] CHESÑEVAR, C., MAGUITMAN, A., AND LOUI, R. Logical Models of Argument. *ACM Computing Surveys 32*, 4 (Dec. 2000), 337–383.

[4] CHESÑEVAR, C. I., AND MAGUITMAN, A. G. Combining argumentation and web search technology: Towards a qualitative approach for ranking results. *Intl. Journal of Advanced Computational Intelligence 9*, 1 (2005), 53–60.

[5] GARCÍA, A., AND SIMARI, G. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming 4*, 1 (2004), 95–138.

[6] HUNTER, A. Hybrid argumentation systems for structured news reports. *Knowledge Engineering Review* (2001), 295–329.

[7] LEAKE, D. B., BAUER, T., MAGUITMAN, A., AND WILSON, D. C. Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *Proceedings of the AAAI-00 Workshop on Intelligent Lessons Learned Systems. Austin, Texas* (2000), AAAI Press, pp. 33–37.

[8] MAGUITMAN, A., LEAKE, D., AND REICHHERZER, T. Suggesting novel but related topics: towards context-based support for knowledge model extension. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces* (New York, NY, USA, 2005), ACM Press, pp. 207–214.

[9] MAGUITMAN, A., LEAKE, D., REICHHERZER, T., AND MENCZER, F. Dynamic extraction of topic descriptors and discriminators: Towards automatic context-based topic search. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM)* (Washington, DC, November 2004), ACM Press.

[10] SALTON, G., AND YANG, C. On the specification of term values in automatic indexing. *Journal of Documentation 29* (1973), 351–372.

[11] SMYTH, B., AND MCCLAVE, P. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning. Vancouver, Canada* (2001).