

Exploiting the Deep Web with DynaBot: Matching, Probing, and Ranking*

Daniel Rocco
University of West Georgia
Carrollton, GA, USA
drocco@westga.edu

James Caverlee, Ling Liu
Georgia Inst. of Technology
Atlanta, GA, USA
{caverlee,lingliu}@cc.gatech.edu

Terence Critchlow
Lawrence Livermore Nat'l Lab
Livermore, CA, USA
critchlow1@llnl.gov

ABSTRACT

We present the design of DYNABOT, a guided Deep Web discovery system. DYNABOT's modular architecture supports focused crawling of the Deep Web with an emphasis on matching, probing, and ranking discovered sources using two key components: service class descriptions and source-biased analysis. We describe the overall architecture of DYNABOT and discuss how these components support effective exploitation of the massive Deep Web data available.

Categories and Subject Descriptors:

H.3.3[Information Search and Retrieval]: Search process; H.4.m[Information Systems]: Miscellaneous

General Terms: Algorithms, Design

Keywords: Deep Web, crawling, service class, probing

1. PROBLEM DESCRIPTION

The Deep Web provides access to huge and growing data repositories on the Web and supports tools for searching, manipulating, and analyzing the information contained in those repositories. Unlike the surface Web, the Deep Web refers to the collection of Web data that is accessible by interacting with a Web-based query interface, and not through the traversal of hyperlinks. Recent estimates suggest that the size of the Deep Web greatly exceeds that of the surface Web – with nearly 92,000 terabytes of data on the Deep Web versus only 167 terabytes on the surface web[2].

Existing search engine indexers often ignore the data offered by Deep Web sources owing to the technical challenges that arise when attempting to locate, access, and index Deep Web data. The most significant challenge is the philosophical difference between the surface and Deep Web with respect to how data is stored: in the surface Web, data is stored in document files, while in the Deep Web, data is stored in databases or produced as the result of a computation. This difference is fundamental and implies that traditional document indexing techniques, which have been applied with extraordinary success on the surface Web, are

*This work is performed under a subcontract from LLNL under the LDRD project. The work of the first three authors is also partially supported by NSF CISE, NSF ITR, DoE SciDAC, CERCs Research Grant, IBM Faculty Award, IBM SUR grant, and HP Equipment Grant. The work of the fourth author is performed under the auspices of the U.S. Dep't of Energy by Univ. of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

inappropriate for the Deep Web.

Standardizing to web services using technologies like XML and SOAP has alleviated some of the heterogeneity of remote invocation, but the problem of discovering relevant sources remains. There have been a number of efforts to alleviate the discovery problem, including the use of Deep Web portals (like the one offered at www.profusion.com), but these solutions rely on manually listing sites and do not scale well. In the context of web services, registry-based solutions like the UDDI web service registries provide standard interfaces for describing, searching, and browsing for registered web services. But, current registries suffer from limited adoption and are limited to searching and browsing by metadata which limits the quality of both discovery and service selection. In addition, registry-based discovery relies on services correctly advertising themselves in a known repository, effectively limiting the number of services that can be discovered. Finally, the limited descriptive power in existing registry standards implies that service analysis is still required to ascertain a service's capabilities.

2. DESIGN AND ARCHITECTURE

With these challenges in mind, we present DYNABOT, a guided Deep Web discovery system. DYNABOT's modular architecture supports focused crawling of the Deep Web with an emphasis on matching, probing, and ranking discovered sources in an effort to exploit the vast amount of Deep Web data. Figure 1 presents the overall architecture of DYNABOT. DYNABOT utilizes an advanced crawler architecture that includes standard crawler components like a URL frontier manager, network interaction modules, global storage and associated data managers, and document processors, as well as the pluggable DYNABOT-specific semantic analyzers, which analyze the candidate Deep Web sources.

In this poster, we report two DYNABOT modules from our research experience: the first uses service class descriptions [3] to determine the capabilities of discovered sources and to *match* Deep Web sources that belong to members of a service class. The second is a suite of source-biased analysis techniques [1] for refined *probing* and *ranking* of Deep Web sources with respect to a domain of interest.

2.1 Service Class Matching

The first module supports guided matching of candidate Deep Web sources through the use of service class descriptions. A *service class* is a set of Deep Web sources that provide similar functionality or data access. A *service class description* (SCD) is an abstract description of a service class that specifies the minimum functionality that a Deep Web source must export to be classified as a member of the service class. An SCD is modeled as a triple: $SCD = \langle T, \mathcal{G}, \mathcal{P} \rangle$,

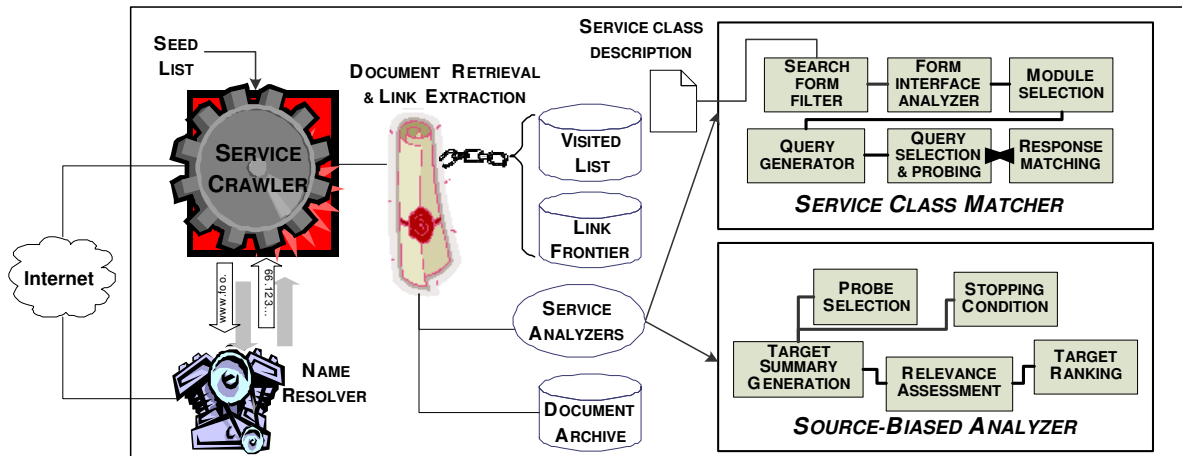


Figure 1: DynaBot System Architecture

where \mathcal{T} denotes a set of *type definitions*, \mathcal{G} denotes a *control flow graph*, and \mathcal{P} denotes a set of *probing templates*. The SCD is initially composed by a user or service developer and can be further revised via automated learning algorithms embedded in the DYNABOT matching process.

Types are used to describe the input and output parameters of a service class and any data elements that may be required during the course of interacting with a service. The DYNABOT service discovery system includes a type system that is modeled after the XML Schema type system with constructs for building atomic and complex types.

Due to the complexity of current sources, we model the underlying control flow of the source with a control flow graph. For many Deep Web sources, a query may have multiple response types depending on a number of factors. For example, a query that results in a *Normal* response under regular load conditions may result in a completely different *Unavailable* or *Wait 30 Seconds* response depending on the server and data availability. By defining a control flow graph to capture these different scenarios, we guide the choice of semantic analyzer for use on each response.

The third component of the service class description is the set of probing templates \mathcal{P} , each of which contains a set of input arguments and an expected response type that can be used to match a candidate service against the service class description and determine if it is an instance of the service class. The template may include hints to supply clues to the service classifier that help select the most appropriate input parameters to match an argument.

2.2 Source-Biased Analysis

The second DYNABOT module supports refined probing and ranking of Deep Web sources with respect to a domain of interest. This second module consists of two sub-modules: source-biased probing and source-biased relevance ranking.

Given a Deep Web site – the *source* – the source-biased probing technique leverages the summary information of the source to generate a series of biased probes for analyzing another Deep Web site – the *target*. This source-biased probing allows us to determine in very few interactions whether a target site is relevant to the source by probing the target with focused probes. Concretely, the source-biased probing algorithm generates a source-biased summary for a target

as follows: It uses the estimated summary of the source σ , denoted by $ESUMMARY(\sigma)$, as a dictionary of candidate probe terms and sends a series of query requests parameterized by probe terms, selected from $ESUMMARY(\sigma)$, to the target service τ ; for each probe term, it retrieves the top m matched documents from τ , generates summary terms and updates $ESUMMARY_\sigma(\tau)$. This process repeats until a stopping condition is met.

Given a source and a target service, we may evaluate the source-biased relevance of a target Deep Web site with respect to the source. We define $focus_\sigma(\tau)$ to be a measure of the topical focus of the target τ with respect to the source of bias σ . The focus metric ranges from 0 to 1, with lower values indicating less focus and higher values indicating more focus. Once a set of target sites have been evaluated with the source-biased relevance metric, we can then rank the target Deep Web sites with respect to the source of bias to identify the most relevant services to the source of bias.

3. CLOSING REMARKS

DYNABOT is designed as a foundation for developing a guided Deep Web discovery system, powered by two novel service discovery modules: the service class description matching module and the source-biased analysis module for probing and ranking. Our research on DYNABOT continues along several directions. First we will continue enhancing the capability and efficiency of these two modules and incorporating additional semantic analyzers for enhanced Deep Web discovery. We are also interested in iterative learning and efficient extraction of data quality information, managing data provenance, and incorporating dynamic adaptations into the service discovery and ranking process.

4. REFERENCES

- [1] J. Caverlee, L. Liu, and D. Rocco. Discovering and ranking web services with BASIL: A personalized approach with biased focus. In *2nd Int'l Conference on Service-Oriented Computing*, 2004.
- [2] P. Lyman and H. R. Varian. How much information. www.sims.berkeley.edu/how-much-info-2003, 2003.
- [3] D. Rocco, J. Caverlee, L. Liu, and T. Critchlow. Focused Crawling of the Deep Web using Service Class Descriptions, U. of West Georgia, 2005.