

Toward Structured Retrieval in Semi-structured Information Spaces

Scott B. Huffman, Catherine Baudin, and Robert A. Nado

Price Waterhouse Technology Centre
68 Willow Road
Menlo Park, CA 94025-3669
{huffman, baudin, nado}@tc.pw.com

Abstract

A semi-structured information space consists of multiple collections of textual documents containing fielded or tagged sections. The space can be highly heterogeneous, because each collection has its own schema, and there are no enforced keys or formats for data items across collections. Thus, structured methods like SQL cannot be easily employed, and users often must make do with only full-text search. In this paper, we describe an intermediate approach that provides structured querying for particular types of *entities*, such as companies, people, and skills. Entity-based retrieval is enabled by *normalizing* entity references in a heuristic, type-dependent manner. To organize and filter search results, entities are categorized as playing particular *roles* (e.g., company as *client*, as *vendor*, etc.) in particular *collection types* (directories, client engagement records, etc.). The approach can be used to retrieve documents and can also be used to construct entity profiles – summaries of commonly sought information about an entity based on the documents' content. The approach requires only a modest amount of meta-information about the source collections, much of which is derived automatically. On a set of typical user queries in a large corporate information space, the approach produces a dramatic improvement in retrieval quality over knowledge-free methods like full-text search.

1 Introduction

Decentralized information-sharing architectures like the World Wide Web and Lotus Notes make it easy for individuals to add information, but as the space grows, retrieval becomes more and more difficult. *Semi-structured* information sharing systems, including Lotus Notes and a variety of meta-tagging schemes being developed for the World Wide Web (e.g. Apple's Meta-Content Format), address part of this problem by providing the ability to structure local parts of the information space. In a semi-structured information

space, documents are sectioned into weakly-typed fields according to user specifications, and documents with the same field structure can be grouped into collections. Within a collection, field values can be used as indexes for easier retrieval.

Unfortunately, semi-structuring document collections does not solve the problem of retrieving information across a large information space. Even if individual collections are well designed for retrieval, users can be overloaded with the sheer number of collections. Retrieval across the entire space is difficult because it is highly heterogeneous. Each collection has its own local schema, and there are no enforced keys or formats for data items within or across collections.

In this paper, we address the problem of finding information across collections in a large semi-structured information space. Our goal is to provide querying that is more powerful and precise than full-text search, but without requiring the collections to be strongly typed, data normalized, and fully mapped to a global schema, as methods like multidatabase SQL require. In addition, to maintain the advantages of decentralized information sharing, we do not want to impose constraints like integrity checks when users create new documents.

Our approach is to provide high quality retrieval of information related to important *entities* in the information space. In our organization (a large professional services firm), important types of entities include people, companies, and consulting skills. A review of our largest collections revealed that nearly always, references to important entities are fielded rather than buried in free-running text. Because the same entity can be referred to in many different ways across a heterogeneous information space, our entity retrieval system *normalizes* references to entities in a heuristic, type-dependent manner. For instance, the person names "Mr. Bob Smith", "Smith, Robert", and "R. J. Smith" are normalized such that a query for any one (or a number of other possible forms) will retrieve documents containing any of them.

In addition to normalization, there are two entity-related *retrieval filters* provided by our retrieval system: *entity roles* and *collection types*. Entities often play identifiable *roles* within collections. For instance, a person can play the role of a partner on an engagement,

Figure 1. An NX search form

a manager, a contact at a client, etc.; a company can play the role of a client, a vendor, a newsmaker, etc. These roles can be useful in organizing or filtering retrieved information (“find mentions of XYZ as a client”).

Based on entities and roles they contain, collections as a whole can be classified into a small number of *collection types*. Examples include *client-engagements* collections, which contain clients, partners, and managers; *directories*, which contain staff names and phone numbers, etc. Like roles, collection types provide a useful way to organize or filter retrieved documents.

We have implemented an entity-based retrieval system called *NX* (for Notes Explorer) that operates over a large semi-structured information space. The space currently includes over one hundred corporate Lotus Notes collections and a small set of web collections, together containing about 300,000 documents. *NX* provides full-text search, entity-based search for people, companies, and skills, and role and collection-type organization and filtering of results. In addition, it provides retrieval of entity profiles -- integrated summaries of useful information about people and companies extracted from multiple, heterogeneous document collections.

NX uses HTML to communicate between a Web browser client and a server program that performs the requested searches. Figure 1 shows one of the system’s search forms. The pick-box with “Analysts Reports” chosen allows a user to choose among collection types. The pick-boxes along the right side, such as the one with “a company” chosen, allow a user to choose an entity type for each search string; “a phrase” indicates that a full-text search for the string should be performed. Figure 2 shows part of the set of results retrieved by the search in Figure 1.

A key hypothesis behind this work is that a *relatively small amount of meta-information* – much less than that required to normalize and map collections to a global schema – can give a large gain in query power and precision over knowledge-free methods like full-text search. *NX* is one illustration of this hypothesis. It requires only a modest amount of meta-information about each collection – its collection type, an indication of fields containing entities in various types and roles,

Figure 2. NX search results

and (for profile queries) pairs of fields that stand in specific semantic relations – and uses it to produce a dramatic improvement in retrieval quality for entity-related queries. Much of the required meta-information can actually be inferred automatically based on field names and data within the collections, using a simple heuristic classifier.

In what follows, we first motivate entity-based retrieval with a real-world example. Next, we describe the main components of our retrieval system, and present an empirical comparison between entity-based retrieval and full-text search for a set of typical queries. We conclude by discussing related and future work.

2 Entity-based retrieval


In a corporate setting, information in different documents is frequently linked through references to entities with business importance, such as people and companies. Often, users search for information about *particular* entities (e.g., “What is Bob Smith’s phone number?” or “Who’s the manager for the XYZ Co. account?”) as opposed to ungrounded, aggregate queries across sets of entities (e.g. “Show me all managers with more than five clients over \$5 million in sales”). We designed *NX* to support this type of search.


Consider a typical, but hypothetical, example from our organization. A staff member is writing a proposal to XYZ Company for some consulting work on XYZ’s new customer tracking system. She needs answers to questions like:


- (a) How large is XYZ Company? E.g., what are their assets, revenues, etc.?
- (b) Does our organization have a prior relationship with XYZ? Have we done other consulting work for them in the past?
- (c) If so, who did that work, and how can they be contacted?


Profile for "HP"

SIC Code:


3570 -- Computer & Office Equipment 

Net income: \$2,433,000,000 

Total assets: \$24,427,000,000 


Net revenues: \$31,519,000,000 

SEC Filings: 


WWW Home Page: 

Client Of:


[Audrey Auditor](#) 


[Tom Taxman](#) 


[Courtney Consultant](#) 

Vendor Relationship Coordinator: [Vince Vendrel](#) 

Analysts' mentions:

 01/17/97. Knowledge Info Transfer: Hewlett Packard: The Role of the Financial Controller in the mid - 1990s and beyond

 01/15/97. Knowledge Info Transfer: Hewlett Packard: Improving Supply Chain Management: A Survey of Best Practices

 01/13/97. Tower Group Research Notes: Hewlett Packard: Parallel Processing in Banking

 01/13/97. Tower Group Research Notes: Hewlett Packard: Client/Server Technology in Banking

Figure 3. Profile Search Results

(d) Do we have staff with expertise on customer tracking systems? How can they be contacted?

Each question refers to entities of various types – XYZ Company, staff members, phone numbers, skills, etc. – and these entities may be referred to differently in different documents. Some questions involve information that may be found in many collections of the same type – e.g., information about prior work for XYZ (b) might be found in numerous collections containing client engagements. Others involve linking information about XYZ with information about another entity -- e.g., question (c) requires finding staff names in documents that list XYZ engagements, and then finding contact information for those staff names.

To find answers using the source collections directly would require that the user:

- know the relevant collections and their locations.
- construct searches that accounts for different forms of an entity (e.g. "XYZ Corporation" vs. "X Y Z Inc.").
- discard hits where entities play irrelevant roles (e.g. XYZ as a vendor in an engagement, not as the client), and hits in irrelevant types of collections (e.g. XYZ mentioned in news rather than client engagements).

NX satisfies these requirements. Figure 3 displays the results of a profile search in NX given "HP" as a company name search string.¹ Normalization allows NX to retrieve information from documents that mention "Hewlett Packard", "Hewlett-Packard, Inc.", etc., as well as "HP". The headings (e.g., "SIC Code:" and "Client Of:") list specific values that have the specified relationship to the company of interest. These values

¹ Actual people names have been replaced in the HTML generated by Notes Explorer to preserve privacy.

are drawn from multiple documents in different collections; the square document icons are hyperlinks to the source documents. In the case of values representing people and companies, the value (e.g., "Audrey Auditor") is also displayed with a hyperlink that initiates a profile search on that value. This allows, for example, contact information to be found for people who have "HP" as a client. Other headings (e.g., "SEC Filings:" and "Analysts' mentions:") are followed only by document links, as it is the document as a whole that is of interest -- not specific information extracted from it.

The next section examines the key components of NX that support entity-based retrieval.

3 Key Components of NX

NX includes five key components:

1. **Semi-automatic field classification.** To build an index of entity references of different types, we must identify where those types occur within collections. NX's field classifier uses field names and sample values from a collection to classify fields as containing entity types (people's names, company names, phone numbers, dollar amounts, etc.) and roles. As classification is not 100% accurate or complete, an interface is provided to alter the entity and role types for each collection's fields.

2. **Entity normalization.** When collections are indexed, entity references are extracted from entity-typed fields and *normalized* in a heuristic manner, using formatting knowledge and synonym tables specific to each entity type. At retrieval time, entity search terms

are normalized in the same way and used to find matches in the index.

3. **Entity-related result organization and filtering.** Once a set of documents is retrieved, it can be viewed or filtered based on entity roles, collection types, and frequently co-occurring entities.

4. **Definition of a Partial Global Schema.** As an alternative to viewing only a set of retrieved documents, NX's profile search extracts and presents commonly desired information from those documents in a summary format. The foundation of profile search is the definition of a partial global schema consisting of global predicates describing those relationships among entities to be included in a profile. The global predicates are then mapped to the appropriate pairs of fields in each relevant document collection.

5. **Extraction of profile information.** After defining global predicates and mapping them to document collections, the information used to construct an entity profile can be extracted from the normalized entity index.

Next, we give a brief description of each component.

3.1 Semi-automatic field classification

Field classification attempts to identify an entity type and role for each collection field. The current version recognizes person names, company names, telephone numbers, geographic locations, office names, and dollar amounts. As in [Li & Clifton, 1994], the attributes used by the classifier are the *field name* and *sample values* from the documents in the collection.

The field classifier proceeds in three steps:

1. *Field name analysis:* Field names are tokenized according to capitalization and other separators. The tokens are then analyzed using a domain-dependent dictionary of entity types and roles. Tokens are matched against the entries in the dictionary using a set of matching heuristics that recognize different types of abbreviations. For instance, the tokens "Tel" and "Phone" match the entity type telephone, the token "Ptr" matches the entity role partner, and the token "Name" is indicative of a name of any type: person, company, or office. In addition, the field name analyzer uses a set of simple patterns. For instance, a field called "ContactPhone" is likely to be of type telephone because it contains a person token followed by a telephone token, whereas a field called "PhoneContact" is likely to be of type person. In the same way, the token "By" in the last position of a field name can indicate an action (e.g., "ServedBy"), suggesting that the field contains person or company names, which are the primary actors in our domain. Based on such patterns, field name analysis produces a weighted set of potential entity types.

2. *Field value analysis:* Sample data values are analyzed in a type-specific manner for each potential entity type produced by field name analysis. For each entity type, the sample values are analyzed to determine how confidently they can be considered that type of

entity. These confidence levels are combined to produce an overall confidence level for each potential type.

Three criteria are used to analyze values:

- a) *Value lookup:* The system looks up field values in tables of common values for different entity types. These include peoples' first names, large company names, and office sites. The confidence level for value lookup is high.
- b) *Common keywords:* The system looks for common words that appear in entity values. For instance, company names often contain "Corp." or "Inc.". Person names often contain titles like "Mr." or "Ms.". The confidence level for common keywords is medium.
- c) *Formatting:* Type-dependent routines attempt to match common formats for values of each entity type. For instance, person names have a small number of capitalized words, possibly including initials, etc. The confidence level for formatting criteria is low.

If the confidence level returned by field name analysis is low for a potential entity type, the system increases the number of sample values, in order to collect enough evidence to make a classification. An entity type is assigned if confidence exceeds a threshold.

3. *Role identification:* If an entity type is assigned, the classifier attempts to assign a role to the field based on its type and a second set of field name patterns. For example, a field of type *person name* with tokens "editor", "author", or "owner" indicates the role *author*.

In a small test, the field classifier analyzed 670 fields in 26 collections. Of these, 159 were of recognizable types. The field classifier correctly classified 154 of the 159 (97%). Most failures were due to failing to recognize tokens in the field name. Of the 154 classified fields, 61 were of the roles client, manager, partner, author, reviewer or vendor. The system correctly recognized 58 of these (95%).

More work is needed to extend the classifier's generality. It relies on a hand-built dictionary of field name tokens, but it should be possible to use standard induction techniques to learn such a dictionary from examples. The field classifier may also be extended to recognize other types, such as skill descriptions, vendor products, and technical terminology.

3.2 Entity Normalization

In a standard relational database, tuples from different tables that contain information about the same entity each contain a *key* for that entity allowing the tuples to be joined. In a semi-structured document space, however, there are rarely unique keys shared by collections. Rather, entities are referred to within text strings in a variety of formats, with a variety of synonyms and abbreviations.

Therefore, to allow search over entities, entity references must be normalized and matched. For maximum retrieval speed, NX normalizes entity references at indexing time. Its entity index stores both the original form and a normalized form of each entity

reference. At retrieval time, a normalized form of the user's search string is created and used to retrieve matches from the normalized entity index. In some cases, values are only partially normalized, and the original forms of retrieved matches and the search string are compared to verify the match.

Normalization and matching of strings that refer to entities has two important properties. First, it is *heuristic*. Generally, it is impossible to know with certainty whether two strings refer to the same entity. "Bob Smith" and "Robert A. Smith" may or may not be the same person; the retrieval system must make a reasonable guess.

Second, it is *type dependent*. Normalizing different types of entities requires different processing: e.g., normalizing person names is different than normalizing company names. Different variations in format are allowed for each type; "Smith, Bob" is a variation of "Bob Smith", but "Computer, Apple" is not a variation for "Apple Computer". In addition, different synonyms, abbreviations, and stop words apply. For company names, for instance, including or omitting words like "Corp." is only a small variation; for person names, "Mr." is a small variation unless the other string contains a female designator like "Ms." NX employs different normalization routines for different types, including small tables of synonyms, abbreviations, and stop words.

Due to space limitations, we will only briefly describe NX's normalization routines here.

For company names, the string containing the entity reference is first processed to produce uniform capitalization, combine leading initials (e.g., "I B M" becomes "IBM"), and expand common abbreviations. Next, we look up the leading words of the string in a table of company name synonyms. If a match is found, a designated synonym is used as the normalized form of the string. For example, the normalized form of "IBM" is "International Business Machines". If no match is found, the first contentful word of the string is used as its normalized form; e.g., the normalized form of "Foobaz Circuit Corp." would be "Foobaz".

At retrieval time, the search string for a company is normalized in the same way. Matches for the normalized form are retrieved from the entity index table. If the search string was found in the synonym table, all matches from the index are returned as retrieved matches, on the basis that appearing in the same synonym set implies a fairly strong match. For instance, all matches normalized to "International Business Machines" would be returned by a search for "IBM". In cases where no synonym is found, entity strings with the same first word are retrieved from the index and scored against the search string on a word-by-word basis. Words that appear in one but not the other reduce the match score by an amount proportional to the total number of words; common company words like "Corporation" are penalized less. Each entry with a match score over a threshold is returned as retrieved match. The search string "Foobaz Circuit Supply", for

instance, matches "Foobaz Circuit Corp." but not "Foobaz Lawn and Garden Products".

The algorithm for person names is slightly more complex, since it must account for more possible variations in word-order, title words ("Dr.", etc.), generation designators ("Jr.", "III", etc.) and so on.

In addition, pre-processing is required to find the portions of the input string containing entity references. Often, a field will contain multiple entity values in a single string, with spurious information interspersed. For example, a typical person name field value might be "Bob J. Smith Jr. – managing partner; Sue Jones, 415-555-1212, Palo Alto." NX's normalization routines extract "Bob J. Smith Jr." and "Sue Jones" out of this field value.

3.3 Entity-related result filtering

Documents that match a query can be filtered and sorted based on the collection types they are drawn from, the roles played by entities in the documents, and other entities that co-occur in the documents. In addition, standard filters/sorts such as date ranges and relevance scores are provided.

In Figure 2, results are displayed sorted by the collection type and collections they are drawn from. Here, the search has been filtered using the "Analysts Reports" collection type. In cases where documents are retrieved from many collection types, the display provides a useful breakdown of the major kinds of information retrieved.

The buttons along the top of the page indicate alternative sortings. 'View by role', for example, sorts the documents according to the roles of entities that matched the query within each document.

The 'Cross References' function retrieves the most frequently occurring other entities (people, companies, and skills) in the set of retrieved documents. Because they have been normalized, the co-occurring entities can be properly grouped independent of how they were referred to in the source documents. NX's profile search capability is more structured version of the 'Cross References' function – in profiles co-occurring entities are grouped by their semantic relations to the profiled entity as described in the following two sections.

3.4 Definition of a Partial Global Schema

The profile search capability of NX is based on a global vocabulary for describing the types of information that may be found about an entity in the different information sources that are available. A key point is that no attempt is made to define a complete global schema characterizing all of the relations that might be extracted from individual collections. Rather, the global schema used by NX is partial – containing only enough meta-information to support the desired entity profiles. Currently, the global vocabulary includes binary predicates of two types – entity predicates represent relationships between two entities, while document predicates represent relationships between an

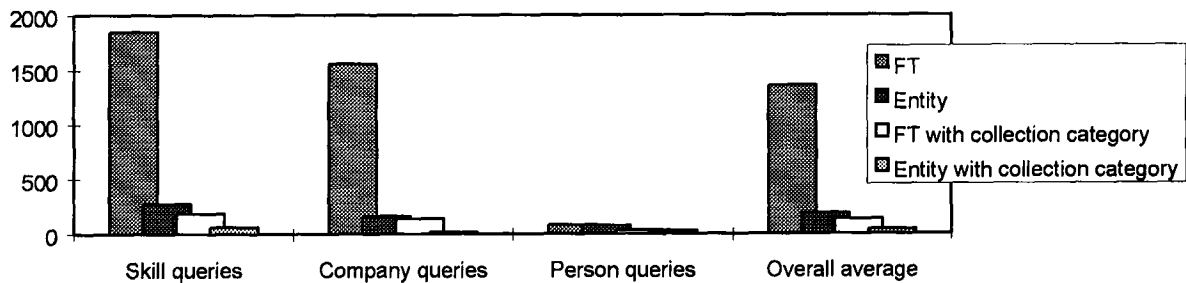


Figure 4. Average number of results for various query types

entity and a document that is “about” that entity. For example, “Work Phone” is an entity predicate representing the relationship between a person and a phone number where that person may be reached at work. Entity types are assigned to the domain and range arguments of a predicate to restrict its applicability.² For example, the “Work Phone” predicate is assigned the entity types *person name* and *phone number*. “Resume” is a document predicate relating a *person name* and a resume document.

In addition to declaring domain and range entity types, each predicate must be mapped to the relevant fields in collections that locally instantiate the predicate. For example, in the PW Name & Address Book, the “Work Phone” relation is mapped to a pair of a domain field (“FullName”) and a range field (“OfficePhoneNumber”). Other collections may also have information relevant to the “Work Phone” predicate but use differently named fields to record the person name and the phone number. An entity predicate may be mapped to multiple pairs of domain and range fields in a single collection. Document predicates have a simpler mapping, requiring only a domain field in each relevant collection.

Currently, the mapping of predicates to fields in collections is performed manually using a Web browser interface. The range of candidate collections and fields for each predicate is considerably narrowed by exploiting the entity types assigned to fields by NX’s field classifier. A collection can be ignored when mapping a predicate if it does not contain fields with entity types matching those specified for the domain and range arguments of the predicate. Given an eligible collection, candidates for the domain and range fields are narrowed to those with the proper entity types.

3.5 Extraction of profile information

A profile for a particular category of entity is defined by listing the global predicates that should make up the profile in the order in which they should be displayed in the results page of a profile search. Information can be associated with individual predicates through a Web browser interface to control the formatting, number, and sorting of profile results displayed for the predicates.

² Similar to sorts in a sorted, first-order language

Retrieving an entity profile involves five steps:

1. Retrieve records from the entity index whose normalized field values match the normalized forms of the search string and that have the correct entity type.
2. Filter out records retrieved in step 1 whose field is not a domain field for any predicate in the profile.
3. For each record A from step 2, retrieve records from the entity index that originate in the same document.
4. Filter out records retrieved in step 3 whose field is not a range field corresponding to A’s field as a domain field for one of the profile predicates.
5. Sort the remaining records by profile predicate and generate an HTML page displaying the results for each predicate.

Because they have been normalized, the results found for a particular profile predicate can be properly grouped independently of how they were referred to in the source documents. In essence, this uses the normalized entity index as a simple data warehouse, enabling an aggregation over entities in document sets.

4 Results

To evaluate NX’s entity-based retrieval and filtering functionality, we used a typical set of user questions from our organization’s staff. For each question, we tested four conditions:

1. *Naïve full-text search*: Our users are not sophisticated full-text users. Thus, we used simple full-text searches for the entities in the question.
2. *Full-text search with collection-type filtering*: To (1), we added that only results from the appropriate collection type be returned.
3. *Entity search*: Instead of full-text search, we used NX’s entity search for the entities in the question.
4. *Entity search with collection-type filtering*: To (3), we added that only results from the appropriate collection type be returned.

We tested these conditions for a set of 24 questions. The results are shown in Figure 4. As the graph shows, collection-type filtering produces a much smaller result set than full-text search alone, and collection-type plus

entity search reduces the result set further. Overall, the average naïve full-text result set was 1358 documents from 15 collections; the average entity-search with collection-type filtering result set was 37 documents from 2 collections.

Without pre-scored document collections, it is difficult to obtain strong measures of recall and precision for large information spaces (too large to search by hand). In the absence of scored answer keys, we have resorted to approximate measures. One way to approximate recall is by comparing to a method known to produce high recall. In this case, the results returned by entity + filtering searches are basically equivalent to the results that would be found by a user submitting a query against each collection's local schema, if the user knew the proper collections, field names and synonymous field values for each query. Thus, entity + filtering search produces a comparable level of recall.

Assuming comparable recall levels, we can obtain a rough measure of precision by examining the total number of documents returned by each retrieval method. Our test questions called for fairly focused answers, as opposed to "deep background" questions, like "find all documents that mention the internet", and thus small result sets are appropriate. In practice, large result sets containing hundreds or thousands of documents are much less useful than small sets, because they require so much hand filtering. In this sense, we consider the dramatically smaller result sets produced by entity + collection filtered search to be a large improvement over naïve full-text search, for entity-related queries.

Because of the synonyming capability of entity search, there were often documents retrieved by entity searches that were not retrieved by naïve full-text searches. For the first ten questions, on average, 14% of the documents returned by entity searches were *not* returned by naïve full-text search. In some cases – in particular, searches for contact information for people – missed documents contained the crucial information to answer the question at hand.

To date, we have not explicitly evaluated the entity profiling capability. In addition to attempting to obtain approximate measures of recall and precision, we plan to evaluate profiles' usefulness to end users, through user feedback and surveys.

5 Discussion and Future Work

The goal of our work is to provide better information retrieval across a large semi-structured space than full-text search, while avoiding excessive meta-information overhead. Our approach is based on observing that in an information space used by a particular organization, important entity types link information together and can be used as a central retrieval cue. This data-driven approach can be contrasted with schema-driven approaches used by multidatabase systems (e.g., [Arens et al., 1993]), and similar systems attempting to integrate structured world-wide web sources [Levy et al., 1996; Farquhar et al., 1995]. In schema-driven approaches, each local schema is mapped to a central

global schema, and mapping rules are used to translate between data formats used by different sources (e.g. [Collet et al., 1991]). These approaches are appropriate for relatively small numbers of tables where the data within each table is well-specified; however, semi-structured information spaces can include hundreds of sources and, even within single sources, data can have multiple formats. A schema integration phase would be burdensome in such a large space [Goh et al. 1994]. Instead, NX relies on heuristics to categorize fields into a small number of entity and role types, and normalizes entity values for retrieval (as in [Huffman and Steier, 1995]). The resulting retrieval system makes it practical to encompass a greater number and variety of data sources than multidatabase systems, although the query language is less general because queries must refer to a specific entity.

One area for future work is classification and normalization of a larger variety of entity types. New types we are considering include locations, product names, and technical terminology appearing in the body of a document. Field classification could be improved by using non-local information, such as data values' appearances in other collections. Indexing could be improved by extracting entity values from free-running text, in addition to tagged fields. This would require text processing tools like name finders [Rau, 1991; Hayes, 1994], technical-term extractors [Justeson and Katz, 1995; Chen and Chen, 1994], or parsers that find characteristic phrases using fonts and other cues (e.g. [Krulwich and Burkey, 1995]).

A related idea we are experimenting with is using text classification and thesaurus induction methods (e.g. [Grefenstette, 1994]) to attach organization-wide categories to documents as a whole. These categories can be used as retrieval cues or filters, similar to roles.

Future work on profile extraction will include extending profiles to other entity types such as service lines and skills and customizing profiles to meet the requirements of particular classes of users. We also plan to address the problem of ambiguous profile searches (e.g., distinguishing profile information for Robert A. Smith and Robert J. Smith when "Bob Smith" is supplied as the search string), use information about recency and reliability to resolve conflicts in information retrieved as part of a profile (e.g., multiple office phone numbers retrieved for a person), and add inference capabilities to the determination of profile results (e.g., determining a person's office telephone number from his assigned office and that office's main switchboard number).

To reduce the burden of providing the meta-information required for profile extraction, we plan to develop automated techniques for mapping global schema predicates to pairs of collection fields by exploiting abstract classifications of collections. For example, "directory" collections are more likely to contain a person's phone number, while client engagement archives are more likely to contain the names of a person's clients.

6 Conclusion

Semi-structured systems are an intermediate point between unstructured collections of textual documents (e.g., untagged Web pages) and fully structured tuples of typed data (e.g., relational databases). Based on observing how information is typically retrieved and used within our organization, we have developed an entity-based retrieval system over a large semi-structured information space. The system incorporates semi-automatic classification of fields, normalization of field values, filtered retrieval using entity roles and collection types, and structured retrieval of commonly required information in the form of entity profiles. For typical queries containing entities, the system provides much more focused and normalized retrieval than full-text search.

References

- [Arens et al., 1994] Arens, Y.; Chee, C. Y.; Hsu, Chun-Nan; and Knoblock, C. Retrieving and integrating data from multiple information sources. *Int'l J. on Intelligent and Cooperative Information Systems*, 1994.
- [Chen and Chen, 1994] Chen, Kuang-Hua, and Hsin-Hsi Chen, Extracting Noun Phrases from Large-Scale Texts: A Hybrid Approach and its Automatic Evaluation. *In Proceedings of the 32nd Annual Meeting of the ACLV. Las Cruces, New Mexico*, 1994.
- [Collet et al., 1991] Collet, C, Huhns, M, and Shen, W. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, pages 55-62, December 1991.
- [Farquhar et al., 1995] Farquhar, A., Dappert, A., Fikes, R., and Pratt, W. Integrating information sources using context logic. *In Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*. AAAI, 1995.
- [Goh et al., 1994] Goh, Cheng Hian; Madnick, Stuart E.; and Siegel, Michael D. Context Interchange: Overcoming the challenges of large-scale interoperable database systems. *In Proceedings of the 3rd International Conference on Information and Knowledge Management*. 1994.
- [Grefenstette, 1994] Grefenstette, Gregory. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers, 1994.
- [Hayes, 1994] Hayes, P. NameFinder: Software that finds names in text. Carnegie Group Inc. technical report, 1994.
- [Huffman and Steier, 1995] Huffman, Scott; Steier, David. Heuristic joins to integrate structured heterogeneous data. *In Working notes of the AAAI Spring Symposium on Information Gathering in Heterogeneous Distributed Environments*. AAAI, 1994.
- [Justeson and Katz, 1995] Justeson, John, and Slava Katz, Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text. *In Natural Language Engineering V 1.1.*, 1995.
- [Krulwich and Burkey, 1995] Krulwich, Bruce; Chad, Burkey.. ContactFinder: Extracting indications of expertise and answering questions with referrals. *In Working Notes of the AAAI Spring Symposium on Knowledge Navigation*. AAAI, 1995.
- [Levy et al., 1996] Levy, A., Rajaraman, A., and Ordille, J. Query-answering algorithms for information agents. *In Proc. Of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 40-47, 1996.
- [Li & Clifton, 1994] Li, Wen-Syan and Clifton, C. Semantic integration in heterogeneous databases using neural networks. *In Proc. 20th International Conference on Very Large Data bases, Santiago, Chile*, 1994.
- [Rau, 1991] Rau, L. Extracting company names from text. *IEEE Conference on AI Applications*, 1991.