Breaking through the syntax barrier: Searching with entities and relations

Soumen Chakrabarti
soumen@cse.iitb.ac.in

IIT Bombay

Abstract. The next wave in search technology will be driven by the identification, extraction, and exploitation of real-world entities represented in unstructured textual sources. Search systems will either let users express information needs naturally and analyze them more intelligently, or allow simple enhancements that add more user control on the search process. The data model will exploit graph structure where available, but not impose structure by fiat. First generation Web search, which uses graph information at the macroscopic level of inter-page hyperlinks, will be enhanced to use fine-grained graph models involving page regions, tables, sentences, phrases, and real-world-entities. New algorithms will combine probabilistic evidence from diverse features to produce responses that are not URLs or pages, but entities and their relationships, or explanations of how multiple entities are related.

1 Toward more expressive search

Search systems for unstructured textual data have improved enormously since the days of boolean queries over title and abstract catalogs in libraries. Web search engines index much of the full text from billions of Web pages and serve hundreds of millions of users per day. They use rich features extracted from the graph structure and markups in hypertext corpora.

Despite these advances, even the most popular search engines make us feel that we are searching with mere strings: we do not find direct expression of the entities involved in our information need, leave alone relations that must hold between those entities in a proper response. In a plenary talk at the 2004 World-wide Web Conference, Udi Manber commented:

> If music had been invented ten years ago along with the Web, we would all be playing one-string instruments (and not making great music).

referring to the one-line text boxes in which users type in 1–2 keywords and expect perfect gratification with the responses.

Apart from classical Information Retrieval (IR), several communities are coming together in the quest of expressive search, but they are coming from very different origins.

Databases and XML: To be sure, the large gap between the user's information need and the expressed query is well-known. The database community has been traditionally uncomfortable with the imprecise nature of queries inherent in IR.

The preference for precise semantics has persisted from SQL to XQuery (the query language proposed for XML data). The rigor, while useful for system-building, has little appeal for the end-user, who will not type SQL, leave alone XQuery.

Two communities are situated somewhere between "uninterpreted" keyword search systems and the rigor of database query engines. Various sub-communities of natural language processing (NLP) researchers are concerned with NL interfaces to query systems. The other community, which has broad overlaps with the NLP community, deals with information extraction (IE).

NLP: Classical NLP is concerned with annotating grammatical natural language with parts of speech (POS), chunking phrases and clauses, disambiguating polysemous words, extracting a syntactic parse, resolving pronoun and other references, analyze roles (eating with a spoon vs. with a friend), prepare a complete computer-usable representation of the knowledge embedded in the original text, and perform automatic inference with this knowledge representation. Outside controlled domains, most of these, especially the latter ones, are very ambitious goals. Over the last decade, NLP research has gradually moved toward building robust tools for the simpler tasks [19].

IE: Relatively simple NLP tasks, such as POS tagging, named entity tagging, and word sense disambiguation (WSD) share many techniques from machine learning and data mining. Many such tasks model unstructured text as a sequence of tokens generated from a finite state machine, and solve the reverse problem: given the output token sequence, estimate the state sequence. E.g., if we are interested in extracting dates from text, we can have a positive and a negative state, and identify the text spans generated by the positive state. IE is commonly set up as a supervised learning problem, which requires training text with labeled spans.

Obviously, to improve the search experience, we need that

- Users express their information need in some more detail, while minimizing additional cognitive burden
- The system makes intelligent use of said detail, thus rewarding the burden the user agrees to undertake

This new contract will work only if the combination of social engineering and technological advances work efficiently in concert.

2 The new contract: Query syntax

Suitable user interfaces, social engineering, and reward must urge the user to express their information need in some more detail. Relevance feedback, offering query refinements, and encouraging the user to drill down into response clusters are some ways in which systems collect additional information about the user's information need. But there are many situations where direct input from the user can be useful. I will discuss two kinds of query augmentation.

Fragments of types: If the token *2000* appears in grammatical text, current technology can usually disambiguate between the year and some other

number, say a money amount. There is no reason why search interfaces cannot accept a query with a type hint so as to avoid spurious matches. There is also no reason a user cannot look for persons related to SVMs using the query `PersonType NEAR "SVM"`, where `PersonType` is the anticipated response type and *SVM* a word to match. To look for a book in SVMs published around year 2000, one might type `BookType (NEAR "SVM" year~2000)`. I believe that the person composing the query, being the stakeholder in response quality, can be encouraged to provide such elementary additional information, provided the reward is quickly tangible. Moreover, reasonably deep processing power can be spent on the query, and this may even be delegated to the client computer.

Attributes, roles and relations: Beyond annotating query tokens with type information, the user may want to express that they are looking for "a business that repairs iMacs," "the transfer bandwidth of USB2.0," and "papers written in 1985 by C. Mohan." It should be possible to express broad relations between entities in the query, possibly the placeholder entity that must be instantiated into the answer. The user may constrain the placeholder entity using attributes (e.g. MacOS-compliant software), roles and relations (e.g., a student advised by X). The challenge will be to support an ever-widening set of attribute types, roles and relations while ensuring ongoing isolation and compatibility between knowledge bases, features, and algorithms.

Compared to query syntax and preprocessing, whose success depends largely on human factors, we have more to say about executing the internal form of the query on a preprocessed corpus.

3 The new contract: Corpus and query processing

While modest changes may be possible in users' query behavior, there is far too much inertia to expect content creators to actively assist mediation in the immediate future. Besides, questions preprocessing can be distributed economically, but corpus processing usually cannot.

The situation calls for relatively light processing of the corpus, at least until query time. During large scale use, however, a sizable fraction of the corpus may undergo complex processing. It would be desirable but possibly challenging to cache the intermediate results in a way that can be reused efficiently.

3.1 Supervised entity extraction

Information extraction (IE), also called named entity tagging, annotates spans of unstructured text with markers for instances of specified types, such as people, organizations, places, dates, and quantities.

A popular framework [11] models the text as a linear sequence of tokens being generated from a Markov state machine. A parametric model for state transition and symbol emission is learned from labeled training data. Then the model is evaluated on test data, and spans of tokens likely to be generated by desired states are picked off as extracted entities.

Generative models such as hidden Markov models (HMMs) have been used for IE for a while [7]. If $\mathbf{s}$ is the (unknown) sequence of states and $\mathbf{x}$ the sequence of output features, HMMs seek to optimize the joint likelihood $\Pr(\mathbf{s}, \mathbf{x})$.

In general, $\mathbf{x}$ is a sequence of feature vectors. Apart from the tokens themselves, some derived features found beneficial in IE are of the form: Does the token

- Contain a digit, or digits and commas?
- Contain patterns like DD:DD or DDDD or DD's where D is a digit?
- Follow a preposition?
- Look like a proper noun (as flagged by a part-of-speech tagger[1])?
- Start with an uppercase letter?
- Start with an uppercase letter and continue with lowercase letters?
- Look like an abbreviation (e.g., uppercase letters alternating with periods)?

The large dimensionality of the feature vectors usually corners us into naive independence assumptions about $\Pr(\mathbf{s}, \mathbf{x})$, and the large redundancy across features then lead to poor estimates of the joint distribution.

Recent advances in modeling conditional distributions [18] directly optimize $\Pr(\mathbf{s}|\mathbf{x})$, allowing the use of many redundant features without attempting to model the distribution over $\mathbf{x}$ itself.

3.2 Linkage analysis and alias resolution

After the IE step, spans of characters and tokens are marked with type identifiers. However, many string spans (called *aliases*) may refer to a single entity (e.g., *IBM*, *International Business Machines*, *Big Blue*, *the computer giant* or www.ibm.com). The variations may be based on abbreviations, pronouns, anaphora, hyperlinks and other creative ways to create shared references to entities. Some of these aliases are syntactically similar to each other but others are not.

In general, detecting aliases from unstructured text, also called *coreferent resolution*, in a complete and correct manner is considered "NLP complete," i.e., requires deep language understanding and vast amounts of world knowledge. Alias resolution is an active and difficult area of NLP research. In the IE community, more tangible success has been achieved within the relatively limited scope of **record linkage**.

In record linkage, the first IE step results in structured tables of entities, each having attributes and relations to other entities. E.g., we may apply IE techniques to bibliographies at the end of research papers to populate a table of papers, authors, conferences/journals, etc. Multiple rows in each table may refer to the same object. Similar problems may arise in Web search involving names of people, products, and organizations.

The goal of record linkage is to partition rows in each table into equivalence classes, all rows in a class being references to one real-world entity. Obviously, knowing that two different rows in the author table refer to the same person (e.g., one may abbreviate the first name) may help us infer that two rows in the paper table refer to the same real-world paper.

A veriety of new techniques are being brought to bear on record linkage [10] and coreferent resolution [20], and this is an exciting area of current research.

[1] Many modern part-of-speech taggers are in turn driven by state transition models.

3.3 Bootstrapping ontologies from the Web

The set of entity types of interest to a search system keeps growing and changing. A fixed set of types and entities may not keep up. The system may need to actively explore the corpus to propose new types and extract entities for old and new types. Eventually, we would like the system to learn how to learn.

Suppose we want to discover instances of some type of entity (city, say) on the Web. We can exploit the massive redundancy of the Web and use some very simple patterns [16, 8, 1, 13]:

"cities" {","} "such as" NPList2
NP1 {","} "and other cities"
"cities" {","} "including" NPList2
NP1 "is a city"

Here { } denotes an optional pattern and NP is a noun phrase. These patterns are fired off as queries to a number of search engines. A set of *rules* test the response Web pages for the existence of valid instantiations of the patterns. A rule may look like this:

NP1 "such as" NPList2 AND
head(NP1)="cities" AND
properNoun(head(each(NPList2)))
⇒ instanceOf(City,head(each(NPList2)))

KnowItAll [13] makes a probabilistic assessment of the quality of the extraction by collecting co-occurrence statistics on the Web of terms carefully chosen from the extracted candidates and pre-defined *discriminator phrases*. E.g., if X is a candidate actor, "X starred in" or "starring X" would be good discriminator phrases. KnowItAll uses the *pointwise mutual information* (PMI) formulation by Turney [24] to measure the association between the candidate instance I and the discriminator phrase D: $\text{PMI}(I, D) = |\text{Hits}(D + I)|/|\text{Hits}(I)|$.

Apart from finding instances of types, it is possible to discover subtypes. E.g., if we wish to find instances of *scientists*, and we have a seed set of instances, we can discover that physicists and biologists are scientists, make up new patterns from the old ones (e.g. "scientist X" to "physicist X") and improve our harvest of new instances.

In Sections 3.5 and 3.6 we will see how automatic extraction of ontologies can assist next-generation search.

3.4 Searching relational data with NL queries

In this section and the next (§3.5), we will assume that information extraction and alias analysis have led to a reasonably clean entity-relationship (ER) graph. The graphs formed by nodes corresponding to authors, papers, conferences and journal in DBLP, and actors/actresses, movies, awards, genres, ratings, producers and music directors in the Internet Movie Database (IMDB) are examples of reasonably clean entity-relationship data graphs. Other real-life examples involve e-commerce product catalogs and personal information management data, with organizations, people, locations, emails, papers, projects, seminars, etc.

There is a long history of systems that give a natural language interface (NLI) to relational engines [4], but, as in general NLP research, recent work has

moved from highly engineered solutions to arbitrarily complex problems to less knowledge-intensive and more robust solutions for limited domains [21]. E.g., for a table `JOB(description,platform,company)` and the NL query *What are the HP jobs on a UNIX system?*, the translation to SQL might be `select distinct description from JOB WHERE company = 'HP' and platform = 'UNIX'`. The main challenge is to agree on a perimeter of NL questions within which an algorithm is required to find a correct translation, and to reliably detect when this is not possible.

3.5 Searching entity-relationship graphs

NLI systems take advantage of the precise schema information available with the "corpus" as well the well-formed nature of the query, even if it is framed in uncontrolled natural language. The output of IE systems has less elaborate type information, the relations are shallower, and the questions are most often a small set of keywords, from users who are used to Web search and do not wish to learn about any schema information in framing their queries.

Free-form keyword search in ER graphs raises many interesting issues, including the query language, the definition of a "document" in response to a query, how to score a document which may be distributed in the graph, and how to search for these subgraphs efficiently.

Multiple words in a query may not all match within a single row in a single table, because ER graphs are typically highly normalized using foreign key constraints. In an ER version of DBLP, paper titles and author names are in different tables, connected by a relation `wrote(author,paper)`. In such cases, what is the appropriate unit of response? Recent systems [6, 3, 17] adopt the view that the response should be some minimal graph that connects at least one node containing each query keyword.

Apart from type-free keyword queries, one may look for a single node of a specified type (say, a paper) with high proximity to nodes satisfying various predicates, e.g., keyword match ("indexing", "SIGIR") or conditions on numeric fields (`year<1995`). Resetting random walks [5] are a simple way to answer such queries. These techniques are broadly similar to Pagerank [9], except that the random surfer teleports only to nodes that satisfy the predicates. Biased random walks with restarts are also related to effective conductance in resistive networks. In a large ER graph, it is also nontrivial to *explain* to the user why/how entities are related; this is important for diagnostics and eliciting user confidence. Conductance-based approaches work well [14]: we can connect +1 V to one node, ground the other, penalize high-fanout nodes using a grounded sink connected to every node, and report subgraphs that conduct the largest current out of the source node.

Recent years have seen an explosion of analysis and search systems for ER graphs, and I expect the important issues regarding meaningfulness of results and system scalability to be resolved in the next few years.

3.6 Open-domain question answering

Finally, the Web at large will continue to be an "open-domain" system where comprehensive and accurate entity and relation extraction will remain elusive. No schema of entities and relationships can be complete at any time, even if

they become more comprehensive over time. Moreover, even a cooperative user will not be able to remember and exploit a universal "type system" in asking questions. Instead, search systems will provide some basic set of *roles* [15] that apply broadly. Questions will express roles or refinements of roles, and will be matched to probabilistic role annotations in the corpus.

In open-domain QA, question analysis and response scoring will necessarily be far more tentative. Some basic machine learning will reveal that the question *When was television invented?* expects the type of the answer (atype) to be a *date*, and that the answer is almost certainly only a few tokens from the word *television* or its synonym. In effect, current technology [22, 2, 12, 23] can translate questions into the general form

```
find x from corpus where x InstanceOf(Atype(question))
  and x RelatedTo GroundConstants(question)
```

Here `Atype(question)` represents the concept of time, and we are looking for a reference to an entity `x` which is an instance of time. (This is where a system like KnowItAll comes into play.) In the example above, *television* or *TV* would be in `GroundConstants(question)`.

Checking the predicate `RelatedTo` is next to impossible in general. QA systems employ a variety of approximations. These may be as crude as linear proximity (the number of of tokens separating `x` from `GroundConstants(question)`. Linear proximity is already surprisingly effective [23]. More sophisticated systems[2] attempt a parse of the question and the passage, and verify that `x` and `GroundConstants(question)` are related in a way specified by (a parse of) the question. As might be expected, there is a trade-off beteen speed and robustness on one hand and accuracy and brittleness on the other.

4 Conclusion

Many of the pieces required for better searching are coming together. Current an upcoming research will introduce synergy as well as build large, robust applications. The applications will need to embrace bootstrapping and life-long learning better than before. The architecture must isolate feature extraction, models, and algorithms for estimation and inferencing. The interplay between processing stages makes this goal very challenging. The applications must be able to share models and parameters across different tasks and across time.

References

1. E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *International Conference on Digital Libraries (DL)*, volume 5. ACM, 2000.
2. E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *WWW Conference*, pages 169–178, 2001.
3. S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *ICDE*, San Jose, CA, 2002. IEEE.

[2] Visit, e.g., http://www.languagecomputer.com/

4. I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Journal of Language Engineering*, 1(1):29–81, 1995.
5. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Authority-based keyword queries in databases using ObjectRank. In *VLDB*, Toronto, 2004.
6. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, San Jose, CA, 2002. IEEE.
7. D. M. Bikel, R. L. Schwartz, and R. M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3):211–231, 1999.
8. S. Brin. Extracting patterns and relations from the World Wide Web. In P. Atzeni, A. O. Mendelzon, and G. Mecca, editors, *WebDB Workshop*, volume 1590 of *LNCS*, pages 172–183, Valencia, Spain, Mar. 1998. Springer.
9. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th World-Wide Web Conference (WWW7)*, 1998.
10. W. Cohen and J. Richman. Learning to match and cluster entity names. In *SIGKDD*, volume 8, 2002.
11. T. G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30. Springer-Verlag, 2002.
12. S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: Is more always better? In *SIGIR*, pages 291–298, 2002.
13. O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. In *WWW Conference*, New York, 2004. ACM.
14. C. Faloutsos, K. S. McCurley, and A. Tomkins. Connection subgraphs in social networks. In *Workshop on Link Analysis, Counterterrorism, and Privacy*, SIAM International Conference on Data Mining, 2004.
15. D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.
16. M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *International Conference on Computational Linguistics*, volume 14, pages 539–545, 1992.
17. V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.
18. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.
19. R. J. Mooney. Learning semantic parsers: An important but under-studied problem. In *AAAI Spring Symposium on Language Learning: An Interdisciplinary Perspective*, pages 39–44, Mar. 2004.
20. V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *ACL*, volume 40, 2002.
21. A. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *Intelligent User Interfaces*, pages 149–157, Miami, 2003. ACM.
22. J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *SIGIR*, pages 184–191. ACM, 2000.
23. G. Ramakrishnan, S. Chakrabarti, D. A. Paranjpe, and P. Bhattacharyya. Is question answering an acquired skill? In *WWW Conference*, pages 111–120, New York, 2004.
24. P. D. Turney. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *ECML*, 2001.