Scaling Information Extraction to Large Document Collections

Eugene Agichtein
Microsoft Research
eugeneag@microsoft.com

Abstract

Information extraction and text mining applications are just beginning to tap the immense amounts of valuable textual information available online. In order to extract information from millions, and in some cases, billions of documents, different solutions to scalability emerged. We review key approaches for scaling up information extraction, including using general-purpose search engines as well as indexing techniques specialized for information extraction applications. Scalable information extraction is an active area of research, and we highlight some of the opportunities and challenges in this area that are relevant to the database community.

1 Overview

Text documents convey valuable *structured* information. For example, medical literature contains information about new treatments for diseases. Similarly, news archives contain information useful to analysts tracking financial transactions, or to government agencies that monitor infectious disease outbreaks. All this information could be managed and queried more easily if represented in a structured form. This task is typically called *information extraction*. More specifically, information extraction systems can identify particular types of entities (e.g., person names, locations, organizations, or even drug and disease names) and relationships between entities (e.g., employees of organizations or adverse interactions between medical drugs) in natural language text. In this paper we focus on *entity extraction* (NER) and *event or relation extraction* (RE). Once created, the structured representation of entities or relations can be used to answer specific questions quickly and precisely by retrieving answers instead of complete documents, for sophisticated query processing, data integration, and data mining. Managing text is an increasingly important use of relational database management systems [9], and information extraction can be a key technology for this effort.

We focus on extracting information from large document collections (e.g., newspaper archives, web snapshots, biomedical literature archives). This setting is particularly important as information extraction is most useful when the collections are too large to process manually. Additionally, as we will describe, some extraction systems perform best precisely when the collection sizes are large (e.g., [1, 25]). Hence, for usefulness and even accuracy, scaling information extraction to large document collections is crucial. The document collection sizes we consider range from a few hundred thousand documents (e.g., Newspaper archives) to millions of documents (e.g., PubMed and other "hidden web" databases) to tens or hundreds of millions of documents (e.g., Web snapshots, focused web crawls). We provide a brief overview of information extraction process in Section 2.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Unfortunately, extracting the entities and relationships from a document is computationally expensive. Even simple information extraction tasks can require days or weeks of running time to process a large collection. For example, Ravichandran et al. [27] estimate that to just perform part-of-speech tagging (a common pre-processing step for information extraction) over a terabyte of text (between 50 and 100 million documents) required 125 days on a 2.5GHz PC, and a shallow syntactic parse required 10 machine-years. Clearly, this is not feasible for large document collections[1]. To scale up information extraction to large collections, four main approaches have been used:

- Scanning the collection using simplified and efficient rules: In this case, every document is processed using patterns and rules highly optimized for speed. In this model the complete scanning process is repeated for each new task (Section 3).
- Exploiting general-purpose search engines: To avoid scanning all documents in a collection, some systems use generic search engines to zoom in on relevant documents (Section 4).
- Using specialized indexes and custom search engines: A special-purpose search engine can index and query annotations useful for a predefined family of information extraction tasks. In some cases this may allow doing extraction over the index only, for dramatic efficiency gains (Section 5).
- Distributed processing: We briefly describe representative distributed data mining solutions that could be applied for scalable text mining and information extraction (Section 6).

Some of the efficiency approaches can degrade extraction completeness and accuracy, as well as generality and applicability of the resulting solutions. We discuss these challenges and promising research directions in Section 7, which concludes the paper.

2 Background: Information Extraction

The general information extraction process is outlined in Figure 1 (adapted from [15]). In general, a document is broken up into chunks (e.g., sentences or paragraphs), and rules or patterns applied to identify entities. For the NER task, systems usually scan each document for textual "clues" indicating presence of a useful entity. Most common clues are the text surrounding the entity and the text of entity itself, as well as part-of-speech tags and word classes if available. Then, for the RE task, scenario-level extraction patterns are applied to infer relationships between the extracted entities (See [15] for natural language processing-focused overview). Some systems can use statistics collected over the whole collection to assign confidence scores to extracted objects. Either after or during the extraction, information can be merged for multiple occurrences of the same object (and different objects with shared attribute values can be disambiguated). These postprocessing steps are relatively fast compared to the actual information extraction process, and are beyond the scope of this paper. Note that entities can be extracted independently of the relation, so that entity annotations can be shared across multiple relation extraction tasks.

The different stages in the extraction process have varying computational requirements. Most probabilistic parsers or taggers use a form of Viterbi algorithm for decoding the most likely sequence of tags (e.g., [22]), which have linear complexity with respect to sequence length and corpus size, but with widely varying constants. Pattern-based extraction systems (e.g., [1]) apply each pattern to each candidate passage in a document, resulting in complexity linear with the size the collection and the number of patterns used (which can be large for partially supervised and unsupervised extraction systems). Complexity of rule-based extraction systems is difficult to estimate, but is consistently reported to be high, as it usually takes seconds to process a medium-size document (3K), resulting in estimates of years [27] required to process large document collections.

[1]Most preprocessing steps only need to be run once if we store the annotated text. Also, the preprocessing step is inherently parallelizable. We discuss these issues in subsequent sections.

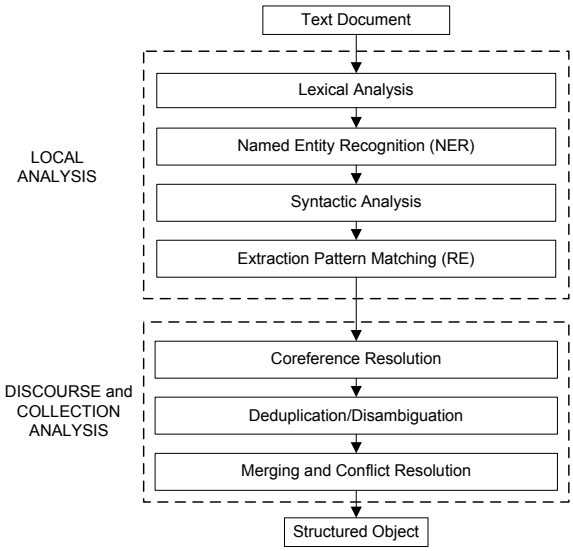

Figure 1: Typical stages in the information extraction process.

3 Scanning Large Document Collections

A traditional information extraction approach is to scan every document in a given collection, possibly using various forms of filtering to discard documents (or passages) as early as possible in the process. One approach is to use a classifier or hand-crafted patterns. Only the documents that match these (presumably "cheap") filters are processed further. For example, a system for extracting information about disease outbreak events [16] uses hand-crafted regular expressions to select documents to process further with the full features extraction system. These filtering patterns are usually designed to have high recall (i.e., not to discard useful documents) while ignoring a large fraction of the non-useful documents. In some settings (e.g., focused crawling), it is possible to discard documents without processing the document text (e.g., by applying rules to the document URLs or links pointing at the document) [5, 8]. Efficient text filtering (e.g., by using optimized regular expression matching and even specialized hardware solutions) were reported for text filtering as early as 1993 [24], and could be naturally adapted to work with information extraction.

A different approach is to use only extremely simple, "cheap" extraction patterns, and apply them to *every* document in the collection [25]. This relies on the assumption that information in large text collections appears *redundantly*, and at least some of the occurrences of a desired entity or relationship will match one of the simple patterns. The authors describe experiments with extracting pairs of noun phrases for the *is-a* relations (e.g., ⟨"MCI WorldCom", "phone company"⟩). The system uses 15 simple lexical and part-of-speech patterns, followed by a more expensive machine learning-based postprocessing step. The authors report requiring 10 days to process a 15GB document collection (approximately 5 million documents) using this implementation, which is still an order of magnitude slower than part-of-speech tagging. Interestingly, the reported accuracy of the simple lexical pattern-based system is comparable to the accuracy of the much slower approach requiring full syntactic parsing of each sentence.

The created annotations can be stored and re-used for all future extraction tasks that require such information (e.g., locations of the named entities in the documents to be used for the relation extraction task). Hence, the initial pre-processing effort would amortize if the annotations are general enough. Another example of such preprocessing is indexing the words and the documents in which they occur, as typically done by general-purpose text search engines. Next we describe two scalable information extraction architectures that make use of such indexing.

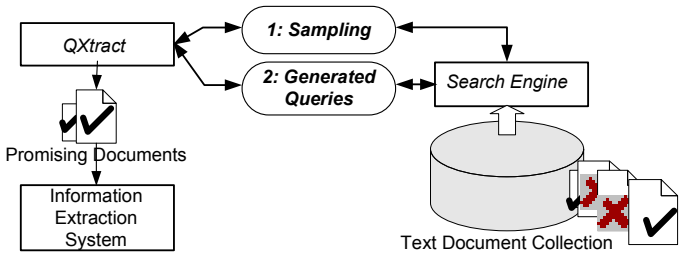

Figure 2: Querying generic search engines for scalable information extraction.

4 Exploiting General-Purpose Search Engines

Often, only a small fraction of the documents contain information that is relevant to the extraction task. Hence it is not necessary for extraction completeness –or desirable from an efficiency viewpoint– to run the information extraction system over every database document. Furthermore, if a document collection is the set of all web pages indexed by a search engine such as Google, then it is virtually impossible to extract information from every page. For these reasons, an intuitive approach is to zoom in on the promising documents, while ignoring the rest. This approach was introduced in the *QXtract* system [2] for efficiently extracting relations from large document collections.

The general *QXtract* architecture is outlined in Figure 2. Starting with a set of user-provided seed tuples for the target relation, *QXtract* retrieves a small sample of documents, likely to be useful to the extraction system, as well as other randomly chosen documents, likely to be useless to the extraction system. The information extraction system is run over this sample set, producing as output a set of extracted tuples and the identifiers of useful documents. The documents in the sample are thus labeled automatically as either positive or negative examples, where the positive examples represent the documents in the sample from which the information extraction system was able to produce tuples. These examples allow *QXtract* to derive queries targeted to match –and retrieve– documents similar to the positive examples. These queries are used to retrieve a set of promising documents from the database, to be returned as *QXtract*'s output and finally processed by the information extraction system. The performance improvement can be substantial: *QXtract* allows a state-of-the-art information extraction system to extract 48% of the tuples in the target relation when retrieving only 5% of the documents in the collection, for an order of magnitude increase in efficiency at the expense of extraction completeness. The *QXtract* approach is general in that any information extraction system could be plugged and use *QXtract* as an interface to large collections, hidden web databases, or, in principle, the web at large.

More recently, Etzioni et. al introduced the KnowItAll system [14] for extracting concepts and relationships from the web (e.g., the "is-a" relationship between noun phrases). KnowItAll uses a set of predefined generic extraction rules (e.g., "NP1 such as NP2", where NP stands for noun phrase, indicating that a string tagged as NP2 in a document is an instance of a class named in NP1.). To retrieve candidate documents, KnowItAll automatically generates queries by instantiating the general patterns with the target class (e.g., for the "cities" class, a query would be "cities such as") and submits these to a generic web search engine such as Google. The returned documents are retrieved, parsed with part-of-speech tagger, and patterns applied following the general information extraction framework of Section 2. As an interesting use of web search engines, KnowItAll estimates the confidence of the extracted values by using web co-occurrence statistics via Google hit counts. Specifically, KnowItAll uses a form of pointwise mutual information (PMI) between words and phrases estimated similarly to Turneys PMI-IR algorithm [32]. PMI-IR estimates mutual information between the class name (e.g., "cities") and a proposed city instance (e.g., "Seattle") by computing web hit counts of each phrase individually, as well as the number of pages containing the phrase "cities such as Seattle". Hence, for each candidate concept or relation tuple, KnowItAll would issue at least three distinct web search queries (first to retrieve a document, and

then two more queries to compute the PMI-IR measure).

In addition to improving the efficiency of extraction, a system that queries a generic search interface might be adapted to extract relations from "hidden-web" databases only accessible via generic search interfaces [4, 18], allowing a system to process relevant documents not otherwise reachable via crawling or scanning mechanisms.

While clearly more feasible than processing every document in the collection, both *QXtract* and KnowItAll can still require days (or even weeks) to extract a large fraction of all relation tuples or concepts hidden in the collection documents. This limitation is addressed by more recent systems in the KnowItAll family, as discussed in the next section. Another shortcoming of both systems is retrieving thousands of results for each query (a functionality rarely supported by generic search engines). By removing reliance on *generic* web search engines and incorporating extraction-specific features at *index time*, it is possible to dramatically increase information extraction efficiency and scalability, as we describe next.

5 Using Specialized Indexes and Search Engines

General-purpose search engines are designed for short keyword queries and for retrieving relatively few results per query. In contrast, information extraction systems can submit sophisticated and specific queries and request many or all query results. To better support information extraction, Cafarella et al. [7] introduced the Bindings Engine (BE), which supports queries containing typed variables and some linguistic functions. For example, in response to the query "Mayors such as ProperNoun(Head(NP))", BE would return a list of proper nouns that appear in that context. To accomplish this, BE indexes the *neighborhood* of words (Figure 3 adapted from Cafarella et al. [7]).

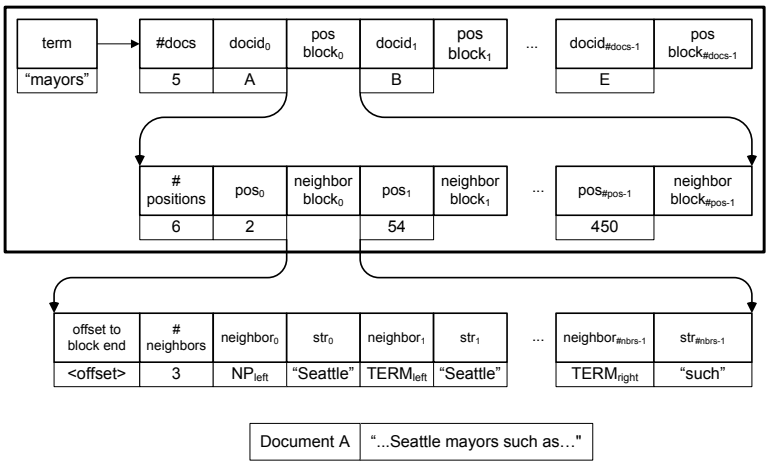

Figure 3: The BE engine neighborhood index.

The neighborhood index is similar to the inverted list index [31], but for each posting BE stores up to K words immediately to the left and to the right of each term. Additionally, BE stores all part-of-speech labels for each term (and, in principle, any other available semantic information) computed at index time. By using this expanded index, a query such as "mayors such as", which might be issued by a class extraction system for extracting names of all mayors, will retrieve the postings list for the word "mayors" and then scan the list returning all proper noun phrases that are preceded by the "such as" string. BE is well suited to extraction patterns using exact phrases (e.g., DIPRE [5] and KnowItAll [14]). As reported by Cafarella et al. in [6], the KnowItNow information extraction system and other systems in the KnowItAll family[2] use the BE search engine to quickly extract information from an indexed web snapshot of 90 million documents.

[2]Available at http://www.cs.washington.edu/research/knowitall/.

A related approach has been used for extraction-based Question Answering (notably by Prager et al. [26] and Chu et al. [10]), where a system retrieves short *answers* to natural language questions extracted at query time from the text documents in the collection. During an indexing pass over the collection, the entities predicted to be potential answers to questions are extracted and stored, and at query time only the documents (or passages) containing an entity of appropriate type (e.g., person name) are retrieved for extracting candidate answers. An intriguing new search engine was recently demonstrated by Resnik et al. [28] for indexing and searching linguistic (e.g., syntactic) structures[3] but it has not yet been formally evaluated for relation extraction or question answering tasks.

Unfortunately, word neighborhood indexing may not be directly amenable for extraction patterns without lexical items (e.g., patterns such as "Adjective ProperNoun(Head(NP))"), for patterns with only frequent words in patterns (e.g., "⟨Organization⟩ in ⟨Location⟩" [1]) or for probabilistic extraction models (e.g., HMMs [23] or CRFs [29]). Furthermore, extractors that rely on web page structures such as HTML lists (e.g., [11, 14]) still have to retrieve the complete document and apply extractors as the original *QXtract* or KnowItAll system would.

More generally, annotations such as part-of-speech tags and sentence boundaries can be viewed as adding partial structure to the text documents, which can then be represented in a semi-structured form (e.g., in XML format), and indexed for fast querying (e.g., [20]). Preliminary question answering results over annotated and indexed XML documents [21] indicate that with a rich schema and carefully constructed XPath queries it may be possible to represent question answering and information extraction as a retrieval task. We explore this idea further in Section 7.

6 Distributed Processing

So far we focused on algorithmic techniques for scaling up information extraction. Parallelization and distributed processing are attractive alternatives for processing extremely large collections, such as the billions of documents on the web. Information extraction is particularly amenable to parallelization, as the main information extraction steps, (e.g., part-of-speech tagging and shallow syntactic parsing) operate over each document independently (e.g., [13]). Hence, most parallel data mining and distributed processing architectures (e.g., Google's MapReduce [12]) might be easily adapted for information extraction over large collections.

Extracting information is only one of the steps in large scale web mining and extraction. Discovering useful document sources [3, 19], crawling (retrieving documents), extracting and indexing relevant document features, and other tasks are all required for a complete, enterprise-scale systems. IBM's WebFountain [13, 17], an influential end-to-end system, puts these steps together for information extraction and text mining from the web. WebFountain retrieves, processes, extracts and indexes information from billions of documents on the web and in local collections. The WebFountain approach includes both algorithmic and hardware solutions, and uses a heavily distributed architecture with clusters of nodes devoted to crawling, extracting and indexing web page content. WebFountain is a blackboard architecture that allows multiple *annotators* (i.e., extraction systems) to store tags (e.g., named entities) or any other annotations with each document for further processing. Unfortunately, a distributed architecture with hundreds of machines (WebFountain) or thousands of machines (Google's Map/Reduce) requires significant resources to create and maintain, which limits the applicability of this approach. As we have shown previously, it is possible to perform scalable information extraction even with modest hardware resources.

[3] Available at http://lse.umiacs.umd.edu:8080/.

7 Opportunities and Challenges

We described four general approaches for scaling information extraction to large document collections. A truism stating that "there is no free lunch" applies. Current algorithmic techniques either trade off information extraction accuracy and completeness for speed (e.g., Sections 3 and 4), or impose restrictions on the types of extraction patterns supported (Section 5). Hence, choosing the appropriate approach is heavily dependent on the application and use requirements.

One promising general approach that we mentioned earlier is to store the semantically annotated documents (e.g., with part-of-speech or named entity tags) in semi-structured form (e.g., in XML). The annotated documents could be indexed to speed up future information extraction runs. While many indexing and querying methods for semi-structured data (e.g. [20]) have been developed in different contexts, these techniques have not been adequately explored for information extraction and are a promising direction for research.

A dimension of information extraction scalability not addressed in this survey is a trade-off between domain independence and extraction accuracy. While named entity extraction technology is relatively mature and is generally accurate for common entity types (e.g., person and location names), domain-independent relation and event extraction techniques are still error-prone, and are an active area of natural language processing and text mining research. One interesting research direction is to apply probabilistic query processing techniques (reviewed in [30]) to derive usable query answers from the noisy information extracted from text.

As we discussed, redundancy and variability in large document collections can mitigate the inherent difficulty in interpreting natural language text. By operating over large collections, information extraction systems can significantly improve both accuracy and coverage of the extracted information. For this, efficient techniques for extracting information from such large document collections are crucial, and would greatly enhance our ability to manage and exploit the available textual information.

References

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries (DL 2000)*, 2000.

[2] Eugene Agichtein and Luis Gravano. Querying text databases for efficient information extraction. In *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE 2003)*, 2003.

[3] Abdullah Al-Hamdani and Gultekin Ozsoyoglu. Selecting topics for web resource discovery: Efficiency issues in a database approach. In *Proceedings of the DEXA Conference*, 2003.

[4] BrightPlanet.com LLC. The Deep Web: Surfacing hidden value. Available at http://www.completeplanet.com/Tutorials/DeepWeb/index.asp, July 2000.

[5] Sergey Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the First International Workshop on the Web and Databases, WebDB 1998*, 1998.

[6] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. KnowItNow: Fast, scalable information extraction from the web. In *Conference on Human Language Technologies (HLT/EMNLP)*, 2005.

[7] Michael J. Cafarella and Oren Etzioni. A search engine for natural language applications. In *Proceedings of the World Wide Web Conference (WWW)*, 2005.

[8] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, May 1999.

[9] Surajit Chaudhuri, Raghu Ramakrishnan, and Gerhard Weikum. Integrating db and ir technologies: What is the sound of one hand clapping? In *Second Biennial Conference on Innovative Data Systems Research*, 2005.

[10] Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, Abraham Ittycheria, and Sasha Blair-Goldensohn. IBM's PIQUANT II in TREC 2004. In *13th Text Retrieval Conference (TREC)*, 2004.

[11] William W Cohen, Matthew Hurst, and Lee S Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Proceedings of the World Wide Web Conference (WWW)*, 2002.

[12] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Sixth Symposium on Operating System Design and Implementation (OSDI)*, 2004.

[13] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Sridhar Rajagopalan Tapas Kanungo, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. SemTag and SemSeeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the World Wide Web Conference (WWW)*, 2003.

[14] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S.Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 2005.

[15] Ralph Grishman. Information extraction: Techniques and challenges. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School, (SCIE-97)*, pages 10–27, 1997.

[16] Ralph Grishman, Silja Huttunen, and Roman Yangarber. Information extraction for enhanced access to disease outbreak reports. *Journal of Biomedical Informatics*, 35(4):236–246, August 2002.

[17] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a WebFountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 2004.

[18] Panagiotis G. Ipeirotis and Luis Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB)*, 2002.

[19] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, count, and classify: Categorizing hidden-web databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.

[20] Quanzhong Li and Bongki Moon. Indexing and querying xml data for regular path expressions. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB)*, 2001.

[21] Ken C. Litkowski. Question answering using xml- tagged documents. In *The Eleventh Text REtrieval Conference (TREC)*, 2002.

[22] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[23] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning*, 2000.

[24] M. Mettler. TREC-II routing experiments with the TRW/Paracel Fast Data Finder. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, 1993.

[25] Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *Conference on Computational Linguistics (COLING)*, 2004.

[26] John Prager, Eric Brown, and Anni Coden. Question-answering by predictive annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2000.

[27] Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. The terascale challenge. In *KDD Workshop on Mining for and from the Semantic Web*, 2004.

[28] Philip Resnik and Aaron Elkiss. The linguist's search engine: An overview (demonstration). In *ACL*, 2005.

[29] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, 2005.

[30] Dan Suciu and Nilesh Dalvi. Foundations of probabilistic query answering. Tutorial at the *ACM SIGMOD International Conference on Management of Data*, 2005.

[31] Amit Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, December 2001.

[32] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *European Conference on Machine Learning (ECML)*, 2001.