Information Extraction in the Business Intelligence Context

Luiz Gomes, Frank W. Tompa

August 28, 2010

1 Introduction

Business Intelligence (BI) relies on corporate data to derive information for supporting strategic decisions. Source data internal to the corporation are used to assess productivity, lucrativeness, quality, etc, whereas external data are used to evaluate market share, expansion opportunities, competitiveness, etc. In a typical BI scenario, structured data, usually gathered from application databases, are integrated in Data Warehouses to support analytical tools. There is, however, a wealth of unstructured data which is harder to integrate but can provide valuable information: formal documents, reports, exchanged messages, and numerous websites on the internet.

Information Extraction (IE) is one of the main players in harvesting the unstructured information embedded in digital documents. Researchers have developed a wide range of techniques to surface structured information from virtually all types of documents.

Access to structured information enables richer analysis of data, which is the main goal of BI. This document describes research in IE and typical approaches to BI.

This document is not intended as a survey on IE theory or techniques – for that we recommend the papers by Sarawagi [46] and Nadeau & Sekine [36]. Nor is the document intended to cover a broad range of BI issues, which are detailed in the books by Imhoff et al. [24] and Jarke et al. [28]. The objective of this document is to present typical approaches to IE and BI, to describe research that integrates the two areas, and to provide some insight on what may improve the applicability of IE techniques in BI scenarios.

Section 2 presents IE research, analyzing techniques according to the type of input document, the dimensionality of the target extraction, and the variations on the typical IE workflow. Section 3 describes stablished BI technology and approaches to leverage documents in a BI scenario. It also suggests in which areas IE systems could be adapted to facilitate their use in BI tasks. Section 4 concludes the paper.

2 Information Extraction

Information Extraction refers to the automatic extraction from unstructured sources of structured information such as entities, relationships between entities,

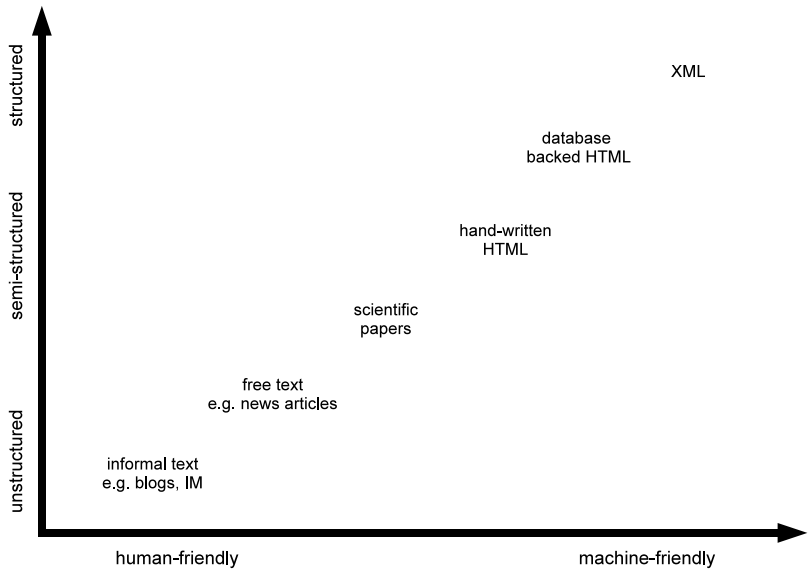

Figure 1: Structure spectrum [12]

and attributes describing entities [29, 46].

The following subsections describe distinct aspects of IE, starting by listing the diverse types of input documents and the techniques used to process them, then focusing on IE from unstructured text and its respective techniques, and finally describing the typical architecture of an IE system as well as variations found in the literature.

2.1 Structure spectrum and respective approaches for IE

The possible inputs for IE tasks range across a variety of types of documents, including news articles, research papers, emails, social media sites, personal web pages, and e-commerce websites. Those are all unstructured documents, meant for human consumption, which makes them hard to be processed by a machine. Nevertheless, all of them have some kind of embodied structuring information, otherwise they would be inaccessible even to humans. Free text as in news articles are written based on grammar rules, research papers have headers and sections, and web pages have HTML markup. Therefore, depending on the kind of document a given IE task is dealing with, there will be different levels of structuring information to be exploited. This section describes various IE tasks – based on the structural nature of the documents – and the respective techniques to deal with them.

Figure 1 shows the IE structure spectrum, correlating the level of structuring information in the input documents with the difficulty of automatic interpretation of their content. Depending on the type of input documents, the IE system has to apply appropriate techniques to exploit the structuring information available. IE from free text must rely on Natural Language Processing (NLP) techniques, whereas web pages can be processed with wrappers and pat-

tern matching algorithms. Over the past decade there has been a revolution in the use of statistical and machine-learning methods for IE [33]. Those techniques are being successfully applied to many IE tasks, all along the structure spectrum. The following subsections analyze the problems and techniques associated with different levels of the structure spectrum.

IE from semi-structured web pages

The advance of the web has allowed the research on IE to exploit more structured types of documents, especially web pages [12]. IE from web pages can rely on syntactical patterns or layout structures of the documents. In this context, the programs that perform the IE task are referred to as wrappers. Wrappers enfold information sources, filtering and integrating the target information.

Wrappers are based on sets of extraction rules to match and retrieve the desired information. Many wrapper generation systems allow users to specify extraction rules based on a declarative language. However, the structure of documents on the web is complex and contents change periodically, making the specification and maintenance of those rules problematic [29]. To alleviate this burden, researchers have focused on developing systems capable of automatic or semi-automatic wrapper generation.

Semi-automatic wrapper generation systems usually provide an interactive user interface where users specify target pages and the information to extract. Based on the user input, the systems create the extraction rules and compile the wrapper program. XWrap [31] has a further debug step where the user follows the extraction process and is able to fine tune the rules. The Lixto system [6] uses a datalog-like language, *Elog*, as its internal rule based language.

RoadRunner [15] addresses data-intensive websites, that is, those that are generated from a back-end database. This type of website has a more regular structure, enabling an automatic extraction procedure. RoadRunner iterates over the pages – two at a time – refining its extraction rules based on the similarities and dissimilarities. By the end of the process, the fields are defined and the information extracted. The user, however, still needs to label the fields to make sense of the data.

IE from semi-structured page elements

The web contains a vast amount of semi-structured data embedded in documents in the form of lists and tables. Identifying and cleaning these data sources has been an active research area.

Cafarella et al. [10] present OCTOPUS, a system aimed at helping users to find relevant tables and manipulate them according to their information needs. The system assumes the existence of a large repository of tables extracted and cleaned from a web crawl. A user begins with the operator *search*, providing a keyword query, to find relevant tables. The selected table can then be manipulated using the operators *context* and *extend*.

The search operator ranks tables according to their relevance to the keyword query. The tables are then clustered so that similar tables are grouped together. The idea is that tables in the same cluster should be similar enough to enable a union operation with little or no extra effort.

The context operator augments tables with attributes that are represented as free text in the original document. For example, a document that displays the universities in Ontario as a table is more likely to have the string "Ontario" in its header, not repeated in each row of the table.

To expand tables with new columns the users apply the extend operator. It takes a topic keyword and a join column as input, searches for joinable tables relevant to the topic, and returns a new table that results from joining the original and the relevant tables on the input join column.

Gupta and Sarawagi [22] tackle table augmentation from lists on the web. The user provides a few seed tuples, and the system, with no supervision other than the seeds, retrieves relevant lists, segmenting and integrating their records. The data source is a search engine that indexes as individual documents the lists retrieved in a web crawl.

The authors use the seed tuples to derive cell and row resolvers. Cell resolvers try to identify the correct column for an arbitrary text segment, while row resolvers try to identify matching records. The resolvers are used to produce training data for the Conditional Random Field (CRF) [30] segmentation model as follows:

The system uses the seed tuples to query the search engine retrieving relevant lists. It then attempts to match the seeds in the top ranked lists. For every match, the system segments the text and assigns segments to columns. The generated information is the training data will be used by the extractor in subsequent steps.

After identifying new records, the system has to consolidate them into the original table. To accomplish that, the system creates improved cell resolvers by training a binary classifier. The new cell resolver is in turn used to create a new row resolver, which is a Bayesian network that captures the interactions between cell-level and row-level matching decisions.

The last step is the ranking of the consolidated records. The authors exploit the confidence score returned by the CRF extractors and add some heuristics that try to capture the importance of a column and consequently the importance of each record in respect to the original query.

IE from Natural Language Text

The bulk of the digital text created and published is still highly unstructured: news articles, reports, emails, and blogs are all examples of content intended for human consumption. Automatic text understanding has long been the ultimate goal of NLP, promoting the development of various techniques to elicit information from natural text. IE, a subtask of NLP, employs many of these techniques to identify structure in unstructured text.

Grishman [20] defined IE in this context as "the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship".

The Message Understanding Conferences (MUCs) were initiated by DARPA to assess and foster research on the automated analysis of military messages containing textual information [21]. The conferences started in 1987, and there were seven of them until 1997. The tasks changed from one conference to the next, with topics ranging from military reports of fleet operations to news reports on negotiation of labor disputes.

FASTUS [4] is a typical system from the early days of IE from natural-language text. It employs a nondeterministic finite-state language model and processes text in four phases. In the first phase, trigger words are identified and full names are indexed. The second step recognizes phrases, especially noun and verb groups. These phases are responsible for matching and merging the patterns that fill the templates with the extracted information. FASTUS is a simple and, for its time, effective system. However, it has to be adapted to each intended task, with specific trigger words and patterns.

WHISK [50] is a more advanced IE system. It employs machine learning techniques to learn the patterns to be extracted from training data. It also guides the training process to minimize the number of tagged documents needed. It works with unstructured and also with semi-structured text. For unstructured text, it produces better results when text is previously annotated for syntactics and semantics.

In order to harvest information from text in a web scale, the researchers focused on more restricted tasks, extracting simple facts or relationships. Section 2.2 describes fact extractions systems, which represent the current state-of-the-art in terms of large-scale information extraction from natural language text.

Any extraction process has to rely on some level of structural information in the source corpus. Typically, that structure comes from grammar or formatting (e.g. HTML tags). There is, however, an important group of documents that defy extraction by not encoding substantial grammatical nor formatting information. Examples of these are online classified ads (e.g. Craigslist) and auction sites (e.g. eBay).

Michelson and Knoblock [35] overcome the lack of structure by exploiting a pre-defined reference dataset for the application domain. For example, to extract information from personal posts of cars on sale, the authors employ a standard dataset of car models. The source posts are matched against the reference set based on a Support Vector Machine classifier modeled with multiple string similarity features. To constrain the search space, the authors apply blocking techniques on the attributes of the reference set. The tests for matching are then restricted to candidate records in a matching block. After finding a matching record, the tokens in the post are aligned with the tokens of the record. The alignment of tokens drives the extraction process. A simple data cleaning procedure follows the extraction, eliminating misspellings and tokens that only introduce noise.

Statistical Machine Learning Techniques

Building an IE system is challenging. The system typically must rely on subtle clues to pinpoint the target information in the text portion, there are many exceptions to deal with, and it is hard to predict every possible future situation that the system will be facing. This is indeed a typical scenario for applying Machine Learning techniques, allowing those subtle clues and exceptions to be learned from labeled training data. These techniques can be applied to any part of the structure spectrum, and this is the main focus of research today.

Some subtasks in IE can be addressed as classification problems, which enables general methods for classification to be employed. The most common approach in IE is to use Support Vector Machines (SVMs). SVMs [16] map input vectors to a high dimensional space and construct a hyperplane that separates

classes. Although SVMs have been successfully applied to IE [26], the approach is not straightforward and demands a considerable amount of engineering (see [23]).

Hidden Markov Models (HMMs) [41] are one of the most widely used models for IE. A HMM defines states and outcomes. Transition probabilities account for transitions between states and emission probabilities account for emitted tokens. In a segmentation task, field labels are states and the outcomes are the tokens from the text to be segmented. The probabilities are learned from training data using Maximum Likelihood. It is a simple and yet powerful model that can capture prior domain knowledge. Examples of application of HMMs to IE are in the works of McCallum & Freitag [34] and Seymore et al. [49].

Generative models such as HMMs make strong independence assumptions, constraining feature definition. Conditional models, such as Maximum Entropy Markov Models (MEMMs) [8] and Conditional Random Fields (CRFs) [30] relax these assumptions, allowing features representing attributes at different levels of granularity to be used simultaneously [30]. The capacity of capturing more features boosts the performance of those methods in IE tasks. CRFs have been applied in many IE tasks and are the state of the art in many of them [33, 38, 54].

Lately, research has been focused on the development of frameworks that provide a more flexible way to define complex probabilistic networks. The goal is to enable the use of long-distance relations among entities, enabling joint and collaborative approaches for IE. The most important techniques in this area are Relational Markov Networks (RMNs) [52] and Markov Logic Networks (MLNs) [43], which have been used to build many collaborative and joint models for IE [40, 9, 58].

2.2 Climbing the text understanding ladder

When there are few formatting clues and little metadata to rely on, the automatic processing of the documents must start with simple grammatical and morphological patterns, leveraging the findings in an incremental cycle towards a higher level of understanding of the contents [20].

Figure 2 arranges diverse NLP subtasks in terms of their computational difficulty and the kind of analysis they apply. Lower in the graph the techniques tend to focus on local, easy to compute characteristics of the text, for example, a word can be classified as a verb based on the presence of the suffix "ing". Techniques up in the graph need to process bigger chunks of text, correlating distinct elements of the document.

IE is a subtask of NLP, sharing many of its objectives and techniques. IE has less ambitious goals, not requiring a strict understanding of the text, thus allowing the use of simpler, more efficient techniques.

The next subsections are dedicated to techniques usually associated with an IE scenario.

Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying references to real world entities in the text. Typical examples are person's names, organizations, and addresses. Other types of references such as dates and monetary values with no clear corresponding real world entity (e.g. Monday, $25,000) have also

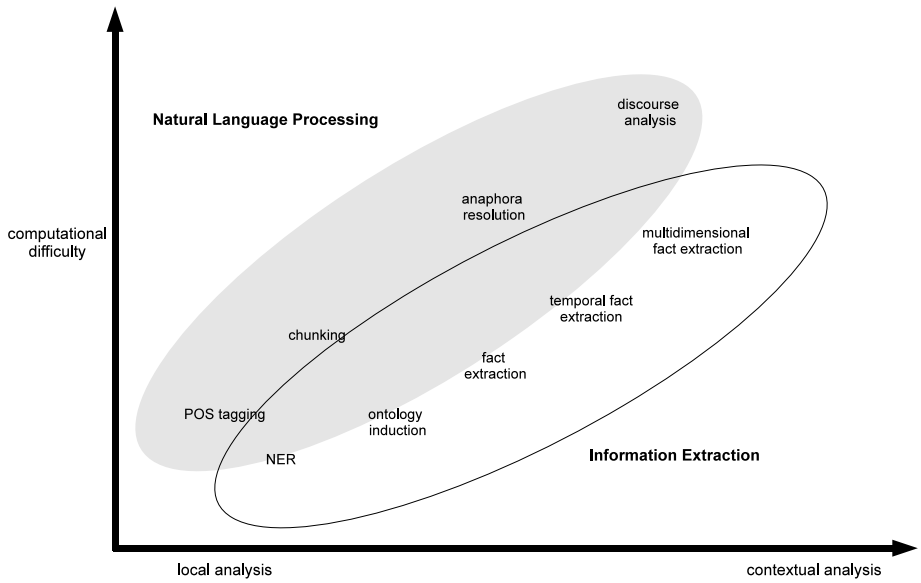

Figure 2: NLP and IE approaches for increasing contextual awareness

been tackled by NER techniques. Sekine and Nobata extend the class hierachy of entities to 200 types [48]. Nadeau and Sekine [36] survey the research in NLP.

The two main approaches used in NER are hand-crafted rules and statistical learning. A simple rule to identify persons' names is to look for the title "Mr." followed by capitalized words. A learning-based approach aims at learning such rules automatically from annotated training data. Rule-based approaches are computationally efficient, but may miss out on many subtle rules and are hard to adapt to new domains. Adapting a learning-based system to a new domain requires no changes in the algorithms, just adequate training data.

Both rule and learning-based systems require the identification of features that characterize references to the entities. Features can be harvested from several characteristics of the text, such as word morphology, grammar, and sentence context. An adequate choice of features can be as important as the choice of technique to be used [45].

NER systems often employ lists of terms (or dictionaries, gazetteers) to produce features. For example, the presence of a word in a list of common first names is highly indicative of a reference to a person and can be used as a feature. Lists of stop words, dictionaries of names, and gazetteers of places are examples of resources used in several NER tasks.

List lookups also define an entire subtask in NER: when the names of the entities are known beforehand, the system has to match the names in the text. The main challenges in this case are (i) references to entities can diverge from the canonical name in the list, and (ii) names vary in number of tokens, which explodes the number of candidate substrings to be tested in each document.

Ananthanarayanan et al. [3] address the first problem in the context of product names. The authors encode rules that generate synonyms to expand

the dictionary of products. The rules are domain-specific, deriving synonyms by removing common prefixes or suffixes and extracting acronyms. Efficiency does not seem to be critical in the scenario, and the authors do not elaborate on their strategy to match the documents against the expanded dictionary.

Capturing the rules that generate synonyms is a challenging task, especially considering the uncontrolled web environment. Chaudhuri et al. [13] also address the product name domain, but exploit a document corpus to identify variations on how the entities are mentioned. The authors rely on the observations that (i) terms are often dropped in product mentions (e.g. "Microsoft Office 2000" vs. "Office 2000"), and (ii) the dropped terms are often mentioned in other parts of the document so that the shortened mention can be disambiguated.

The idea behind the algorithm is, for each entity in the reference table, to find mentions omitting some tokens and to assess their correlation with the original entity based on the presence of the missing tokens in other parts of the document. The authors aim at employing a very large corpus (tens of TB) to extract the synonyms. To cope with the load, the system employs indices for candidate synonyms and partial results. Furthermore, to minimize the number of string comparisons, the authors develop a filter that first prunes the substrings that cannot be candidates.

Fact extraction in the web

IE began receiving substantial attention in the early 90's. At that time the researchers, highly influenced by NLP ambitions, focused on deep analysis of text, identifying complex grammatical patterns to infer meaning from the writings. The algorithms had to be adapted to each specific task, text processing was computationally intensive, and results did not meet quality expectations.

Later expansion of the web and improvements in hardware enabled a new brute force approach, aimed at extracting simple, easy to identify facts, and relying on information redundancy to assess quality of the extracted information. This new approach to web data extraction has been termed Web-Scale, Large Scale, or Open Information Extraction.

Etzioni et al. [19] proposed KnowItAll, the first system capable of extracting facts in large scale from the web. The system requires a taxonomy of concepts on which it bases the retrieval of documents from search engines for further instance extraction. The extracted facts are associated with a probability value, a means of assessing the support for that fact in the corpus. The system's reliance on part-of-speech tagging and the iterative retrieval of documents from search engines have a big impact on the performance, requiring days to extract thousands of facts.

The systems that followed the success of KnowItAll made significant advances on scalability. Pasca et al. [37] propose a system that takes as input a few sample pairs that encompass a "hidden" relation – such as (Athens, Greece). The seeds are used to extract context words in the corpus that are associated with the entities in the seed. The context words are further generalized by replacing individual words by their classes of similar words. The generalized patterns are then applied to the corpus to extract millions of facts.

TextRunner [5] employs an unsupervised learner to classify its own training data, based on linguistic knowledge. The authors apply a deep parser to

a small sample of the corpus, identifying target tuples based on heuristics for the generated syntax tree. The tuples extracted in the first step become the training data that is then used to train a Naive Bayes classifier. The classifier, more efficient than the parser-based approach, is used to extract instances from the entire corpus. Like KnowItAll, the extracted facts are associated with probability values that reflect their support in the corpus.

Most fact extraction systems focus on extracting facts explicitly mentioned in the text. Cullota et al. [17] go one step further, analyzing the relational paths formed between the entities. This analysis enables the discovery of implicit facts in the text, for example, the fact "cousin of" can be inferred whenever the relational path "mother-sibling-son" is found to be true between two entities. These relations might also reveal trends such as the fact that children are likely to attend the same college as their parents. The authors use this relational information to improve the extraction process, adding the relation paths as features in the training of their CRF-based extractor.

Higher order relationships

Most fact extraction systems focus on binary relationships. This approach is good for immutable and independent facts, such a person's birthday, but does not capture the nuances of ephemeral facts (e.g. the host of the olympic games) or of facts that require contextual information (e.g. total sales for a large company requires at least a time period and a geographic boundary).

Zhang et al. [57] develop techniques to identify the temporal validity of facts. They employ the deep grammatical parser of Laila [51] to locate prepositions that indicate time constraints on the facts. To cope with the uncertainty of the task, the start and end times are represented by intervals. The authors also extract document annotations to identify publication dates, enabling the grounding of references such as "last month".

Wick et al. [55] propose a task that they called *record extraction*. The goal is to group related entities in a documents into records. This allows, for example, the composition of a contact database from personal web pages. The authors employ the notion of field compatibility to cluster the entities. The field compatibility features are domain specific and have to be provided by he user. The system is trained to learn the parameters that highlight the most relevant features.

2.3 Typical Anatomy of an IE system

Figure 3 shows the typical architecture of an IE system. Most projects follow a similar flow, where users specify the extraction task ((a) in the figure), data is extracted from source documents ((b) and (d)), and stored in a database (c). There are, however, many variations of how each step is specified. The next subsections describe the typical approaches for each part of the architecture and highlight systems that propose new strategies.

User input

As in any information system, the user interface in an IE setting must allow users to express their information needs. Often, systems also require users to

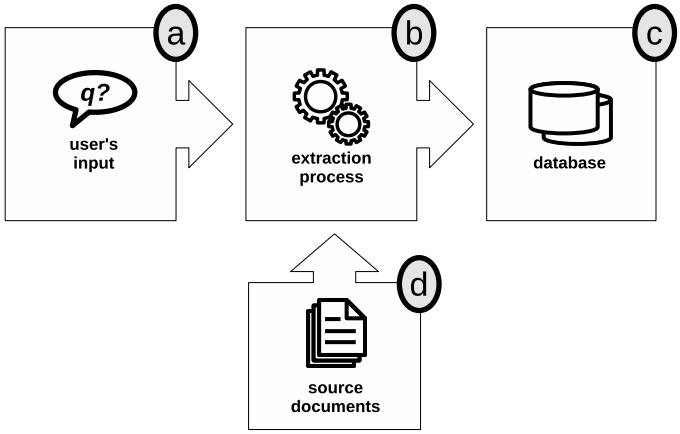

Figure 3: Typical architecture of an IE system

input artifacts that aid the extraction process, such as seed facts or tagged documents. Seed facts enable the system to harvest mentions of the fact in the corpus and learn the correlated patterns [37, 42]. Annotated documents are used by machine learning approaches in the training of the extractors [32, 56, 33].

Jain et al. [27] enable *online processing* of simple SQL queries (selection-projection-join) over the text database. The relations are pre-defined but the tuples are extracted on demand, based on the query. The system optimizes query executions, selecting good document retrieval and extraction techniques.

In most IE tasks, the target schema is fixed, specified as input. Some systems require further input to guide the extraction process. The system proposed by Mansuri and Sarawagi [32] has no fixed schema. Instead, the user indicates a target database, complete with schema, constraints and data. The contents of the database is used throughout the extraction process, complementing training data, defining constraints, and aiding data cleaning.

Kylin [56] exploits Wikipedia articles as input to bootstrap IE. The goal is to augment incomplete infoboxes (summary tables containing attributes of the page's subject) with information extracted from text. The authors construct a class hierarchy for Wikipedia subjects, using class members to determine relevant attributes. The authors then identify sentences in pages that mention attributes present in its respective infoboxes. Those sentences are used to train statistical models that can extract values from pages that do not have the equivalent infobox attribute. The authors also employ pages retrieved from the web to extract complementary facts, increasing recall.

The proposed system relies on the assumption that attribute values present in the infoboxes are likely to be mentioned in the main text of the article. This is very important in order to enable automatic generation of training data, the biggest challenge in statistical learning techniques. As described in the experimental reports, Wikipedia does have enough of this kind of redundancy. The authors, however, do not discuss the applicability of the proposed techniques in other contexts.

Users can also be more involved in the extraction process. Chu et al. [14] face the extraction as an incremental and interactive process where the system

and users cooperate in discovering the structure hidden in the text corpus. The authors propose a relational workbench consisting of three operations: extract, integrate, and cluster, whose specific algorithms must be chosen by the DBA on a convenience basis. The schema evolves as a result of the execution and analysis of these operations.

In the proposed scenario, a new text corpus is stored in a table, one row per document, and users can issue keyword-based queries straightaway. New attributes are then discovered by the extract operator; the attributes are correlated by the integration operation; and common properties among documents are derived from the cluster operation. There is no specific order for the execution of the operators and the structures discovered can either be reflected in the table or fed as input to another operator. The authors employ a special table structure, meant to support the high degree of sparseness of attribute values.

Extraction

The strategy employed in the extraction phase is influenced by several factors: (i) the nature of the input document, and how much structure is available, (ii) the desired output, such as simple facts or higher level relationships, and (iii) the scalability requirements. Typical extraction techniques and variations have been described in Sections 2.1 and 2.2.

Source documents

The traditional approach to information extraction applies the extraction algorithm to every source document, maximizing recall. However, the extraction algorithms are computationally expensive and the number of input documents tend to be large. One major goal for extraction optimization is, therefore, to reduce the number of input documents to be analyzed. This can expedite the process even to the point of enabling online extraction, but at the expense of a potentially lower recall.

The first approaches to selective document extraction focused on deriving filters to select the documents likely to contain instances for a given extraction task. The filtering process can be based on pattern matching or keyword search. Agichtein [2] reviews diverse approaches in this area.

Ipeirotis et al. [25] analyze three different text-centric tasks – including information extraction – together with the available strategies to tackle their document selection phase. The goal is to propose a unifying cost-based optimization scheme. Four document selection strategies are considered: (i) scan, (ii) filtered scan, which uses a classifier to determine useful documents, (iii) iterative set expansion, which uses pre-defined seed tokens to find useful documents, deriving new tokens for subsequent iterations, and (iv) automatic query generation, which extract rules from a trained classifier to construct queries that return potentially useful documents. The authors define a cost model that covers the different strategies, which enables the selection of an efficient plan based on a given target recall.

Jain et al. [27] focus on information extraction and advance previous work [25] in two main aspects: (i) considering multiple extraction systems to be chosen by the optimizer, and (ii) enabling *online processing* of simple SQL queries over the text database. Online processing changes the traditional workflow of

information extraction, where the entire corpus must be processed to extract all relations before any query can be issued. The proposed approach avoids processing and extracting tuples that may never be used, and enables more efficient monitoring of entities in an evolving corpus.

The framework proposed by Jain et al. also considers data cleaning of the extracted tuples in the query process. To account for a user's quality requirements, instead of fixing a target recall, the user must define a *goodness* value that captures the intended balance between efficiency and recall. The main problem in this scenario is to estimate the statistics needed for query optimization. The experiments showed that the heuristics proposed in the paper provided good estimates for query optimization.

Output

The typical IE system starts with a fixed schema and the expected output are tuples that comply to the schema. Information encoded in a corpus of natural language documents is however intrinsically unbounded. Having a fixed schema limits extraction to the pre-defined fields. For many applications, it is also unlikely that all relevant attributes can be defined a priori. Some systems have addressed the problem of fixed output.

Cafarella et al. [11] propose an algorithm for automatic schema discovery from extracted facts. The input for the algorithm is composed of facts extracted by a standard named entity and relationships extraction system (e.g. TextRunner [5]). The goal of the algorithm is to arrange attributes and values in a set of tables that conform to application-specific parameters. The adjustable parameters encode desirable proportions of the resulting tables (width and height) as well as their attribute value density; for example, a typical data browsing application requires a manageable number of attributes (to fit in the screen) and a high density of attribute values. In the application scenario of the proposal, no prior knowledge about the text corpus is assumed. It is up to the user to analyze and make sense of the resulting unnamed tables.

The approach presented by Ravi and Pasca [42] envisages a more focused discovery task, in which the user knows the target class for extraction and agrees to provide some information to bootstrap the process. The input of the system consists of the name of the target class, the target instances (that can be generated by an external automatic process), and a few seed attributes. The objective is to find relevant attributes of the class as well as their respective instance values. Unlike the approach proposed by Cafarella et al. [11], the text corpus is composed of semi-structured web documents and the extraction process is driven by HTML formatting clues in the documents. Attributes are selected based on the similarity of their hierarchical structure pattern (basically, the XML ancestor sequence) and the patterns found in the reference seed attributes.

The extraction process does not need to be unidirectional, always from the input documents towards the extracted tuples. Information from the underlying database can improve accuracy of the extraction process: tuples in the database can help disambiguating the extraction of new tuples, and the data constraints in the schema can help distinguishing correct tuples for extraction. Moreover, existing relational data can be used to generate training data in learning-based approaches.

Mansuri & Sarawagi [32] exploit clues in the database to augment the CRF extraction model, reducing the dependence on manually labeled training data. The proposed architecture encompasses the extraction and matching processes.

For the extraction phase, the authors employ a semi-Markov CRF [47] model that defines features over multi-word segments of the text (as opposed to single tokens used in typical CRF models). Sequences are a more natural abstraction for this labeling task, yielding higher accuracy. The authors exploit instances present in the database to derive many features used in the model: word similarity (i.e. whether the word has similar words as attribute values in the database), pattern similarity (e.g. is the capitalization similar to that of the values of an attribute?), and other less important factors (such as typical length of entities in the database) that are considered together to model an entity classifier.

Mansuri & Sarawagi also extend the Viterbi inference algorithm (typically used in Markov models) to support the cardinality constraints expressed in the schema. Instead of keeping track of the label sequence of maximum probability, the algorithm has to keep track of multiple sequences since now any sequence that conflicts with the cardinality constraints must be discarded.

For the matching phase, the authors design a CRF classifier based on multiple string similarity measures. Variations of the same entity (e.g. J. Ullman and Jeffrey Ullman) are preserved, with the different occurrences linking to a canonical version (choosing the canonical version is out of the scope of the paper). The matching model also leverages the relational constraints so that the matching of a given record takes into account associated records from other tables.

3 Business intelligence and unstructured data

Companies, governments, and scientists have to make decisions based on precise information. The right information is, however, often difficult to get to: typically, data is spread around distinct systems that use incompatible formats. Moreover, often the amount of information required by a task is overwhelming and needs to be aggregated and summarized before analysis. These issues have driven the development of data warehouses and data analysis techniques, especially in the area referred to as Business Intelligence (BI).

BI infrastructures must accommodate data from diverse data sources, which creates several data integration issues. Especially challenging is the use of data from documents. Some research has been done on leveraging document information in BI tasks and on employing BI techniques for document retrieval.

3.1 Traditional BI

A data warehouse is an integrated, non-volatile aggregate of multiple data sources. The main challenges in creating a data warehouse are (i) retrieving and consolidating the heterogeneous data from the different sources (ii) organizing the data to enable efficient analysis.

Data retrieval and consolidation is managed by the industry in the context of ETL tools, in charge of Extracting, Transforming, and Loading the data into the data warehouse. An important related research problem is that of data cleaning.

The temporal aspect of a data warehouse is critical. A data warehouse must maintain historical data even though the source databases often do not implement this notion. Therefore, time attributes have to be added to the integrated schema [24].

The multidimensional model

Data analysis is the ultimate goal of business intelligence, but traditional database tools and models fall short in providing convenient means to deal with the large data collections in a data warehouse.

A more convenient data model for analysis is the multidimensional model. It consists of an n-dimensional array that represents the data values along distinct attributes. Dimensions offer a concise, intuitive way of organizing and selecting data for retrieval, exploration, and analysis [28].

In this model, the data in a data warehouse defines one or multiple "cubes" (also called hypercubes) [28]. Each cube refers to measures (also known as variables or metrics) of a fact of interest (e.g. sales for a retail store). Each dimension in the cube represents a different aspect that contextualizes the fact (e.g. time period or the region of the sales). Furthermore, the dimensions are organized as hierarchies that enable aggregation of values on their levels (e.g. sales can be aggregate by month, quarter, or year in the time period dimension).

Data manipulation in the multidimensional model is referred to as OLAP (OnLine Analytical Processing).

OLAP

OLAP implementations allow interactive data exploration, enabling users to manipulate the data cube, selecting the desired portion of the data and aggregating results along the dimensions. The most common operations present in an OLAP implementation are:

Roll up – used to navigate upwards in the hierarchy of a dimension, aggregating the values (e.g. aggregating monthly sales by quarters);

Drill down – opposite of roll up, breaks down the values according to a more detailed level of a dimension;

Slice – fixes a value for a given dimension, reducing the dimensionality of the cube;

Dice – selects a single measure from the cube, as if slicing in all dimensions;

Pivot – rotates the cube to change its perspective.

There are two main approaches to implement the underlying infrastructure that enables the multidimensional model: MOLAP and ROLAP [28]. MOLAP (Multidimensional OLAP) employs a special purpose multidimensional database system, mapping OLAP operations directly to the database API.

ROLAP (Relational OPLAP) is a common approach that leverages pre-existing relational infrastructures. It maps OLAP operations to SQL queries that are processed by a traditional relational database.

A major concern with the ROLAP approach is query performance: the relational model and the hierarchical nature of the dimensions require the composition of queries that perform several joins. To minimize the number of joins in the queries the data is stored according to special schemas that aim at superimposing a multidimensional structure upon the two-dimensional relational model [28].

The *star schema* is composed of one fact table and multiple dimension tables. The fact table contains the measures that defines the contents of the cubes (e.g. total sales or total profit for the sales fact). The dimension tables contain the unnormalized attributes of the dimensions (e.g. the time period table would have attributes like year, quarter, month and so on). The unnormalized dimensions minimize joins but imply data redundancy that consumes extra space and that is hard to maintain.

In certain scenarios a *snow-flake* schema is more appropriate. The snow-flake schema normalizes the dimension tables, simplifying updates in the hierarchy of the dimensions. Another option is the *galaxy* scheme, that shares dimensions among multiple star or snow-flake fact tables, promoting reuse and simplifying management.

Multidimensional queries

The user interaction in an OLAP system has been vendor specific, with no standardized querying mechanism. The XML for Analysis initiative is working towards the development of query standards for OLAP databases.

XML for Analysis (XMLA) is the industry's current attempt for a standard OLAP API. The proposal employs HTML, SOAP, and XML web standards in the communications. XMLA specifies XML schemas that contextualize the queries [1]. The multidimensional queries are encoded with mdXML, which is basically the MDX (Multi-Dimensional eXpressions) language developed by Microsoft.

mdXML is an SQL-like declarative language for multidimensional data. The concepts of tables, columns and rows of the relational model are replaced by cubes, axes and measures. The result of an mdXML query is a data cube. mdXML also includes a rich set of functions for performing statistical analysis.

The simplified mdXML interface model has two methods. The Discover method obtains metadata, such as the list of available data sources on a server or details about a specific data source, and the Execute method sends action requests to a server, such as mdXML queries to retrieve data or update statements to modify it[1].

A basic mdXLD query is composed of SELECT, FROM, and WHERE clauses. SELECT specifies the dimensions that compose the result cube; FROM defines the cube (fact table for ROLAP) that contains the data; and WHERE defines the slicer dimension.

Figure 4 show an example of an XMLA query. It returns the total sales and total cost for all stores in the US in the years 2007 and 2008. COLUMNS and ROWS are convenience aliases for the first two axes of a cube (PAGES being the third). Measures are considered as dimensions when used to define axes.

```
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV=
   "http://schemas.xmlsoap.org/soap/envelope/"
   SOAP-ENV:encodingStyle=
   "http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
 <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle=
  "http://schemas.xmlsoap.org/soap/encoding/">
   <Command>
    <Statement>
      SELECT
        {[Time].[2007],[Time].[2008]} ON COLUMNS,
        {[Measures].[Sales], [Measures].[Cost]} ON ROWS
      FROM Sales
      WHERE  ([Store].[All Stores].[USA])
    </Statement>
   <Command>
   <Properties>
  <PropertyList>
     <DataSourceInfo>
               Provider=Essbase;Data Source=local;
     </DataSourceInfo>
     <Catalog>Foodmart 2000</Catalog>
     <Format>Multidimensional</Format>
     <AxisFormat>ClusterFormat</AxisFormat>
    </PropertyList>
   </Properties>
  </Execute>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 4: Example of an Execute method call with $< Statement >$ containing an OLAP MDX SELECT statement

3.2 Unstructured data and Business Intelligence

Although text documents may contain valuable information for most BI tasks, it is hard to integrate unstructured data in the analysis process. Researchers have explored two main approaches to integrate text documents and BI: (i) redefining the cube model to encompass documents and their metadata, and (ii) applying IE techniques to obtain data for specific BI tasks.

The first approach documents are accessed in an information retrieval setting enhanced with a BI-like modelling that enables multidimensional queries. There is no attempt to process textual information to derive structured facts. Tseng and Chou [53] define dimensions for document data that can encompass document metadata (such as author, subject, publisher, etc), keywords, summaries, and category classification. Summarization and keyword extraction are automated in the proposal. The only measure in the fact table is a default count of the documents.

The approach taken by Perez-Martinez et al. [39] focuses on the integration of a (XML) document warehouse and the traditional data warehouse. The goal is to identify references to the facts of the data warehouse's fact table in the text content of the document warehouse. To define a query, the user provides the usual multidimensional expression for the data warehouse together with (i) an XPath expression filtering the intended nodes in the documents, and (ii) an IR-like keyword query describing the context of the analysis. The relevant documents are associated with the facts in the OLAP cube, allowing users to investigate documents that might explain the data.

IE in BI tasks

A more data-driven approach to the integration of documents in the analysis process is to apply information extraction tools to handle specific data needs. If the user knows precisely what information is required and where it can be found, IE extraction systems can be adapted to suit the needs. This approach can be thought of as a simple extension of typical ETL (Extract, Transform, Load) workflows, having documents, as opposed to the usual structured data sets, as data sources.

Baumgartner et al. [7] present applications of the Lixto System [6] in business intelligence scenarios. The mechanisms in the system mimic standard ETL tools, facilitating usage by data warehouse professionals. The extraction process begins with the specification of the wrappers, through the use of an interactive user interface. The wrapper agents are then embedded in the runtime environment of the transformation service, where the extracted data can be aggregated, re-formatted and delivered. The authors showcase two applications for monitoring product prices in the market. There is no mention of specific requirements that distinguish the BI scenario from a traditional IE task.

Saggion et al. [44] adapt the GATE language engineering system [18] to handle business intelligence tasks. The authors employ ontologies to represent the target concepts. The paper discusses techniques to address extraction tasks for company profiles, country and region facts, and financial statements. Only the first task yielded results deserving a performance analysis section. There is no reference of whether the adaptations are specific to the business intelligence domain or if the system could be used in other scenarios.

The obvious problem with these proposals is the need to reconfigure the system for every analysis task. The adaptations may require domain and IE experts, rendering this type of solution impractical.

3.3 IE and BI mismatches

IE and BI have similar goals, providing means to improve data analysis. IE can be seen as part of a text-based ETL process. We see, however, many aspects of the typical IE system that could be improved to facilitate its use in BI tasks. The first aspect corresponds to the dimensionality of the facts extracted in IE systems.

While fact extraction systems often deal with facts that are typically discrete and static (e.g. Einstein was born in 1879), BI applications require the analysis of fact evolution according to various dimensions. For example, a user may need to analyze the population growth of France in the past five years, and then compare it to the growth of the European population in the same period. The IE system must be able to extract dimension-specific facts accordingly.

Another problem with current IE extraction systems in a BI context regards user interaction. Often, the only way the user can influence the extraction process in IE systems is by providing the seeds that bootstrap the extractors. After the facts are extracted, the user can either pose relational queries or resort to a query answering system, depending on the available infrastructure. However, a BI workflow is more task-specific, and the interface should allow exploratory queries over the data. Moreover, BI users typically work at the document level or higher, analyzing and producing reports. The fact level is, therefore, too fine grained and low level for most of the tasks.

Finally, there is the problem of IE's dependency on a fixed corpus: the main IE systems extract facts from either the internet, posing queries to search engines, or from an internal corpus, typically the result of a web crawl. In a BI scenario, there is no unique corpus that can provide all the needed data. For example, the user may need to compose data available in the company's internal documents with data from various industry publications, a government website, or other data gathered on the web. The extraction system must be able to identify the best source for the user's information need based on the source's reliability and extraction costs or other features.

4 Conclusions

The areas of BI and IE have seen steady development in the last decades. They are key players in the challenge of organizing data and enabling more expressive querying and analysis.

There has been some attempts to integrate the two fields, but we believe that better results would come from a true end-to-end integration: the extraction could be adapted to exploit the fact that the target schema is multidimensional; and BI queries should be aware that the queried data comes form an extraction process, adding uncertainty and provenance challenges.

References

[1] XML for Analysis Specification. http://www.xmla.org/xmla1.1.doc as of Jul 2010.

[2] Eugene Agichtein. Scaling information extraction to large document collections. *IEEE Data Eng. Bull*, 28(4):3–10, 2005.

[3] Rema Ananthanarayanan, Vijil Chenthamarakshan, Prasad M. Deshpande, and Raghuram Krishnapuram. Rule based synonyms for entity extraction from noisy text. In *2nd Workshop on Analytics for Noisy Unstructured Text Data*, volume 303 of *ACM International Conference Proceeding Series*, pages 31–38. ACM, 2008.

[4] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David J. Israel, and Mabry Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *IJCAI*, pages 1172–1178, 1993.

[5] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, pages 2670–2676, 2007.

[6] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with Lixto. In *The VLDB Journal*, pages 119–128, 2001.

[7] Robert Baumgartner, Oliver Frölich, and Georg Gottlob. The Lixto systems applications in business intelligence and semantic web. In *ESWC*, volume 4519 of *Lecture Notes in Computer Science*, pages 16–26. Springer, 2007.

[8] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[9] Razvan C. Bunescu and Raymond J. Mooney. Collective information extraction with relational markov networks. In *ACL*, pages 438–445, 2004.

[10] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. Data integration for the relational web. In *PVLDB*, volume 2, pages 1090–1101, 2009.

[11] Michael J. Cafarella, Dan Suciu, and Oren Etzioni. Navigating extracted data with schema discovery. In *WebDB*, 2007.

[12] Chia-Hui Chang, Mohammed Kayed, Moheb R. Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng*, 18(10):1411–1428, 2006.

[13] Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. Mining document collections to facilitate accurate approximate entity matching. In *PVLDB*, volume 2, pages 395–406, 2009.

[14] Eric Chu, Akanksha Baid, Ting Chen, AnHai Doan, and Jeffrey F. Naughton. A relational approach to incrementally extracting and querying structure in unstructured data. In *VLDB*, pages 1045–1056. ACM, 2007.

[15] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.

[16] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[17] Aron Culotta, Andrew McCallum, and Jonathan Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *HLT-NAACL*. The Association for Computational Linguistics, 2006.

[18] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: an architecture for development of robust HLT applications, July 10 2002.

[19] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll. In *13th international conference on World Wide Web*, page 100, March 26 2004.

[20] Ralph Grishman. Information extraction: Techniques and challenges. In *SCIE '97: International Summer School on Information Extraction*, pages 10–27, London, UK, 1997. Springer-Verlag.

[21] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics*, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[22] Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. *PVLDB*, 2(1):289–300, 2009.

[23] Hui Han, C.L. Giles, E. Manavoglu, Hongyuan Zha, Zhenyue Zhang, and E.A. Fox. Automatic document metadata extraction using support vector machines. In *JCDL'03: Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, Automatic metadata creation, pages 37–48, 2003.

[24] Claudia Imhoff, Nicholas Galemmo, and Jonathan G. Geiger. *Mastering Data Warehouse Design: Relational and Dimensional Techniques*. Wiley, 2003.

[25] Panagiotis G. Ipeirotis, Eugene Agichtein, Pranay Jain, and Luis Gravano. To search or to crawl?: towards a query optimizer for text-centric tasks. In *SIGMOD Conference*, pages 265–276. ACM, 2006.

[26] Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli. Evaluating machine learning for information extraction. In *ICML*, volume 119 of *ACM International Conference Proceeding Series*, pages 345–352. ACM, 2005.

[27] Alpa Jain, AnHai Doan, and Luis Gravano. Optimizing SQL queries over text databases. In *ICDE*, pages 636–645. IEEE, 2008.

[28] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2003.

[29] Katharina Kaiser and Silvia Miksch. Information extraction - a survey. Technical Report Asgaard-TR-2005-6, Vienna University of Technology, May 2005.

[30] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of 18th Int. Conf. on Machine Learning*, pages ?–?, 2001.

[31] Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *ICDE*, pages 611–621, 2000.

[32] Imran R. Mansuri and Sunita Sarawagi. Integrating unstructured data into relational databases. In *ICDE*, page 29. IEEE Computer Society, 2006.

[33] Andrew McCallum. Information extraction: distilling structured data from unstructured text. *Queue*, 3(9):48–57, 2005.

[34] Andrew McCallum and Dayne Freitag. Information extraction with HMMs and shrinkage. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, 1999.

[35] M. Michelson and C. A. Knoblock. Creating relational data from unstructured and ungrammatical data sources. *Journal of Artificial Intelligence Research*, 31:543–590, 2008.

[36] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae investigationes*, 30(1):326, 2007.

[37] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *AAAI*. AAAI Press, 2006.

[38] Fuchun Peng and Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.

[39] Juan Manuel Pérez-Martínez, Rafael Berlanga Llavori, María José Aramburu Cabo, and Torben Bach Pedersen. Contextualizing data warehouses with documents. *Decision Support Systems*, 45(1):77–94, 2008.

[40] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *AAAI*, pages 913–918. AAAI Press, 2007.

[41] L. R. Rabiner and B. H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3:4–16, January 1986.

[42] Sujith Ravi and Marius Pasca. Using structured text for large-scale attribute extraction. In *CIKM*, pages 1183–1192. ACM, 2008.

[43] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[44] Horacio Saggion, Adam Funk, Diana Maynard, and Kalina Bontcheva. Ontology-based information extraction for business intelligence. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 843–856. Springer, 2007.

[45] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the coNLL-2003 shared task: Language-independent named entity recognition. pages 142–147, June 12 2003.

[46] Sunita Sarawagi. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[47] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, 2004.

[48] Satoshi Sekine and Chikashi Nobata. Definition, dictionaries and tagger for extended named entity hierarchy, June 18 2004.

[49] Kristie Seymore, Andrew McCallum, and Ronald Rosenfeld. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*, Orlando, Florida, USA, July 1999.

[50] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233, 1999.

[51] Fabian M. Suchanek, Georgiana Ifrim, and Gerhard Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *KDD*, pages 712–717. ACM, 2006.

[52] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, 2002. Morgan Kaufmann.

[53] Frank S. C. Tseng and Annie Y. H. Chou. The concept of document warehousing for multi-dimensional modeling of textual-based business intelligence. *Decision Support Systems*, 42(2):727–744, 2006.

[54] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 593–601, Arlington, Virginia, United States, 2004. AUAI Press.

[55] Michael L. Wick, Aron Culotta, and Andrew McCallum. Learning field compatibilities to extract database records from unstructured text. In Dan Jurafsky and Éric Gaussier, editors, *EMNLP*, pages 603–611. ACL, 2006.

[56] Fei Wu, Raphael Hoffmann, and Daniel S. Weld. Information extraction from wikipedia: moving down the long tail. In *KDD*, pages 731–739. ACM, 2008.

[57] Qi Zhang, Fabian M. Suchanek, Lihua Yue, and Gerhard Weikum. TOB: Timely ontologies for business relations. In *WebDB*, 2008.

[58] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, pages 101–110. ACM, 2009.