Models and indices for integrating unstructured data with a relational database

Sunita Sarawagi

IIT Bombay
sunita@iitb.ac.in

Abstract. Database systems are islands of structure in a sea of unstructured data sources. Several real-world applications now need to create bridges for smooth integration of semi-structured sources with existing structured databases for seamless querying. This integration requires extracting structured column values from the unstructured source and mapping them to known database entities. Existing methods of data integration do not effectively exploit the wealth of information available in multi-relational entities.

We present statistical models for co-reference resolution and information extraction in a database setting. We then go over the performance challenges of training and applying these models efficiently over very large databases. This requires us to break open a black box statistical model and extract predicates over indexable attributes of the database. We show how to extract such predicates for several classification models, including naive Bayes classifiers and support vector machines. We extend these indexing methods for supporting similarity predicates needed during data integration.

1 Introduction

Current research in the area of database mining integration is about finding patterns in data residing within a single structured data box. Most data around us is unstructured but is largely ignored in the data analysis phase. The only effective way to exploit this abundance of unstructured data is to map it the structured schema implicit in a database system. Not surprisingly, a lot of excitement in recent learning and KDD community has been on dealing with partially structured or semi-structured data. Although, in sheer volume structured data is small, it is precious data that captures the language in which data is to be analyzed. Ideally, we would like to be able to map the huge insanity of unstructuredness in terms of this database, and perform our querying and mining the same way.

The KDID community has a lot to offer in this quest. We need to understand and build models to statistically describe and recognize the entities stored in the database. Given the huge volume of unstructured data involved, we have to rely extensively on indexed access to both the database and the unstructured world.

Here are some examples of scenarios where a database and unstructured sources meet.

Consider a company selling electronics products that maintains a table of its products with their features as column names. Companies routinely monitor the web to find competing companies offering products with similar features and to find reviews of newly introduced features. Ideally, they would like to map these unstructured webpages to additional rows and columns in their existing products database.

Another interesting area where there is strong need for integrating unstructured data with a structured database is personal information management systems. These systems organize all information about an individual in a structured fixed-schema database. For example, the PIM would contain structured entries for documents along with their titles, authors and citations organized as a bibtex entry, people including colleagues and students along with their contact information, projects with topics, members and start dates. Links between the structured entities, like members pointing to people and authors pointing to people, establish relationships between the entities. Such an interlinked database opens up the possibility of a query interface significantly richer than has been possible through grep on file-based unstructured desktops.

Given the legacy of existing file-based information systems, the creation of such a database will not happen naturally. Separate data integration processes are required to map unstructured data as it gets created as files into the existing structured database. For example, as a user downloads a paper he would like the bibtex entry of the paper to get automatically extracted and added in his PIM. When a resume appears in an email, he might want to link them to relevant projects.

This is a difficult problem involving several stages of information gathering, extraction and matching. We are very far from this goal. In this article, I will go over the pieces of the puzzle that are relevant and being solved today. We explicitly limit the scope to the following concrete problem. We are given a large multi-relational database and an optional small labeled unstructured set. Our goal is to perform the following on an input unstructured string:

- Extract attributes corresponding to columns names in the database and assign relationships through foreign keys when attributes span multiple linked tables. We call this the information extraction problem.
- Map the extracted entities to existing entries in the database if they match, otherwise, create new entries. We call this the matching problem.

On each of these subproblems a lot of work has already been done. These span a number of approaches starting from manually-tuned set of scripts to plain lookup-based methods to a bewildering set of pattern learning-based methods. However, there is still a need to develop unified solutions that can exploit existing networked structured databases along with labeled unstructured data. We would like a proposed solution to have the following properties:

- Automated, domain-independent, database-driven: Our goal is to design a system that does the integration in as domain-independent and automated a manner as possible. Ideally, the database system should be the only domain-specific component of the whole system. We should exploit it in the most effective way possible.

- Unified learning-based model for all integration tasks: Instead of building one classifier/strategy for recognizing year fields and another one for author-names and a third one for geography, we want a unified model that recognizes all of these through a single global model.
- Probabilistic output for post-querying and mining: We prefer a model that can output probabilities with each extraction/matching it outputs. Integration is not a goal by itself. It is often followed by large aggregate queries and soft-results with probabilities will provide better answers to these queries.
- Exploit all possible clues for extraction/matching in a simple combined framework: Real-life extraction problems will need to exploit a rich and diverse set of clues spanning, position, font, content, context, match in dictionary, part-of-speech, etc. We want an extensible model where it is easy to add such clues in a combined framework.
- Efficient, incremental training and inferencing: Finally we would like the system and the trained models to continuously evolve with the addition of new data and user corrections.

Conditional Random Fields [6,11], a recently proposed form of undirected graphical models, is holding great promise in taking us toward this goal. I will present an overview of CRFs and later concentrate on how they apply for extraction and matching tasks.

2 Conditional Random Fields

We are given $\mathbf{x}$ a complex object like a record or a sequence or a graph for which we need to make n interdependent predictions $\mathbf{y} = y_1 \dots y_n$. During normal classification we predict one variable. Here the goal is to predict n variables that are not all independent. The dependency between them is expressed as a graph G where nodes denote the random variable $\mathbf{y}$ and an edge between two nodes y_i and y_j denotes that these variables are directly dependent on each other. Any other pair of nodes y_i and y_k not connected by a direct edge are independent of each other given the rest of the nodes in the graph. This graph allows the joint probability of $\mathbf{y}$ (given $\mathbf{x}$) to be factorized using simpler terms as:

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{\Phi(\mathbf{y}, \mathbf{x})}{Z(\mathbf{x})} = \frac{\prod_{\mathbf{c}} \Phi_c(\mathbf{y_c}, \mathbf{x}, \mathbf{c})}{Z(\mathbf{x})}$$

This provides a discriminative model of $\mathbf{y}$ in terms of $\mathbf{x}$. The c terms refer to cliques in the graph. For each clique a potential function captures the dependency between variable $\mathbf{y_c}$ in the clique. The denominator $Z(\mathbf{x})$ is a normalizer and is equal to $\sum_{\mathbf{y}'} \Phi(\mathbf{y}', \mathbf{x})$. In exponential models, the potential function takes the form:

$$\Phi_c(\mathbf{y_c}, \mathbf{x}, \mathbf{c}) = \exp(\sum_m w_m f_m(\mathbf{y_c}, \mathbf{x}, c))$$

The terms within the exponent are a weighted sum of features that capture various properties of the variables $\mathbf{y_c}, \mathbf{x}, \mathbf{c}$. Features can take any numerical value and are not required to be independent of one other. This is one of the strengths of the exponential models because it allows a user to exploit several properties of data that might provide clues to its label without worrying about the relationship among them. The w_m terms are the parameters of the model and are learnt during training. We will use $\mathbf{W}$ to denote the vector of all w_ms.

The inference problem for a CRF is defined as follows: given $\mathbf{W}$ and $\mathbf{x}$, find the best labels, $\mathbf{y} : y_1, y_2 \ldots, y_n$

$$\text{argmax}_{\mathbf{y}} \Pr(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}} \sum_c \mathbf{W}.\mathbf{f}(y_c, \mathbf{x}, c)$$

In general it is too expensive to enumerate all possible values of each of the ys and pick the best. However, the limited dependency among variables can be exploited to significantly reduce this complexity. The message passing algorithm is a popular method of solving various kinds of inference problems on such graphs. For a graph, without cycles it can find the best $\mathbf{y}$ and/or various marginals of the distribution in at most two passes over the graph. In a graph with cycles it is used to provide an approximation. An excellent survey of these techniques and how they solve the problems of training and inferencing appear in [5].

We will now see how various forms of information extraction and matching problems can be modeled within this unifying framework of conditional random fields.

3 Information Extraction(IE)

Traditional models for information extraction take as input labeled unstructured data and train models that can then extract the labeled fields from unseen unstructured data. We will review these first. Next, we will see how these can be extended to exploit an existing large database of structured entities.

3.1 IE using only labeled unstructured data

The state of the art methods of IE model extraction as a sequential labeling problem. Typically, IE models treat the input unstructured text as a sequence of tokens $\mathbf{x} = x_1 \ldots x_n$ which need to be assigned a corresponding sequence of labels $\mathbf{y} = y_1 \ldots y_n$ from a fixed set $\mathcal{Y}$. The label at position i depends only on its previous label, thus the corresponding dependency graph on the variables is a *chain*. For instance, $\mathbf{x}$ might be a sequence of words, and $\mathbf{y}$ might be a sequence in $\{I, O\}^{|\mathbf{x}|}$, where $y_i = I$ indicates "word x_i is inside a name" and $y_i = O$ indicates the opposite. The simpler chain structure of the graph allows for more efficient training and inferencing as discussed in [11]. The conditional form of the CRF models allows us to exploit a variety of useful features without worrying about whether these overlap or not. For example, we can add features that capture the following diverse kinds of properties of a word: word ends in "-ski", word is capitalized, word is part of a noun phrase, word is under node X in WordNet, word is in bold font, word is indented, next two words are "and Associates", previous label is "Other".

3.2 IE using labeled data and structured databases

We now consider the case where in addition to the labeled data, we have large databases of entity names. For example, in trying to extract journal names from citations, we can have access to an existing list of journals in a bibtex database.

The conditional model provides one easy way to exploit such databases of entities. Assume we have columns in the database corresponding to different entity types like people and journals that we wish to extract. We simply add one additional binary feature for each such column D, f_D which is true for every token that appears in that column of entity names: *i.e.*, for any token x_i, $f_D(x_i) = 1$ if x_i matches any word of the entity column D and $f_D(x_i) = 0$ otherwise. This feature is then treated like any other binary feature, and the training procedure assigns an appropriate weighting to it relative to the other features.

The above scheme ignores the fact that entity names consist of multiple words. A better method of incorporating multi-word entity names was proposed by Borthwick *et al* [1]. They propose defining a set of four features, $f_{D.unique}$, $f_{D.first}$, $f_{D.last}$, and $f_{D.continue}$. For each token x_i the four binary dictionary features denote, respectively: (1) a match with a one-word dictionary entry, (2) a match with the first word of a multi-word entry, (3) a match with the last word of a multi-word entry, or, (4) a match with any other word of an entry. For example, the token x_i="flintstone" will have feature values $f_{D.unique}(x_i) = 0$, $f_{D.first}(x_i) = 0$, $f_{D.continue}(x_i) = 0$, and $f_{D.last}(x_i) = 1$ (for the column D consisting of just two entries: "frederick flintstone" and "barney rubble".

A major limitation of both of these approaches is that the proposed exact match features cannot handle abbreviations and misspellings in unstructured source. For example, a person names column might contain an entry of the form "Jeffrey Ullman" whereas the unstructured text might have "J. Ullmann". This problem can be solved by exploiting state-of-the-art similarity metrics like edit distance and TF-IDF match [3]. The features now instead of being binary are real-valued and return the similarity measure with the closest word in a dictionary.

A second limitation is that single word classification prevents effective use of multi-word entities in dictionaries. Similarity measures on individual words is less effective than similarity of a text segment to an entire entry in the dictionary. We address this limitation by extending CRFs to do semi-markov modeling instead of the usual markov models. In a semi-markov model we classify segments (consisting of several adjacent words) instead of individual words. The features are now defined over segments and this allows us to use as features similarity measures between a segment and the closest entry in the entity column. During inference, instead of finding a fixed sequence of labels $y_1 \ldots y_n$ we find the best method of segmenting the text and assign labels for each segment. Although, computationally this appears formidable, we can design efficient dynamic programming algorithms as shown in [4] and [9].

Experimental results on five real-life extraction tasks in the presence of large database of entity names show that the semi-markov models along with the use of similarity features increase the overall F1 accuracy from 46% to 58%.

We believe that semi-markov models hold great promise in providing effective use of multi-word databases for IE. More experiments are needed to establish the usefulness of this approach in a general multi-column setting. An interesting direction of future work is how existing foreign key/primary key relationships can be exploited to get even higher accuracies.

4 Entity Matching

We now consider the problem of matching an extracted set of entities to existing entries in the database. In the general case, an input unstructured record will be segmented into multiple types of entities. For example, a citation entry can be segmented into author names, title, journal names, year and volume. The existing database will typically consist of multiple tables with columns corresponding to the extracted entities and linked through foreign and primary keys.

4.1 Pair-wise single-attribute matching

Consider first the specific problem of matching a single extracted entity to a column of entity names, if it exists and returning "none-of-the-above" if it does not. Typically, there are several non-trivial variations of an entity name in the unstructured world. So, it is hard to hand-tune scripts that will take into account the different variations and match an extracted entity to the right database entry. We therefore pursue the learning approach where we design a classifier that takes as input various similarity measures between a pair of records and returns a "0" if the records match and a "1" otherwise. This is a straight-forward binary classification problem where the features are real-valued typically denoting various kinds of similarity functions between attributes like Edit distance, Soundex, N-grams overlap, Jaccard, Jaro-Winkler and Subset match [3]. Thus, we can use any binary classifier like SVM, decision trees, logistic regression. We use a CRF with a single variable for later extensibility. Thus, given a record pair $(x_1 x_2)$, the CRF predicts a y that can take values 0 or 1 as

$$\Pr(y|x_1, x_2) = \frac{\exp(\mathbf{W}.\mathbf{F}(y, x_1, x_2))}{Z(x_1, x_2)} \tag{1}$$

The feature vector $\mathbf{F}(y, x_1, x_2)$ corresponds to various similarity measures between the records when $y = 1$.

An important concern about this approach is efficiency. During training we cannot afford to create pairs of records when the number of records is large.

Typically, we can use some easy filters like only include pairs which have at least one common n-gram to reduce cost. During prediction too we cannot afford to explicitly compute the similarity of an input record with each entry in the database. Later we will discuss how we can index the learnt similarity criteria for considering only a subset of records with which to match.

4.2 Grouped entity resolution

The "match" relation is transitive in the sense that if a record r_1 matches with r_2 and r_2 matches with r_3 than r_1 has to match with r_3. When the input is a group of records instead of a single record as in the previous section, the pair-wise independent classification approach can output predictions that violate the transitivity property. McCallum and Wellner [7] show how the CRF framework enables us to form a correlated prediction problem over all input records pairs, so as to enforce the transitivity constraint.

Assume new the sets of records are not already in the database. Given several records x=$x_1, x_2, \ldots x_n$, we find n^2 predictions, $\mathbf{y} = y_{ij} : 1 \leq i \leq n, 1 \leq j \leq n$ so as to enforce transitivity

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{i,j} \mathbf{W}.\mathbf{F}(y_{ij}, x_i, x_j) + \sum_{i,j,k} w'.f(y_{ij}, y_{ik}, y_{jk}))}{Z(\mathbf{x})}$$

The value of the feature $f(y_{ij}, y_{ik}, y_{jk})$ is set to 0 whenever transitivity constraint is preserved otherwise it is set to $-\infty$. This happens when exactly two of the three arguments are set to 1.

The above formulation reduces to a graph partitioning problem whose exact solution is hard. However, it is possible to get good approximate solutions as discussed in [7]. The authors show that compared to simple pair-wise classification, the combined model increases the accuracy of two noun co-referencing tasks from 91.6% to 94% and 88.9% to 91.6% respectively.

4.3 Grouped multi-attribute entities

In the general case, the entity groups to be matched will each consist of multiple attributes. Grouped matching of multi-attribute records presents another mechanism of increasing accuracy by exploiting correlated predictions using a graphical model like CRF as discussed in [8]. Consider the four citation records below (from [8]).

Record	Title	Author	Venue
b1	Record Linkage using CRFs	Linda Stewart	KDD-2003
b2	Record Linkage using CRFs	Linda Stewart	9th SIGKDD
b3	Learning Boolean Formulas	Bill Johnson	KDD-2003
b4	Learning of Boolean Expressions	William Johnson"	9th SIGKDD

The similarity between b1 and b2 could be easy to establish because of the high similarity of the title and author fields. This in turn forces the venues

"KDD-2003", "9th SIGKDD" to be called duplicates even though intrinsic textual similarity is not too high. These same venue names are shared between b3 and b4 and now it might be easy to call b3 and b4 duplicates in spite of not such high textual similarity between the author and title fields.

Such forms of shared inferencing are easy to exploit in the CRF framework. Associate variables for predictions for each distinct attribute pair and each record pair. In the formulation below, the first set of terms express the dependency between record pair predictions and predictions of attributes that they contain. The second set of terms exploits the text of the attribute pairs to predict if they are the same entity or not.

$$\Pr(\mathbf{y}|\mathbf{x}) = \frac{\exp(\sum_{i,j}\sum_k \mathbf{W}.\mathbf{F}(y_{ij}, A^k_{ij}) + \mathbf{W}'.\mathbf{F}'(A^k_{ij}, x_i.a^k, x_j.a^k))}{Z(\mathbf{x})}$$

The main concern about such formulations is the computation overhead and [8] presents some mechanisms for addressing them using graph partitioning algorithms. The combined model is shown to increase the match accuracy of a collection of citations from 84% to 87% ([8]).

5 Indices for efficient inferencing

For both the extraction and matching tasks, efficient processing will require that we break open the classification function learnt by a CRF and define appropriate indices so that we can efficiently select only that data subset that will satisfy a certain prediction. All aspects of this problem are not yet solved.

We will next consider a specific matching scenario of Section 4.1 where it is possible to design indices to reduce the number of entries in the database with which a query record is compared.

After the model in Equation 1 is trained we have a weight vector $\mathbf{W}$ for each feature in the vector $\mathbf{F}(y, x_1, x_2)$. When applying this model during inferencing, we are given a string x_q and our goal is to find the x_j-s from the database with the largest value of $\mathbf{W} \cdot \mathbf{F}(1, x_q, x_j)$. We claim that for most common similarity features, this function can be factorized as

$$\mathbf{W} \cdot \mathbf{F}(1, x_q, x_j) = w_1(x_q)f_1(x_j), \ldots w_r(x_q)f_r(x_j).$$

Consider an example: The original function is:

$$\mathbf{W} \cdot \mathbf{F}(1, x_q, x_j) = 0.3 \text{ t}f - idf(x_j, x_q) + 0.4 \text{ common-words}(x_j, x_q)$$

. This can be rewritten as:

$$\sum_{word\ e \in x_q} (0.3 \text{ w}eight(e, x_q)\text{w}eight(e, x_j) + 0.4[\![e \in x_j]\!])$$

The factorized form above allows us to index the data for efficiently finding the best match for a given query record as follows. We create inverted index for each of the r features f_i. Thus, for each feature we keep the list of (record identifiers, feature-value) pair for all records that have a non-zero value of the feature. The query records assigns a weight for a subset of these features. We create a weighted merge of these lists to find the record identifiers that will have the largest value of the dot-product. A number of techniques have been proposed in the database or IR literature to efficiently perform this merge and find the top-k matching records without performing the full merge. These details can be found in [10,2,12].

A number of interesting problems in designing indices for pulling parts that are likely to contain entities of a given type still remain. We can expect to see lot of work in this area in the future.

6 Conclusion

In this article we motivated the research area of developing techniques for information extraction and integration by exploiting existing large databases. Recent advances in graphical models provide a unified framework for structure extraction and reference resolution. This is a call to researchers in the KDD community to investigate the problems of developing practical models for these problems and providing methods for efficient training and inferencing.

References

1. A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Sixth Workshop on Very Large Corpora New Brunswick, New Jersey. Association for Computational Linguistics.*, 1998.
2. Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, and Rajeev Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, 2003.
3. William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003. To appear.
4. William W. Cohen and Sunita Sarawagi. Exploiting dictionaries in named entity extraction: Combining semi-markov extraction processes and data integration methods. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004. To appear.
5. M. I. Jordan. Graphical models. *Statistical Science (Special Issue on Bayesian Statistics)*, 19:140–155, 2004.
6. John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML-2001)*, Williams, MA, 2001.
7. Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, Acapulco, Mexico, August 2003.

8. Parag and P. Domingos. Multi-relational record linkage. In *Proceedings of 3rd Workshop on Multi-Relational Data Mining at ACM SIGKDD*, Seattle, WA, August 2004.
9. Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPs (to appear)*, 2004.
10. Sunita Sarawagi and Alok Kirpal. Efficient set joins on similarity predicates. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004.
11. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *In Proceedings of HLT-NAACL*, 2003.
12. Martin Theobald, Gerhard Weikum, and Ralf Schenkel. Top-k query evaluation with probabilistic guarantees. In *VLDB*, pages 648–659, 2004.