Linking, mapping, and clustering entity records in information-based solutions for business and professional customers

Jack G. Conrad, Tonya Custis, Christopher Dozier, Terry Heinze, Marc Light, Sriharsha Veeramachaneni
Thomson Corporation
610 Opperman Drive, Saint Paul, MN 55123 USA
E-mail: marc.light@thomson.com

Abstract

This is a position paper that describes a number of use cases and their corresponding evaluation metrics. We discuss three types of resolution problems: linking entity mentions in text to records in a database, mapping records in one database to those in another database, and clustering records in a single database. The use cases arose at the Thomson Corporation and the systems developed support a number of products.

1. Introduction

The aim of this paper is to provide the reader with an overview of the entity resolution tasks we have worked on, the methods we have employed, and the evaluations we have used.

To provide context for our discussion, it is useful to have some idea of what our company does: the Thomson Corporation provides information-based solutions for lawyers, business people, nurses, doctors, scientists, and other professionals. Many of these solutions involve textual sources in combination with more structured sources such as databases of numeric and nominal information. Both the text and the databases contain information about entities ranging in type from genes to cities. Part of the "intelligent information" that Thomson products use is the mapping, and clustering of entity records along with linking of these records to text mentions.

Historically this mapping, clustering, and linking has been done manually. However, increasingly, automated systems are being used. In some cases, automated systems assist humans, improving their accuracy and efficiency. In other cases, the accuracy of the automated systems is sufficient alone. Our department, Thomson Research and Development, has been involved in such work and has developed a number of automated systems including systems that support products such as Westlaw Profiler (http://west.thomson.com/westlaw/profiler/), Westlaw Medical Litigator (http://west.thomson.com/westlaw/litigator/medical.aspx), and West's Monitor Suite (http://www.firm360.com/).
In addition to working in the legal domain, in recent years, we have worked on systems for Thomson Financial, Thomson Scientific, and Thomson Healthcare.

The remainder of the paper is structured as follows. First, we discuss tasks of linking entity mentions in text to records in a database. Next we discuss mapping records in one database to those in another database; such a task arises when two databases need to be merged. Finally we discuss clustering records in a single database; such a task arises when a database contains numerous records for the same entity but there is no explicit information denoting the relation. For each of these three general tasks, we describe our general approach and evaluation methods and then describe one or more case studies.

2. Linking entity mentions in text to records in a structured database

We have created a number of applications that are based on extracting named entities from text and attaching them to structured records in an entity database. The basic method consists of the following two steps. First we extract from the text the entity names of interest along with information that can be used as evidence for entity resolution. Then we place the extracted text segments into a structured record called a template record and attempt to resolve (link or match) the template record to a record in an entity database. The first step in this process is called the extraction phase. The second step is called the entity resolution phase. We will only discuss the resolution phase here.

The entity resolution phase is based on record linkage techniques. The entity resolution phase can be separated into two phases: blocking and matching. In the blocking phase, we use some element of the extracted person name to read a subset of the records from the database likely to contain any existing database record matching the extracted person name. A typical blocking key might consist of all or part of a person's last name. Blocking is necessary because it is usually not computationally feasible to perform the full matching function on every database record for each extracted name. Blocking and its role in record linkage is further discussed in (Winkler, 1995) and (Baxter, et al., 2003). The second phase is matching and consists of comparing each database record in the block to the current template record and computing the likelihood that the template record and a given database record refer to the same person (i.e. match). The complexity of the resolution step is determined by the size and similarity of the entities in the database, the quality of the extracted data in the template record, the comprehensiveness of the database, and any contextual knowledge about the text that indicates whether the person names from the text are likely to belong the same set of people covered by the database.

For person names, the features we often use in our matching functions include the degree of match between the first, middle, and last name of the person and also include location information, appositive information indicating person's profession, and organization names with which the person is affiliated. We usually combine features to compute a match belief score using either naïve Bayes and support vector machine classifiers   In some cases, we have used heuristic rules to combine the features to arrive at a decision. At this point, we do not have a principled process for deciding which type of classifier to use on a new problem.

We typically collect positive training data by asking editors to provide between 500 and 1000 manually matched examples chosen at random. We then collect very large amounts of negative training data automatically by pairing the template record from the positive data with all of the database records except the one identified as matching in the positive set.

After we learn our match function from the training data and compute match scores between every database record in the block and a given template record, the highest scoring database record is linked to the template record provided the match score exceeds a match threshold determined by the training data. If the highest scoring record falls below the match threshold, we check the score against a low threshold to determine if the template record is far enough away from all database records to warrant the creation of a new database record. If the match score falls below the low threshold, it is likely the template record refers to a new person and we therefore add it to the database. If the highest score falls between the match and low thresholds, we log the template record for manual review.

We usually measure the quality of our text to database linking systems using precision and recall as measured against a held out test set. We like to have a least 300 test records available, which often gives us a small enough confidence interval around the resulting precision and recall numbers. Our baselines start with a system that chooses at random from the returned block size. Thus, if the average block size is 2, then the first baseline would have an accuracy of 50% (precision 50%, recall 50%, and F-measure of 50%). Then, we provide progressively more intelligent baselines by using heuristics based on frequent high precision features, e.g., pick the record that has a location field closest in edit distance to the template field.

In the subsections that follow, we describe two specific applications that are based on the text-to-database record linkage methodology described above.

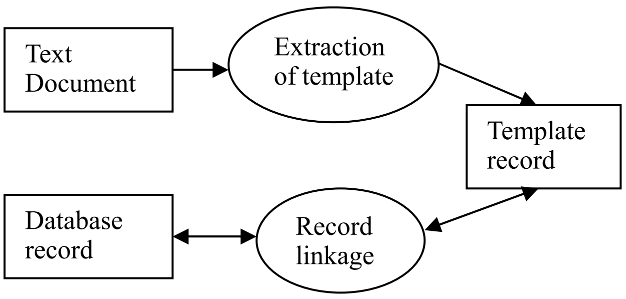

Figure 1: System diagram for linking entities in text to database records

2.1 Case study: linking legal professionals from caselaw documents to legal directories

In this task we extracted attorney, judge, and expert witness names from American caselaw, briefs, and professional journals. Then we attached these names to unique person records in a comprehensive database of U.S. legal professionals (Dozier & Haschart, 2000). By establishing these links, we are able to offer users the ability to browse through documents in which an individual is mentioned and to offer users the ability to jump to an individual's curriculum vitae from a name mentioned in text. New records are continually added to the person database when mined names do not match any individuals currently residing in the database.

A typical paragraph in caselaw that identifies the attorneys involved in a case is shown below.

> H. Patrick Weir, Jr., Lee Hagen Law Office, ltd., Fargo, N.D., Jeffrey J. Lowe, Gray & Ritter, P.C., St. Louis, MO, and Joseph P. Danis and John J. Carey, Carey & Danis, LLC, St. Louis, MO, for plaintiff and appellant.

Figure 2: Attorney paragraph

In the example paragraph, our system extracts and links H. Patrick Weir, Jr., Jeffrey J. Lowe, Joseph P. Danis, and John J. Carey to attorney records in our legal directory.

We use regular expression patterns to extract names and name matching evidence which includes law firm, city, and state information. Our name matching evidence consists of features that compare each of the following fields: first name, middle name, last name, firm name, and city/state. The values of the features are: matches exactly, matches in a fuzzy way, is unknown, or mismatches. An example of fuzzy matching would be if one name is a nickname of the other or if one name is an initial only and matches the first letter of the other name.

We use several thousand positive training examples to train a naïve Bayes match classifier. The size of our database was approximately 1 million records. We blocked on last name first, and, if we failed to find a match with this block, we blocked on first name. This multiple blocking method allowed us to capture cases where an attorney has changed her last name through marriage for example.

We compared our method to three other matching techniques for an attorney name. We measured the precision and recall we would get (1) if we link attorney names only when the first, middle, last name, and city-state match exactly, (2) if we link attorney names only when the first, middle, and last name match exactly without regard to city-state or firm information, and (3) if we link attorney names only when the first and last name match exactly without regard to middle name, city-state, or firm. The results are shown below and are compared with the naïve Bayes matching. As can be seen, the naïve Bayes technique significantly outperforms the baseline methods. For this comparison, we used a single match threshold of 0.25. High template and database record pairs scoring above the threshold were considered matched and those falling below were considered to signify an unmatchable template record.

	Prec.	**Recall**	**F**
Naïve bayes with threshold 0.25	0.993	0.916	0.953
Exact Match on first name, middle name, last name, and city-state	0.994	0.422	0.592
Exact Match on first, middle and last name	0.950	0.613	0.745
Exact Match on first and last name only	0.939	0.590	0.725

Table 1: Attorney matching methods comparisons

2.2 Case study: linking persons, companies, and locations from financial newswires to corresponding directory listings

We have also tagged mentions of companies, locations, and persons in financial news text and resolved them to corresponding authority files. Our biggest challenge in this application has been the resolution of persons. Our authority file consists of 677,765 person records: the officers and directors of publicly traded companies.

Our template record consists of the first, middle initial, last name, and companies named in the article. We block using the first and last name of the record. The blocks contain 4 or less records 96% of the time; however, some contain over 80 records. The matching phase is performed using a set of heuristics. Rules for positive resolution are applied in order of greatest-to-least evidence and confidence. Measures of evidence and confidence include the degree to which a name mention in the text is an exact match with the authority file and whether or not the company name associated with a particular name record is also mentioned in the document text. Names that are common with respect either to having many records associated with them, or in terms of a measure of overall name commonness (as determined by counts in a credit header database) are considered to be low-confidence and require more evidence for positive resolution.

Our system achieves an F-measure of 92.2% on person resolution (91.7% precision, 92.7% recall). This can be compared against a baseline of 50% accuracy. This baseline is produced by randomly choosing a match from the block which average 2 records in size.

3. Mapping records in one database to those in another database

We consider one of the databases to be the target and then, as in the previous section, the task of matching records in a database with those in the target database consists of the two phases mentioned in the previous section: blocking and matching.

Blocking can be explained in terms of extracting sets of candidate records from the target database that satisfy certain query parameters — the goal of which is to select only those blocks of data that meet certain requirements for further processing (e.g., last name matches query AND zip code matches query). When a given blocking function does not yield any candidate match, a broader blocking function is tried. Matching is done by scoring a feature vector of similarities over the various fields. The feature values can be either binary (verifying the equality of a particular field in the update and a master record) or continuous (some kind of normalized string edit distance between fields like *street address*, *first name,* etc).

As in the previous section, the evaluation of such a matching task typically includes precision and recall in an IR sense, as well as the associated F-measure. We may also wish to measure our progress in terms of precision among the non-matches (how often is our “don’t match” decision correct)? Speed in terms of resolutions-per-second is another metric that real-time production applications often monitor.

3.1 Case study: the physician database

The task consists of merging a physician record from an *“update” database* to the record of the same physician in a *master record database*. The update database has fields that are absent in the master record database and *vice versa*. The fields in common include the *name* (first, last and middle initial), several *address* fields, phone, specialty, and the *year-of-graduation*.

More specifically, the system merges each of 20,000 physician records to the record of the same physician in the *master record database* consisting of approximately 1 million records. The fields in common include the *name* (first, last and middle initial), several *address* fields, phone, specialty, and the *year-of-graduation*.

Although the *last name* and *year of graduation* are consistent when present, the *address*, *specialty* and *phone* fields have several inconsistencies owing to different ways of writing the address, new addresses, different terms for the same specialty, missing fields, etc. However, the *name* and *year* alone are insufficient for disambiguation. We had access to ~500 manually matched update records for training and evaluation (about 40 of these update records were labeled as unmatchable with the information available).

We performed blocking by querying the master record database with the *last name* from the update record. Matching was done by scoring a feature vector of similarities over the various fields. The feature values were either binary (verifying the equality of a particular field in the update and a master record) or continuous (some kind of normalized string edit distance between fields like *street address*, *first name* etc.).

The logistic-regression-based matching algorithm assigns to each feature vector the probability that it corresponds to a match. All the records in the block are ranked according to this probability and the highest scoring record is assigned as the match if its score exceeded some appropriate threshold.

The training of the logistic regression algorithm was done by a semi-supervised algorithm called *surrogate learning*, which is based on the property that the binary *year of graduation* feature is independent of the other features if the two records are not matches. The reader is referred to (Veeramachaneni & Kondadadi, 2008) for a description of the algorithm and experimental results.

The matching algorithm was evaluated on 500 manually matched records with n-fold cross-validation. From this assessment, the precision and recall of the algorithm were determined to be 96% and 95% respectively.

4. Clustering records in a single database

In some cases, a single database table contains many records for the same entity but there is no explicit link expressing the identity relationship. The task then is to partition the table into equivalence classes where each class contains all the records for a specific entity. Again the task breaks down into the subtasks of blocking and matching; however, a third task of clustering is also required. We have successfully employed the similar blocking and matching techniques to those described in the previous sections. For clustering, we have used agglomerative clustering but other methods could also be employed (Jain & Dubes, 1988).

Evaluation, by contrast, does not follow the approach of the previous tasks. Instead of statistics based on counts of record pair linkages correctly found, incorrectly proposed, missed, etc., the statistics are based on counts with in clusters and then averaged over clusters.

4.1 Case study: account rolling

Within one of our internal accounting systems, multiple database records may exist for a single customer. Each record corresponds to a separate license for a single product. The customer database totals approximately 1.5 million records. The record format allows for flexibility in identifying the customer: up to four text fields may be used to name the customer entity, contact entity, and secondary entities such as departments, offices, regions, etc. The database is populated by multiple systems and consistent text field usage is not enforced. To help facilitate the assignment of sales representatives, the application needs to resolve account clusters by customer, using textual information only (the four name fields and address fields). Customer types include corporations, state and federal governmental agencies, and educational institutions. Corporate names tended to vary over time, reflecting mergers. Governmental customer names could also be non-unique: the same name may be utilized by similar entities in different cities, counties, states, and federal jurisdictions.

The database did indicate the market segment, if known, of the record. Therefore, clustering could be performed within each segment separately. Two thirds of the records had a non-null market segment. Unknown records were to be matched against the resulting segment clusters and added if matched.

The large corporations were expected to produce a relatively small number of large population clusters. A typical large corporation might have several hundred accounts. Approximately 50,000 accounts were expected to produce about 250 clusters. Far more problematic were the state governmental accounts. These represent the largest number of records, over 350,000. Clusters were expected to be numerous and very sparsely populated.

An SVM was used to compare record pairs. The feature data in each segment varied in completeness, location, and structure. In each of the segments, we wanted to match and cluster on the name of the entity. Feature selection involved selecting the optimum combination of the four text fields for each segment to determine the best cross match between records to keep expensive string comparisons to a minimum. The Jaro-Winkler algorithm was predominantly used in order to weight the first part of the string.

The SVM was trained on user provided gold data pairs. We selected a ratio of positive to negative training pairs of 1/2 (2000 and 4000 pairs respectively were used); 80% of the sampled pairs were used for training and the remaining 20% used for model validation. We performed validation experiments to select the optimal combination of SVM parameters (C, gamma, and kernel). An RBF kernel was used.

A basic agglomerative clustering technique was employed. The first record was set aside as the first cluster. The second record was compared to the first. If it matched (the SVM score exceeded a configurable threshold), it was added to the cluster. Otherwise a new cluster was created. Each subsequent record in the input data set was compared to existing clusters. When comparing a record to a cluster, the record was compared to each record in the cluster until either a match was found that exceeded the threshold or a negative match was found. If there were more than one matched cluster, the matched clusters were merged together.

After all of the records had been processed, the single valued clusters (i.e. clusters with only one element) were extracted and re-run through the process using the multi-valued clusters as the starting point. This was repeated until the number of single valued clusters reached equilibrium.

A final cluster merging was performed on the multi-valued clusters. The most frequently occurring entity name in each cluster was determined. For any two clusters, if the they had the same majority entity name and a similarity score (the product of the ratios of the number of occurrences of the majority entity name to the number of records in the cluster - a modified cosine similarity) between the two exceeded a configurable threshold (usually .80), the two clusters were merged.

Standard precision and recall metrics lacked a precise definition when applied to clustering. We initially devised two related metrics, purity and fragmentation to compare our cluster results with the gold data. Purity, a measure of how many records in the cluster belong together, measures the precision of the clusters at both the macro and micro level. Fragmentation attempted to quantify how many clusters it took to represent the true cluster. Purity is defined with respect to the generated clusters and fragmentation is defined with respect to the gold standard clusters. A purity of 1 and a fragmentation of 0 would indicate a perfect cluster.

The fragmentation scores were not informative enough. Similar fragmentation scores did not indicate how and to what extent the records were distributed across the set of clusters. A detailed tabular approach provided much better measurements:

Let **G** be a gold data cluster:
the set off all accounts, $\mathbf{a_i}$, that belong to a single customer.

Let **C** be the set of all generated clusters that completely enclose **G**:
for all $\mathbf{a_i}$ in **G**, $\mathbf{a_i}$ is a member of a cluster in **C**

Fragmentation of **G** equals the number of clusters in **C - 1**

Let $\mathbf{C_j}$ be a generated cluster:

Purity of $\mathbf{C_j}$ equals (size of largest gold standard contributor to the cluster) / (size of $\mathbf{C_j}$)

For any given sample, we determined the gold data clusters (record ids and count). For each gold data cluster, we found all generated clusters that contained an occurrence of a record id. For each of these clusters, we calculated the coverage ratio of id occupancies to the size of the cluster. For the three largest clusters, we reported the coverage ratios (this is a measure of how well any one of these clusters covers the target gold data cluster). We then accumulated average coverage scores for all clusters and macro coverage scores over the entire sample. We also reported the number of times a single cluster is generated that exactly covers the corresponding gold data cluster.

For each of the three largest clusters reported on for each gold data cluster, we calculated the purity of the cluster by taking the ratio of correct matches to the size of the cluster, then accumulated both micro and macro averages.

Let us now apply these metrics to our system's output. When compared against the customer's existing method of clustering (a rule based system), we produced higher coverage scores for the largest generated cluster. We placed more records in a single large cluster while the existing method tended to distribute records over two or more large clusters. Both approaches had residual single records. Purity scores were consistently high (0.99 for large sized clusters) so the comparison and clustering techniques were valid. Nonetheless, fragmentation could not be reduced due to insufficient evidence in the remaining single valued clusters.

Other comments on the output:

- There were a large number of single records that could not be clustered. In most cases, a valid entity name was missing (not present in any of the four possible record fields) or only a contact name (a person) was entered. The appearance of just a person name caused over-rolling (records placed in the wrong cluster) because of similarity of the person names (filtering techniques removed most of these problems).

- The entity names in governmental segments were not unique. The same name could indicate both a match and a mismatch. For example "Court Magistrate" was a match within the same circuit court, but a mismatch otherwise (this also resulted in positive and negative training vectors that were identical).

- There were a large number of single records that could not be merged into their respective clusters. This results in large fragmentation (e.g. we could generate one large cluster that covered 90% of the records in a gold data cluster, but the remaining records resulted in single clusters that could not be merged).

- Collections that are comprised of a relatively small number of large clusters are best suited to our techniques. Collections that consist of a very large number of very small or singular clusters did not perform as well. It looks like our techniques did quite well when the clusters were large enough to establish strong similarity measurements between records. For sparsely populated clusters, there wasn't enough evidence.

5. Summary

We have described a number of entity record tasks. The first two tasks were (i) linking mentions of people and companies in legal text to structured authority files and (ii) linking mentions of entities in financial newswires to structured authority files. The next task involved mapping records in one database to those in another: record matching for a physician database. Finally, we described the task of clustering record accounts so that clusters contained all accounts for a single company.

Although each of the entity record linking, mapping, and clustering problems described above are distinct, and invite their own innovative solutions, there also exists among them some common dimensions and broader lessons to be learned. Some of these common dimensions include the following. In an IR-like manner, there exists a clear trade-off between precision and recall. One generally cannot make dramatic gains in one without witnessing degradation in the other. It may only be the ratio of the benefit-to-cost that may change (e.g., a two point gain in recall costing five points in precision). Just as significantly, precision and recall only tell part of the story, and tend to understate other challenges associated with the problem space, for instance, deciding that a candidate pair does *not* represent a solid match (i.e., avoiding false positives, a.k.a., *non-match* precision) can be just as challenging as deciding that a match is validated. Other auxiliary metrics like average block size, in the case of linking or mapping, or maximum obtainable coverage or purity, in the case of clustering, can be equally informative indicators of problem difficulty or solution quality, and cannot be ignored when striving for globally optimal solutions. Still other issues carry additional lessons relating to the scale of the problem, the diversity of the available data sources, and the dynamic nature of the underlying entity data. Each of these dimensions compound the entity resolution challenge, and require real-world solutions in order to satisfy the underlying practical constraints. Because our solutions are focused on industrial applications, results that surpass existing baselines but ignore these critical dimensions (scale, varying record quality, dynamic environments) are not acceptable. Ultimately these approaches need to deliver high performance solutions in terms of result quality, scalability, and robustness, not to mention speed.

6. References

Baxter, R., Christen, P., and Churches, T. (2003). A Comparison of Fast Blocking Methods for Record Linkage. In *Proceedings of ACM SIGKDD 2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation.* Washington, D.C., USA.

Dozier, C. and Haschart, R. (2000). Automatic Extraction and Linking of Person Names in Legal Text. In *Proceedings of RIAO 2000 (Recherche d'Information Assistee par Ordinateur).* Paris, France: pp. 1305--1321.

Jain, A.K. and Dubes, R.C. (1988). Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs, NJ.

Veeramachaneni, S. and Kondadadi, R.K. (2008). *Surrogate Learning—From Feature Independence to Semi-supervised Classification.* Submitted to ICML 2008.

Winkler, W. (1995). Matching and Record Linkage. In Cox, B. G., ed., *Business Survey Methods*, Wiley.