recibido 18-05-2007; aceptado 22-06-2007

Alias Assignment in Information Extraction

Emili Sapena, Lluís Padró and Jordi Turmo
TALP Research Center
Universitat Politècnica de Catalunya
Barcelona, Spain
{esapena, padro, turmo}@lsi.upc.edu

Resumen: Este artículo presenta un método general para la tarea de asignación de alias en extracción de información. Se comparan dos aproximaciones para encarar el problema y aprender un clasificador. La primera cuantifica una similaridad global entre el alias y todas las posibles entidades asignando pesos a las características sobre cada pareja alias-entidad. La segunda es el clásico clasificador donde cada instancia es una pareja alias-entidad y sus atributos son las características de ésta. Ambas aproximaciones usan las mismas funciones de características sobre la pareja alias-entidad donde cada nivel de abstracción, desde los carácteres hasta el nivel semántico, se tratan de forma homogénea. Además, se proponen unas funciones extendidas de características que desglosan la información y permiten al algoritmo de aprendizaje automático determinar la contribución final de cada valor. El uso de funciones extendidas mejora los resultados de las funciones simples.

Palabras clave: asignación de alias, extracción de información, entity matching

Abstract: This paper presents a general method for alias assignment task in information extraction. We compared two approaches to face the problem and learn a classifier. The first one quantifies a global similarity between the alias and all the possible entities weighting some features about each pair alias-entity. The second is a classical classifier where each instance is a pair alias-entity and its attributes are their features. Both approaches use the same feature functions about the pair alias-entity where every level of abstraction, from raw characters up to semantic level, is treated in an homogeneous way. In addition, we propose an extended feature functions that break down the information and let the machine learning algorithm to determine the final contribution of each value. The use of extended features improve the results of the simple ones.

Keywords: Alias Assignment, Information Extraction, Entity Matching

1 *Introduction*

Alias assignment is a variation of the entity matching problem. Entity matching decides if two given named entities in the data, such as "George W. Bush" and "Bush", refer to the same real-world entity. Variations in named entity expressions are due to multiple reasons: use of abbreviations, different naming conventions (for example "Name Surname" and "Surname, N."), aliases, misspellings or naming variations over time (for example "Leningrad" and "Saint Petersburg"). In order to keep coherence in extracted or processed data for further analysis, to determine when different mentions refer to the same real entity is mandatory.

This problem arises in many applications that integrate data from multiple sources. Consequently, it has been explored by a big number of communities including statistics, information systems and artificial intelligence. Concretely, many tasks related to natural language processing have been involved in the problem such as question answering, summarization, information extraction, among others. Depending on the area, variants of the problem are known with some different names such as identity uncertainty (Pasula et al., 2002), tuple matching, record linkage (Winkler, 1999), deduplication (Sarawagi and Bhamidipaty, 2002), merge/purge problem (Hernandez and Stolfo, 1995), data cleaning (Kalashnikov and Mehrotra, 2006), reference reconciliation (Dong, Halevy, and Madhavan, 2005), mention matching, instance identification and so others.

Alias assignment decides if a mention in one source can be referring to one or more entities in the data. The same alias can be shared by some entities or, by the opposite, it can be referring to an unknown entity. For instance, alias "Moore" would be assigned to the entity "Michael Moore" and also to "John Moore" if we have both in the data. However, alias "P. Moore" can not be assigned to any of them. Therefore, while entity matching problem consists of determining when two records are the same real entity, alias assignment focuses on finding out whether references in a text are referring to known real entities in our database or not. After alias assignment, a disambiguation procedure is required to decide which real entity among the possible ones is the alias pointing to in each context. The disambiguation procedure, however, is out of the scope of this paper.

There is little previous work that directly addresses the problem of alias assignment as a main focus, but many solutions have been developed for the related problem of entity matching. Early solutions employ manually specified rules (Hernandez and Stolfo, 1995), while subsequent works focus on learning the rules from training data (Tejada, Knoblock, and Minton, 2002; Bilenko and Mooney, 2003). Numerous solutions focus on efficient techniques to match strings, either manually specified (Cohen, Ravikumar, and Fienberg, 2003), or learned from training data (Bilenko and Mooney, 2003). Some others solutions are based in other techniques taking advantage of the database topology like clustering a large number of tuples (McCallum, Nigam, and Ungar, 2000), exploiting links (Bhattacharya and Getoor, 2004) or using a relational probability model to define a generative model (Pasula et al., 2002).

In the last years, some works take advantage of some domain knowledge at the semantic level to improve the results. For example, Doan *et al.* (Doan et al., 2003) shows how semantic rules either automatically learned or specified by a domain expert can improve the results. Shen *et al.* (Shen, Li, and Doan, 2005) use probabilistic domain constraints in a more general model employing a relaxation labeling algorithm to perform matching.

Some of the methods used for entity matching are not applicable to alias assignment because the information contribution of the pair alias-entity is poorer than that of an entity-entity pair. An alias is only a small group of words without attributes and, normally, without any useful contextual information. However, using some domain knowledge, some information about the entities and some information about the world, it is possible to improve the results of a system that uses only string similarity measures.

This paper presents a general method for alias assignment task in information extraction. We compared two approaches to face the problem and learn a classifier. The first one quantifies a global similarity between the alias and all the possible entities weighting some features about each pair alias-entity. The algorithm employed to find the best weights is Hill Climbing. The second is a classical pairwise classification where each instance is a pair alias-entity and its attributes are their features. The classifier is learned with Support Vector Machines. Both approaches use the same feature functions about the pair alias-entity where every level of abstraction, from raw characters up to semantic level, is treated in an homogeneous way. In addition, we propose a set of extended feature functions that break down the information and let the machine learning algorithm to determine the final contribution of each value. The use of extended features improves the results of the simple ones.

The rest of the paper is structured as follows. In section 2, it is formalized the problem of alias assignment and its representation. Section 3 introduces the machine learning algorithms used. Next, section 4 presents the experimental methodology and data used in our evaluation. In section 5 we describe the feature functions employed in our empirical evaluation. Section 6 shows the results obtained and, finally, we expose our conclusions in section 7.

2 Problem definition and representation

The alias assignment problem can be formalized as pairwise classification: Find a function $f : N \times N \rightarrow \{1, -1\}$ which classifies the pair alias-entity as positive (1) if the alias is representing the entity or negative (-1) if not. The alias and the entity are represented as strings in a name space N. We propose a variation of the classifier where we can use also some useful attributes we have about the entity. In our case, function to find will be: $f : N \times M \rightarrow \{1, -1\}$ where M represents a different space including all entity's attributes.

We define a *feature function* as a function that represents a property of the alias, the entity, or the pair alias-entity. Once a pair alias-entity is represented as a vector of features, one can combine them appropriately using machine learning algorithms to obtain a classifier. In section 3 we explain how we learn classifiers using two different approaches. Most of the feature functions used here are *similarity functions* which quantify the similarity of the pair alias-entity according to some criteria. In a *similarity function* the returned value r indicates greater similarity in larger values while shorter values indicates lower similarity (dissimilarity).

Feature functions can be divided in four groups by its level of abstraction from raw characters up to semantic level. In the lower level, the functions focus on character-based similarity between strings. These techniques rely on character edit operations, such as deletions, insertions, substitutions and subsequence comparison. Edit similarities find typographical errors like writing mistakes or OCR errors, abbreviations, similar lemmas and some other difference intra-words.

The second level of abstraction is centered in vector-space based techniques and it is also known as token-level or word-level. The two strings to compare are considered as a group of words (or tokens) disregarding the order in which the tokens occur in the strings. Token-based similarity metrics uses operations over sets such as union or intersection.

In a higher level we find some structural features similar to the work in (Li, Morie, and Roth, 2004). Structural features encode information on the relative order of tokens between two strings, by recording the location of the participating tokens in the partition.

The highest level includes the functions with added knowledge. This extra knowledge can be obtained from other attributes of the entity, from an ontology or can be knowledge about the world. Some previous works (Shen, Li, and Doan, 2005; Doan et al., 2003) use this extra knowledge as rules to be satisfied. First, rules are specified manually or obtained from the data, and then they need to assign some weight or probability to each rule and also distinguish hard rules from soft ones. In (Shen, Li, and Doan, 2005) weights are established by an expert user or learned from the same data set to classify. In our work, we present another way to use this information. We propose to add more feature functions to increase the number of attributes for our classifier. Each new feature function describes some characteristic of the alias, of the entity, or of the pair alias-entity that needs some extra knowledge. The contribution of each feature will be learned as any other similarity function when some machine learning method is applied.

3 Learning classifiers

Two approaches are used and compared in order to obtain a good classifier using feature functions introduced above, Hill Climbing (Skalak, 1994) and Support Vector Machines (Cortes and Vapnik, 1995). Each one has different points of view of the problem. The first one, treats the problem as a nearest neighbor model and tries to determine a global Heterogeneous Euclidean-Overlap Metric (HEOM) from the target alias to all the entities in the database. The alias will be assigned to the entities with a HEOM shorter than some *cut-value*. Each pair alias-entity has a HEOM composed by all the values of similarity. The second point of view is a classical classifier based on the instance's attributes projected in a multidimensional space. The classifier consist in an hyperplane that separates samples in two classes. Each pair alias-entity with the values of the feature functions as attributes is an instance for the classifier that can be classified as positive (matching) or negative (not matching).

The first point of view determines a HEOM composed by the values returned by the similarity functions. All the similarity functions are normalized and transformed to *dissimilarities* in order to obtain a small value of HEOM when alias and entity are similar and large value otherwise. HEOM is obtained with all the *dissimilarities* weighted in a quadratic summatory:

$$HEOM = \sqrt{\sum_i w_i (d_i)^2}$$

where d_i is the *dissimilarity* corresponding to the similarity function i and w_i is the weight assigned to this value. Using a training data set, Hill Climbing determines the best weight for each feature and the *cut-value* in order to achieve the best possible performance. The algorithm in each step increases and decreases each weight in a small step-value and selects the modification with best results. The process is repeated until no modification is found to improve the result of the current solution. The method is executed several times starting with random weights. Some of the advantages of Hill Climbing is that it is easy to develop and can achieve good results in a short time.

The second approach consist in a pair alias-entity classifier using Support Vector Machines (SVM) (Cortes and Vapnik, 1995). SVM have been used widely as a classifier (Osuna, Freund, and Girosi, 1997; Furey et al., 2000). This technique has the appealing feature of having very few tunable parameters and using structural risk minimization which minimizes a bound on the generalization error. Theorically, SVM can achieve more precise values than Hill Climbing (for our task) because they search in a continuous space while hill climbing is searching discrete values. In addition, using kernels more complex than linear one, they might combine attributes in a better way. Moreover, statistical learning avoids one of the problems of local search, that is to fall in local minimums. In the other hand, SVM computational cost is higher than hill climbing.

4 Evaluation framework

We evaluated both algorithms in the alias assignment task with a corpus of organizations. Developing an IE system in the domain of football (soccer) over the Web, one of the problems we found is that clubs, federations, football players, and many other entities related with football have too long official or real names. Consequently, some nicknames or short names are used widely in either free and structured texts. Almost all texts use this short names to refer to the entities assuming that everyone is able to distinguish which real entity is pointed. For instance, to refer to “Futbol Club Barcelona”, its typical to find “FC Barcelona” or “Barcelona”. We based the results of this paper in our study in the specific domain of football, however, we are presenting a general method for the alias assignment task useful in any other domain.

The corpus consist in 900 football club aliases assigned by hand versus a database with 500 football club entities. Some of them are assigned to more than one club while some others are not assigned because the referring club is not in our database. Each algorithm is trained and tested doing a five-fold cross-validation. Some examples of annotated corpus can be seen in table 1.

Several aliases found across the Web are referring to organizations not included yet in the database. Furthermore, for each alias-entity matching sample (classified as positive) we have almost 500 samples not-matching (classified as negative). This situation would drive accuracy always near 100% even in a blind classifier deciding always negative. In order to have a reasonable evaluation only the set of positive predictions M_p are used in evaluation and compared with the set M_a of examples annotated as positive. The measures used are Precision (1), Recall (2) and F_1 (3). Only F_1 values are shown and compared in this paper.

$$P = \frac{|M_p \cap M_a|}{|M_p|} \tag{1}$$

$$R = \frac{|M_p \cap M_a|}{|M_a|} \tag{2}$$

$$F_1 = \frac{2PR}{P+R}. \tag{3}$$

5 Experiments

We evaluated the task of alias assignment in two experiments. In the first one, we compared the performance of Hill Climbing and SVM using a set of similarity functions. The second is focused on an improvement of feature functions breaking them down in several values representing more specific aspects of their characteristics.

5.1 Algorithm comparison

In the first approach, functions return a value of similarity depending on some criteria. In this case, we are trying to simplify the classification process including only the information we consider important. The larger number of features included, the longer takes an algorithm to train and achieve good results. Based in this principle, we tried to insert as much information as we could in a few values.

The feature functions used in this first experiment (example in figure 1) are the following:

Alias	Assigned entities
Sydney FC	Sydney Football Club
Man Utd	Manchester United Football Club
Nacional	Club Universidad Nacional AC UNAM, Club Deportivo El Nacional, Club Nacional, Club Nacional de Football
Steaua Bucharest	*-not assigned-*
Newcastle United	Newcastle United Jets Football Club Newcastle United Football Club
Krylya Sovetov	Professional Football Club Krylya Sovetov Samara

Table 1: Example of some pairs alias-entity in the football domain

5.1.1 Character-based

- **Prefix and Suffix similarities** count the words of the alias that are the begin (prefix) or the end (suffix) of a word in the entity name.
- **Abbreviations similarity**. If a word s in the alias is shorter than a word t in the entity name they start with the same character and each character of s appear in t in the same order, the function concludes that s is an abbreviation of t. For example "Utd" is an abbreviation of "**U**ni**t**e**d**" and "St" is an abbreviation of "**S**ain**t**".

5.1.2 Token-based

- **Lexical similarity** compares the words between alias A and entity name B without case sensitivity. A classical lexical similarity is:

$$Sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

where $|x \cap y|$ correspond to a function that returns the number of coincidences between words in x and y, and $|x \cup y|$ symbolize the number of different words in the union of x and y.

However, in the case of study, we know that some word in the entity name may not occur in the alias but, almost always, if a word occur in the alias, it must be in the entity name. In other words, an alias use to be a reduced number of words of the entity name. Although, it is difficult to find an alias using words that do not occur in the entity name (it is possible, however). In order to take advantage of this asymmetry in our lexical similarity, words of the alias not appearing in the entity name decrement the similarity as is shown bellow:

$$Sim(A, B) = max(0, \frac{|A \cap B| - |W_a|}{|A \cup B|})$$

where W_a represents the words appearing in A but not in B and `max` function is used taking care that similarity function never returns a value lower than zero.

- **Keywords similarity** is another lexical similarity but avoiding typical domain related words. These kind of words occur in several names and can cause a good lexical similarity when the important words (keywords) are not matching. For example, "Manchester United Football Club" and "Dundee United Football Club" have a good lexical similarity but bad keyword similarity because "football" and "club" are considered typical domain-related words. It uses the same formula as Lexical similarity but not including typical domain-related words in A and B. *Lexical similarity* and *Keywords similarity* could be combined in a lexical similarity weighted with TF-IDF. However, the true contribution of each token to similarity is domain-specific and not always proportional to TF-IDF. Some words have many occurrences but are still important while some others appear few times but are not helpful at all.

5.1.3 Structural

- **Acronyms similarity** looks for a correspondence between acronyms in the alias and capitalized words in the entity name. This feature takes care of the words order because the order of

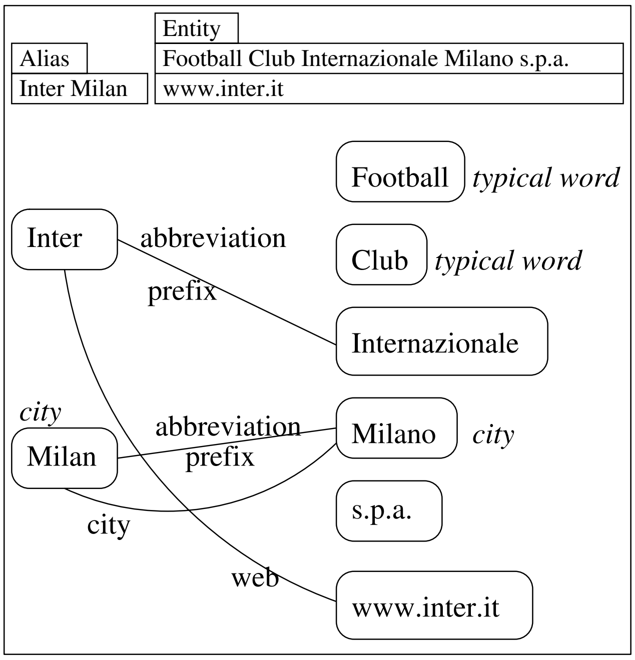

Figure 1: Example of a pair alias-entity and its active features

the characters in an acronym defines the order that words must have in the entity name. An example of acronym is "PSV" which match with "**P**hilips **S**port **V**ereniging Eindhoven".

5.1.4 Semantic

- **City similarity** returns 1 (maximum similarity) only when one word in the alias correspond to a city, one word in the entity name corresponds to a city and both are the same city. In other cases, returns 0 (no similarity). It can be useful when some cities can have different names depending on the language. For instance, "Moscow" and "Moskva" are the same city or "Vienna" and "Wien". This feature requires a world knowledge about cities.

- **Website similarity** function compares the alias with the URL of the organization's website if we have it. Avoiding the first TLD (.com, .de, .es) and sometimes the second (.co.uk, .com.mx) its usual for an organization to register a domain name with the most typical alias for it. The return value of this function is the ratio of words of alias included in the domain name divided by total number of words in the alias. We can use this similarity function because we have more information about the entity than only the official name. In case we don't have this information the return value would be zero.

5.2 Extended features

The second experiment uses *extended* feature functions. This means that most of the feature functions used previously are modified and now they return more than one value breaking down the information. The feature functions are the same but returning a vector of values instead of one value. The classifier may use this extra information if it is helpful for classification. For instance, lexical similarity now returns: number of words in the alias, number of words in the entity name and number of equal words. Combining these values the classifier can achieve a function like our original lexical similarity or maybe a better one.

In this second approach the target is to compare the original feature functions with the extended ones. We choose SVM for this experiment because SVM can use polynomial kernels that may combine attributes in a better way than a linear classifier. Consequently, in this experiment we compare the best classifier obtained in the first experiment with two SVM classifiers using the extended feature functions. One SVM will use a linear kernel while the other will try to take advantage of a quadratic one.

Table 2 shows the modifications realized in each feature function.

6 Results

In our first experiment described in section 5.1, we tried the two algorithms mentioned above, Hill Climbing and SVM, with the feature functions described previously. Table 3 shows the results comparing it with a baseline consisting of some simple rules using only lexical, keywords, acronyms and abbreviations similarities.

The first aspect to emphasize is that the baseline, a simple rule-based classifier, achieves a F_1 measure over 80%. This indicates that the alias assignment task has a high percentage of trivial examples. The use of machine learning and new features may help with difficult ones. Actually, the results show how machine learning algorithms significantly outperform the results obtained by

Feature	Return Values
Prefix	**Pre1**: # words in the alias that are prefixes in the entity name
Suffix	**Suf1**: # words in the alias that are suffixes in the entity name
Abbrev.	**Abr1**: # words in the alias that are an abbreviation of a word in the entity name
Lexical	**Lex1**: # words in the alias **Lex2**: # words in the entity name **Lex3**: # equal words **Lex4**: # equal words case sensitive
Keywords	**Key1**: # keywords int the alias (words excluding typical domain words (football, club, etc)) **Key2**: # keywords in the entity name **Key3**: # of equal keywords
Acronym	**Acr1**: the alias have an acronym (boolean) **Acr2**: the alias acronym matches with capitalized words in the entity name (boolean) **Acr3**: # words in the alias without acronyms **Acr4**: # words in the entity name without words involved in acronyms **Acr5**: # equal words without words involved in acronyms
City	**Cit1**: some word in the alias is a city (boolean) **Cit2**: some word in the entity name is a city (boolean) **Cit3**: both are the same city (boolean)
Website	**Web1**: The entity has a value in the website field (boolean) **Web2**: # words occurring both in the alias and in the URL of the entity

Table 2: Extended features used in the second experiment

the baseline. In the other hand, we find that perform of Hill Climbing and SVM are similar. SVM seems to achieve better results but the difference is not significant since the confidence interval at 95% significance level is 0.8%.

In the second approach we wanted to use the power of SVM combining features and we break down the components of feature functions as explained in section 5.2. SVM may use this extra information if it is helpful for classification. In table 4 two SVM with diffe-

	Baseline	Hill Climbing	SVM
F_1	80.3	87.1	87.9

Table 3: Results of experiment (1) comparing simple rule-based baseline with hill climbing and SVM

Features	Simple	Extended	
Algorithm	SVM	SVM	SVM
Kernel	linear	linear	quadratic
F_1	87.9	93.0	93.0

Table 4: Results of experiment (2) comparing original features with extended features

rent kernels using extended features are compared with results obtained in the first experiment.

The results indicates that extended features outperform the original ones. In the other hand, we can see that a quadratic kernel does not improve the results of the linear kernel.

7 Conclusions

In this paper we have proposed a homogeneous model to deal with the problem of classifying a pair alias-entity into true/false categories. The model consists in using a set of feature functions instead of the state-of-art approach based on distinguishing between a set of lexico-ortographical similarity functions and a set of semantic rules.

Some experiments have been performed in order to compare different configurations for the proposed model. The configurations differ in the set of feature functions and in the discretization strategy for feature weights. Also, two learning techniques have been applied, namely, Hill Climbing and SVMs.

We have seen that Hill Climbing and SVM perform similar. Both algorithms used has some advantages and disadvantages. On one hand, Hill Climbing is simple and fast but has two drawbakcs. The first one is that it looks for weights by steps and it causes that the weights are always discrete values decreasing sometimes the final accuracy. The other drawback is that local search can fall in local minima. Although, it may be palliated by executing the algorithm several times starting with random values. On the other hand, SVM work in a continuous space and learn statistically which avoids the two drawbacks of hill climbing. Although, SVM take longer to be tuned correctly.

In the second experiment, since SVM can handle richer combinations of features when using polynomial kernels, we tested SVMs using a linear kernel and a quadratic one, obtaining similar results. The feature set used in this experiment was a refinement of the previous one, that is, the features contained the same information, but coded with finer granularity. The results pointed out that although the similarity functions used in the first approach produced accurated results, letting the SVM handle all the parameters results in a significative improvement.

References

Bhattacharya, Indrajit and Lise Getoor. 2004. Iterative record linkage for cleaning and integration. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18, New York, NY, USA. ACM Press.

Bilenko, Mikhail and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, New York, NY, USA. ACM Press.

Cohen, W., P. Ravikumar, and S. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks.

Cortes, Corinna and Vladimir Vapnik. 1995. Support-vector networks. In Springer, editor, *Machine Learning*, pages 273–297. Kluwer Academic Publishers, Boston.

Doan, AnHai, Ying Lu, Yoonkyong Lee, and Jiawei Han. 2003. Profile-based object matching for information integration. *IEEE Intelligent Systems*, 18(5):54–59.

Dong, Xin, Alon Halevy, and Jayant Madhavan. 2005. Reference reconciliation in complex information spaces. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96, New York, NY, USA. ACM Press.

Furey, T. S., N. Christianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Hauessler. 2000. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914.

Hernandez, Mauricio A. and Salvatore J. Stolfo. 1995. The merge/purge problem for large databases. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 127–138, New York, NY, USA. ACM Press.

Kalashnikov, Dmitri V. and Sharad Mehrotra. 2006. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Trans. Database Syst.*, 31(2):716–767.

Li, Xin, Paul Morie, and Dan Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 419–424. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

McCallum, Andrew, Kamal Nigam, and Lyle H. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, New York, NY, USA. ACM Press.

Osuna, Edgar, Robert Freund, and Federico Girosi. 1997. Training support vector machines: an application to face detection. *cvpr*, 00:130.

Pasula, H., B. Marthi, B. Milch, S. Russell, and I. Shpitser. 2002. Identity uncertainty and citation matching.

Sarawagi, Sunita and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278, New York, NY, USA. ACM Press.

Shen, W., X. Li, and A. Doan. 2005. Constraint-based entity matching. In *Proceedings of AAAI*.

Skalak, David B. 1994. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *International Conference on Machine Learning*, pages 293–301.

Tejada, Sheila, Craig A. Knoblock, and Steven Minton. 2002. Learning domain-independent string transformation weights for high accuracy object identification. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–359, New York, NY, USA. ACM Press.

Winkler, W. 1999. The state of record linkage and current research problems.