Appears in *Proceedings of the 2005 Dagstuhl Seminar on Machine Learning for the Semantic Web,* Dagstuhl, Germany, February 2005.

Alignments and String Similarity in Information Integration: A Random Field Approach

Mikhail Bilenko and Raymond J. Mooney
{mbilenko, mooney}@cs.utexas.edu
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78705, USA

Abstract

Several problems central to information integration, such as ontology mapping and object matching, can be viewed as *alignment tasks* where the goal is to find an optimal correspondence between two structured objects and to compute the associated similarity score. The diversity of data sources and domains in the Semantic Web requires solutions to these problems to be highly adaptive, which can be achieved by employing probabilistic machine learning approaches. We present one such approach, Alignment Conditional Random Fields (ACRFs), a new framework for constructing and scoring sequence alignments using undirected graphical models. ACRFs allow incorporating arbitrary features into string edit distance computation, yielding a learnable string similarity function for use in tasks where approximate string matching is needed. We outline possible applications of ACRFs in information integration tasks and describe directions for future work.

1 Introduction

In recent years, the vision of the Semantic Web moved towards reality with the spreading popularity of early standards (e.g. XML and SOAP), improved standards coming up to replace them (e.g. OWL), and availability of Web applications that integrate heterogeneous semi-structured data from multiple sources (e.g. CiteSeer and Froogle). To perform well, these applications need to extract information from text, learn and map schemas, and perform data cleaning tasks such as duplicate detection (object matching). Solving these problems accurately requires accounting for the lexical and structural variations across data sources and domains. To deal with this challenge, recent work on information integration tasks has successfully employed machine learning techniques that utilize labeled training data, user-provided constraints, and auxiliary knowledge to adapt algorithms to specific domains [1, 7, 10, 14, 5, 24, 2].

One group of such learning algorithms are methods based on undirected graphical models such as Conditional Random Fields (CRFs) [14] and Relational Markov Networks [23]. Approaches based on these models have performed well on such information integration tasks as named entity extraction [14, 3] and reference matching [16, 18]. Key advantages of these methods stem from their ability to incorporate large numbers of arbitrary features in a principled framework trained with discriminative methods, with the possibility of inducing new informative features dynamically [15]. As a result, CRFs and RMNs outperform techniques based on generative models with strong independence assumptions, such as Naive Bayes or Hidden Markov Models.

In this work, we introduce Alignment Conditional Random Fields (ACRFs), an undirected graphical model for the *sequence alignment* problem, which is the task of finding a correspondence between two sequences (e.g. strings) with an associated similarity score. Edit distance algorithms [11] solve exactly this problem; our approach results in a *learnable* edit distance algorithm that can utilize a large number of arbitrarily defined parameters whose weights are learned from training data. The flexibility in the choice of parameters is particularly relevant for information integration tasks, where similarity between strings may depend on a variety of domain-dependent features, e.g. spelling variations or abbreviations-related features.

ACRFs have applications in integration tasks for the Semantic Web where similarity between strings or sequences is estimated. Accurate string distance computation is essential for object matching methods as well as other problems such as ontology mapping and data cleaning where accurate string comparisons are required.In the following sections, we describe the model and its relevance to Semantic Web applications, as well as possible extensions to other information integration problems that we hope to address in future work.

2 Alignments in Information Integration

Ontology mapping and object matching are two problems central to successful integration of data from multiple sources. The two problems have different goals: ontology mapping attempts to reconcile *meta-data* for each information source, while the object matching seeks to identify *data instances* that represent the same underlying entity. The two tasks are similar, however, in that both are concerned with finding a correspondence between pairs of structured objects and computing similarity between them. Additionally, both tasks involve computing string similarity between textual attributes describing either categories in the ontology, or the objects' string attribute values.

Ontology Mapping as Alignment. In previous work, two types of mappings were considered: (1) *one-to-one* mapping, where an attribute in one schema may be matched to one attribute from another schema [20], and (2) *complex* mapping, where combinations of attributes may be matched between schemas [6]. Both of these mapping types can be represented as alignments where for one-to-one mapping individual attributes from two schemas are linked, while for complex mapping correspondences may link groups of multiple attributes.

String Matching as Alignment. String edit distance computation is often used when matching objects described by textual attributes or when string similarity estimates are needed for data cleaning and ontology mapping tasks [17, 2, 4, 8]. Since edit distance computation is equivalent to finding the optimal alignment between two strings, accurately computing alignment scores between sequences is an essential step for robust information integration approaches.

Alignments are typically constructed using dynamic programming algorithms that compute the optimal correspondence of structures (e.g. strings or trees) along with a distance score [11, 13]. Score computation depends on per-element *transformation costs* representing the relative importance of mismatches in the alignment. The costs are commonly set manually, and are difficult to tune when the number of parameters is large.

Classic approaches to alignment scoring such as variants of edit distance computation can be represented by finite-state automata which encode possible alignments via emissions resulting from state transitions [11]. In the last decade, probabilistic analogs of these algorithms were developed based on Hidden Markov Models [21, 9, 2]. These approaches allow learning string edit distance parameters via maximum-likelihood training on a corpus of matching sequences. However, Hidden Markov Models cannot account for arbitrary features or long-range dependencies in the alignment since it is modeled as a generative process under the Markov assumption. We attempt to overcome the limitations of HMMs by proposing a learnable framework for sequence alignment based on an undirected graphical model.

3 The ACRF Model

We view the task of computing similarity between strings through a probabilistic framework, where similarity is equivalent to the conditional probability of the most likely alignment of the strings. All possible alignments of two strings $\boldsymbol{x}$ and $\boldsymbol{y}$ can be represented by an Alignment Conditional Random Field (ACRF) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where states $\mathcal{V} = \{v_{i,j}\}_{i=0..|\boldsymbol{x}|, j=0..|\boldsymbol{y}|}$ are located in the nodes of a $(|\boldsymbol{x}|+1) \times (|\boldsymbol{y}|+1)$ lattice, connected by edges in set $\mathcal{E}$. Each state $v_{i,j}$ corresponds to an unobserved discrete random variable $o_{i,j}$ that encodes the *label* in which the alignment of prefixes $\boldsymbol{x}_{[1:i]}$ and $\boldsymbol{y}_{[1:j]}$ ends. A three-label alphabet $\mathcal{A}_3 = \{S, I_1, I_2\}$ is typically used for text applications. Label S denotes a substitution (including exact matching), I_1 denotes insertion into the first string, and I_2 denotes insertion into the second string. Larger label alphabets can be used to represent more complex alignment models.

The edge structure $\mathcal{E}$ is determined by the label alphabet $\mathcal{A}$. Each edge represents a conditional dependency between two state variables. For the three-label alphabet $\mathcal{A}_3$, three types of edges are instantiated, which intuitively correspond to the three types of edit operations. A vertical edge is created between every pair of states $(v_{i-1,j}, v_{i,j})$ that represents a possible insertion of the i-th symbol into the first string (corresponding to $o_{i,j} = I_1$); a horizontal edge connects every pair of states $(v_{i,j-1}, v_{i,j})$ representing a possible insertion of the j-th symbol into the second string (if $o_{i,j} = I_2$); a diagonal edge connecting states $v_{i-1,j-1}$ and $v_{i,j}$ corresponds to substitution of x_i for y_j (corresponding to $o_{i,j} = S$).

A sample ACRF is shown on Fig.1. The figure omits two non-shaded observed states that encapsulate the two strings and are connected to to every shaded, non-observed state in the lattice, representing the dependency of each label assignment on the actual strings.

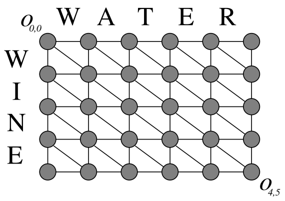

Figure 1: A sample alignment CRF

Let $O = \{o_{i,j}\}_{i=0..|\boldsymbol{x}|, j=0..|\boldsymbol{x}|}$ denote the complete assignment of labels to the random variables in $\mathcal{G}$ corresponding to an alignment of strings $\boldsymbol{x}$ and $\boldsymbol{y}$. The similarity of $\boldsymbol{x}$ and $\boldsymbol{y}$ is then equivalent to the conditional probability of a label assignment to O, and can be factorized as follows by the Hammersley-Clifford theorem [12]:

$$\begin{aligned} p_{\boldsymbol{\lambda}}(O|\boldsymbol{x},\boldsymbol{y}) &= \frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{y})} \prod_{c \in C} \Phi(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y}) \\ &= \frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x},\boldsymbol{y})} \exp \sum_{c \in C} \boldsymbol{\lambda} \boldsymbol{f}(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y}) \qquad (1) \end{aligned}$$

where $\mathcal{C}$ is the clique set of $\mathcal{G}$, $\boldsymbol{o}_c$ is the subset of O corresponding to vertices in clique c, $\Phi(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y})$ is a clique potential function represented by an exponentiated sum of binary clique features $\boldsymbol{f}(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y}) = \{f_k(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y})\}$

weighted by a vector of real weights $\boldsymbol{\lambda}$: $\Phi(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y}) = \exp \boldsymbol{\lambda} \boldsymbol{f}(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y})$. $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y})$ is the normalization factor over all possible label assignments:
$Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_O \prod_{c \in C} \Phi(\boldsymbol{o}_c, \boldsymbol{x}, \boldsymbol{y})$. We describe possible features in detail in Section 3.3.

3.1 Computing Edit Distance with ACRFs

Since $\mathcal{G}$ contains $2(|\boldsymbol{x}|+1)(|\boldsymbol{y}|+1)$ triangular cliques, exact inference for the general factorization in Eqn.(1) is computationally prohibitive. However, the semantics of edge structure result in constraints that allow a different factorization of the model, which permits exact inference via the Viterbi algorithm. If edge potentials are defined to be non-zero only when the label on the end state corresponds to that edge type, it can be proved that the non-zero potential edges form a tree rooted at vertex $v_{o,o}$. With this constraint, it is possible to use exact inference to estimate the distribution $p_{\boldsymbol{\lambda}}(O|\boldsymbol{x}, \boldsymbol{y})$. Fig.2 below demonstrates an example of a possible label assignment to the ACRF with the non-zero potential edges highlighted and the corresponding alignment below.

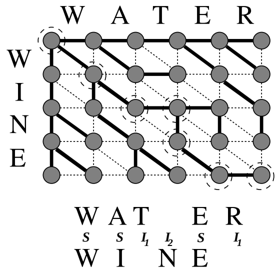

Figure 2: A possible label assignment with the corresponding alignment

Then, we can factorize $p_{\boldsymbol{\lambda}}(O|\boldsymbol{x}, \boldsymbol{y})$ as follows:

$$p_{\boldsymbol{\lambda}}(O|\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y})} \exp \sum_{i=0}^{|\boldsymbol{x}|} \sum_{j=0}^{|\boldsymbol{y}|} \boldsymbol{\lambda} \boldsymbol{f}(o_{i,j}, \pi(o_{i,j}), \boldsymbol{x}, \boldsymbol{y}, i, j) \quad (2)$$

where $\pi(o_{i,j}) = o_{i',j'}$ such that $e = (v_{i',j'}, v_{i,j})$ is the edge with a non-zero potential corresponding to the value of $o_{i,j}$. $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y})$ is the normalization factor: $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_O \exp \boldsymbol{\lambda} \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}, O)$, where $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}, O)$ is the *global* feature vector: $\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}, O) = \sum_{i=0}^{|\boldsymbol{x}|} \sum_{j=0}^{|\boldsymbol{y}|} \boldsymbol{f}(o_{i,j}, \pi(o_{i,j}), \boldsymbol{x}, \boldsymbol{y}, i, j)$.

Given the factorization in Eqn.(2), the forward, backward, and Viterbi algorithms in alignment CRFs are analogous to those in pairwise HMMs [9]. We describe the forward algorithm in detail.

Given strings $\boldsymbol{x}$ and $\boldsymbol{y}$, an 3×3 transition matrix M_{ij} is computed for each state $v_{i,j}$ as follows, where entry $M_{ij}(a, a')$ encodes the forward value for $o_{i,j} = a$ after the transition from previous state with label a':

$$M_{ij}(a, a') = \exp \boldsymbol{\lambda} \boldsymbol{f}(a, a', \boldsymbol{x}, \boldsymbol{y}, i, j) \quad (3)$$

Three $(|\boldsymbol{x}|+1) \times (|\boldsymbol{y}|+1)$ forward matrices A_S, A_{I_1}, and A_{I_2} are created, where each entry $A_a(i, j)$ corresponds to the probability of the alignment of substrings $\boldsymbol{x}_{[1:i]}$ and $\boldsymbol{y}_{[1:j]}$ ending with label a. Each matrix A_a is computed using dynamic programming as follows:

$$A_a(i,j) = \begin{cases} 1, & i=0, j=0 \\ \sum_{a' \in \mathcal{A}_3} A_{a'}(i_p, j_p) M_{ij}(a, a') & \text{otherwise} \end{cases} \quad (4)$$

where i_p and j_p are determined by the edge corresponding to label a (vertical for $a = I_1$, horizontal for $a = I_2$, diagonal for $a = S$).

The forward matrix also provides the normalization factor value: $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{a \in \mathcal{A}_3} A_a(|\boldsymbol{x}|, |\boldsymbol{y}|)$. Actual string similarity score is obtained by running the Viterbi algorithm. The overall computational complexity of forward, backward, and Viterbi algorithms is $O(|\boldsymbol{x}|, |\boldsymbol{y}|)$, same as for classic edit distance.

3.2 Learning in ACRFs

Given a training corpus of N pairs of matching strings $D = \{(\boldsymbol{x}_k \sim \boldsymbol{y}_k)\}_{k=1}^N$, the goal is to maximize the conditional log-likelihood of observing the alignments:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\lambda}) &= \sum_k \log p_{\boldsymbol{\lambda}}(\boldsymbol{x}_k \sim \boldsymbol{y}_k | \boldsymbol{x}_k, \boldsymbol{y}_k) \\ &= \sum_k \left(\boldsymbol{\lambda} \boldsymbol{F}(\boldsymbol{x}_k, \boldsymbol{y}_k) - \log Z_{\boldsymbol{\lambda}}(\boldsymbol{x}_k, \boldsymbol{y}_k) \right) \end{aligned} \quad (5)$$

To find $\boldsymbol{\lambda}$ that maximizes $\mathcal{L}(\boldsymbol{\lambda})$ we can follow the L-BFGS method previously introduced for linear CRFs by Sha and Pereira [22]. Overfitting can be controlled by introducing a prior $p(\boldsymbol{\lambda})$ over the weight vector and maximizing $\mathcal{L}(\boldsymbol{\lambda}) + \log p(\boldsymbol{\lambda})$.

3.3 Features in ACRFs

The principal advantage of the proposed model comes from the ability to incorporate a variety of representative and possibly overlapping features. Following are some of the features that can be employed for comparing textual strings:

- Base features that encode symbol-specific potentials, such as costs of edit operations for particular characters. E.g., feature $f(o_{i,j}, \pi(o_{i,j}), \boldsymbol{x}, \boldsymbol{y}, i, j) = 1$ iff $o_{i,j} = S$, $x_i =$ `' '`, $y_j =$ `','` corresponds to the cost of substituting a space with a comma.
- Next-symbol and previous-symbol potentials, e.g., feature $f(o_{i,j}, \pi(o_{i,j}), \boldsymbol{x}, \boldsymbol{y}, i, j) = 1$ iff $o_{i,j} = I_1$ and $x_{i+1} =$ `'.'` encodes insertions before a dot, representing possible abbreviations.
- Local features specific to natural language that capture such artifacts as capitalization, common suffixes and typographic errors, and abbreviations, e.g. a feature that encodes whether a sequence of characters being inserted is `'Inc.'`.

- Domain-specific features for common domains such as postal addresses, proper names, or titles, e.g., feature $f(o_{i,j}, \pi(o_{i,j}), \boldsymbol{x}, \boldsymbol{y}, i, j) = 1$ iff $o_{i,j} = I_2$, $y_j \in \texttt{Digits}$ represents costs of inserting digits.

Along with the base features, it is possible to incorporate conjunction features induced automatically using the method previously used for linear-chain CRFs [15]. Overall, ACRFs provide a flexible method for edit distance estimation that can be tuned to individual domains.

4 Future Work and Conclusions

We described alignment conditional random fields, a probabilistic framework for sequence alignment and similarity computation. The model allows using a variety of informative features whose weights are learned. This capacity is particularly important for information integration tasks since it provides a way to adapt string similarity computation to the domain at hand. We are currently working on experimental evaluation of the framework and comparing it to prior approaches to learnable string similarity.

In future work, we would like to extend our framework to represent alignments of trees, yielding a probabilistic approach to ontology mapping. Since computational complexity of such algorithms is typically at least quadratic in structure size, developing approximate methods is an important issue. Since a number of integration tasks emloys string similarity computations (e.g. data cleaning, ontology mapping), incorporating ACRFs as a component of these tasks is another area for future research.

Information extraction, object matching, and ontology mapping tasks are related since they represent a continuum of information integration tasks. Developing *joint methods* for these problems is an important challenge for future research, since such approaches could leverage uncertainty between the tasks. Advantages of a integrated approach to information extraction and coreference resolution have been shown in recent work [19, 25], and extending the continuum of joint models to ontology mapping may provide further improvements in dealing with heterogeneous, semi-structure data from multiple sources.

References

[1] R. Agrawal and R. Srikant. On integrating catalogs. In *Proc. of WWW-2001*.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of SIGKDD-2003*.

[3] R. C. Bunescu and R. J. Mooney. Collective information extraction with relational Markov networks. In *Proc. of ACL-2004*.

[4] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proc. of IJCAI IIWeb-2003 Workshop*.

[5] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proc. of SIGKDD-2002*.

[6] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMap: Discovering complex semantic matches between database schemas. In *Proc. of SIGMOD-2004*.

[7] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. of SIGMOD-2001*.

[8] A. Doan, Y. Lu, Y. Lee, and J. Han. Object matching for information integration: A profiler-based approach. In *Proc. of IJCAI IIWeb-2003 Workshop*.

[9] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[10] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proc. of AAAI-2000*.

[11] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[12] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1971.

[13] T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143:137–148, 1995.

[14] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML-2001*.

[15] A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. of UAI-2003*.

[16] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS 17*, 2005.

[17] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proc. of SIGKDD-1996*.

[18] Parag and P. Domingos. Multi-relational record linkage. In *Proc. of SIGKDD MRDM-2004 Workshop*.

[19] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS-15*, 2003.

[20] E. Rahm and P. A. Bernstein. On matching schemas automatically. *VLDB Journal*, 10(4):334–350, 2001.

[21] E. S. Ristad and P. N. Yianilos. Learning string edit distance. In *Proc. of ICML-1997*.

[22] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL-2003*.

[23] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. of UAI-2002*.

[24] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of SIGKDD-2002*.

[25] B. Wellner, A. McCallum, F. Peng, and M. Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proc. of UAI-2004*.