Boosting Relational Sequence Alignments

Andreas Karwath
Institut für Informatik
Albert-Ludwigs Universität
Georges-Köhler-Allee 79
79110 Freiburg, Germany
karwath@informatik.uni-freiburg.de

Kristian Kersting
Dept. of Knowledge Discovery
Fraunhofer IAIS
Schloss Birlinghoven
53754 St Augustin, Germany
kristian.kersting@iais.fraunhofer.de

Niels Landwehr
Dept. of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Heverlee, Belgium
niels.landwehr@cs.kuleuven.be

Abstract

The task of aligning sequences arises in many applications. Classical dynamic programming approaches require the explicit state enumeration in the reward model. This is often impractical: the number of states grows very quickly with the number of domain objects and relations among these objects. Relational sequence alignment aims at exploiting symbolic structure to avoid the full enumeration. This comes at the expense of a more complex reward model selection problem: virtually infinitely many abstraction levels have to be explored. In this paper, we apply gradient-based boosting to leverage this problem. Specifically, we show how to reduce the learning problem to a series of relational regressions problems. The main benefit of this is that interactions between states variables are introduced only as needed, so that the potentially infinite search space is not explicitly considered. As our experimental results show, this boosting approach can significantly improve upon established results in challenging applications.

1. Introduction

Sequential data is ubiquitous and of interest to many communities. It can be found in virtually all application areas of machine learning, including computational biology, activity recognition, information extractions, among others. Therefore, it is not surprising that sequential data has been the subject of active research for decades. One of the many tasks investigated for sequential data is that of sequence alignment. Informally speaking, a sequence alignment is a way of arranging sequences to emphasize their regions of similarity. For this task, dynamic programming (DP) has become the standard model. Classical DP approaches, however, require the explicit state enumeration in the reward model, which is often impractical: the number of states grows very quickly with the number of domain objects and relations among these objects such as in web communities evolving over time. In contrast, relational sequences alignment aims at exploiting symbolic structure in the reward model to avoid the full alphabet enumeration.

Crucial to the performance of any alignment method (relational or not) is an accurate reward model for transforming one sequence into another. In the relational case, in which e.g. there is a varying number of objects and relations among them, specifying the reward model by hand is difficult if not impossible. Even estimating it from data is challenging as virtually infinitely many abstraction levels have to be explored. Triggered by the observation that *finding many rough rules of thumb of how to transform one sequence into another can be a lot easier than finding a single, highly accurate reward model*, we apply Friedmann's gradient boosting [3] to learning the reward model. That is, we represent reward models as weighted sums of regression models grown in a stage-wise optimization. The benefits of this approach, which we call BOOSTEDREAL, are twofold. First, interactions between states variables are introduced only as needed, so that the potentially infinite search space is not explicitly considered. Second, existing off-the-shelf regression learners can be used to deal with propositional and relational domains in a unified way. As our experimental results show, this approach can significantly improve upon established results in challenging applications.

We proceed as follows. We start off by discussing related work. Afterwards, we briefly review DP-based alignment approaches in Section 3. In Section 4, we then develop BOOSTEDREAL. Before concluding, we present our experimental evaluation and its results.

2. Related Work

Surprisingly few works have investigated sequences of complex objects so far. Ketterlin [11] considered the clustering of sequences of complex objects but did not employ logical concepts. Likewise, Jiang *et al.* [8] and Weskamp

et al. [21] proposed alignment algorithms for trees respectively graphs. Lee and De Raedt [13] and Jacobs [7] introduced ILP frameworks for reasoning and learning with relational sequences. Recently, Tobudic and Widmer [19] used relational instance-based learning for mining music data, where sequential, relational information is employed. To the best of our knowledge, however, none of these works investigate the alignment of relational sequences.

Indeed within bioinformatics most advances of sequence alignment for biological sequence analysis (see [2] for a good overview) have been made by incorporating additional sources of information such as sequence profiles or secondary structure predictions. For instance, Sato and Sakakibara [18] use conditional random fields (CRFs) to discriminatively learn edit distances for trees. CRFs allow to use arbitrary, even relational features [5] to define the potential functions involved. CRFs are, however, considerably more difficult than DP approaches and as we will show boosted DP methods can yield competitive performance. Recently, kernel methods have been developed [20]. Due to the finite alphabet assumption the learned reward models do not easily generalize to symbols not seen at training time.

Most similar to our work, Parker *et al.* [16] also proposed gradient-based boosting for learning the reward model. They do not, however, consider relational domains and they consider a different learning setting, which cannot easily be applied in our experimental domains: whereas we only assume a given set of classified sequences, Parker *et al.* assume properly aligned pairs of query and target sequences to which a correct score is given.

3. Alignments by Dynamic Programming (DP)

The Needleman-Wunsch (NW) algorithm [14] for **global alignment** is one well known DP-based alignment algorithm. It finds the alignment of two sequences with the maximal overall similarity w.r.t. a given pairwise similarity model. In biological domains, the similarity model is typically represented as matrices over pairwise (dis-)similarity scores of pairs of amino acids. The scores, or costs, associated with a match or mismatch between two amino acids, reflect to some extent the probability that this change in amino acids has occurred over time in evolution.

To construct an alignment NW constructs a matrix with $m+1$ columns and $n+1$ rows for two sequences of length m and n. After initialization of the first row and the first column, the matrix is then progressively filled with the maximum score as follows:

$$M_{i,j} = \max \begin{cases} M_{i-1,j-1} + S_{i,j} & \text{: (mis-)match} \\ M_{i,j-1} + g & \text{: insertion} \\ M_{i-1,j} + g & \text{: deletion} \end{cases} \quad (1)$$

where $S_{i,j}$ is the pairwise similarity of the symbols in an alphabet $\mathcal{A}$, e.g. the alphabet of amino acids, and g reflects a linear gap (insertion or deletion) penalty. The overall score of the alignment can be found in cell $M_{m,n}$. To provide more flexibility, one often employs affine gaps, i.e., different kind of gap costs for opening a gap (g_{open}) or for extending one (g_{extend}). To discourage the splitting of connected regions due to the enforcement of a gap in the middle of the alignment, commonly extra gaps are allowed to be inserted at the end and at the beginning at either sequence in the alignment with no or relatively low additional costs. We refer to these costs as $g_{paddingFront}$ and $g_{paddingBack}$.

The alignment algorithms discussed so far assume a given similarity measure $S_{i,j}$. Typically, this similarity measure is flat because the considered sequences consist of flat symbols. Many sequences occurring in real-world problems such as in computational biology, planning, or user modeling, however, exhibit internal structure, that can elegantly be represented as "objects" in a relational logic.

Relational sequence alignment simply denotes the alignment of sequences of such structured objects. More formally, the relational alignment problem can be defined as follows: **Given** two sequences of logical objects $\mathtt{x} = \langle \mathtt{x}_i \rangle_{i=1}^{n}$, $n > 0$, and $\mathtt{y} = \langle \mathtt{y}_i \rangle_{i=1}^{m}$, $m > 0$, and let $S_{i,j}$ be a similarity measure indicating the score of aligning object $\mathtt{x}_i$ with object $\mathtt{y}_j$. Then, the **global alignment** problem seeks to **find** the match with highest score of both sequences in their entirety. Notice that this is a generic problem definition that generalizes the classical setting as it leaves the type of logical objects encountered open. So, given an appropriate reward model, one can indeed apply the classical DP machinery to solve the more general setting. Classical DP approaches, however, require the explicit state enumeration in the reward model, which is often impractical: the number of states grows very quickly with the number of domain objects and relations among these objects. Relational alignment aims at exploiting symbolic structure in the reward model to avoid the full alphabet enumeration of classical approaches. Specifically, they replace the flat similarity measure $S_{i,j}$ in Eq. (1) by a structured one, depending on the type of logical objects encountered. Several such distance measures have been developed, see for example [17]. We will now briefly review two of the existing measures.

As the first example, consider sequences of **ground atoms**. In general, atoms are expressions of the form $\mathtt{p}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$, where $\mathtt{p}/n$ is a predicate symbol (of arity n) and the $\mathtt{t}_i$ are terms. Terms are built from constants, variables (placeholders), and functor symbols. Ground atoms do not contain variables. For instance, $\mathtt{st}(\mathtt{pl}, \mathtt{sh})$ could denote a strand of certain type and length. The Nienhuys-Cheng measure [15] treats ground atoms as hierarchies, i.e., trees, where the top structure is most important and the deeper, nested sub-structures are less important. Let $\mathcal{A}$ denote the set of all symbols, then the

Nienhuys-Cheng distance d_{nc} is inductively defined as follows. For all *matching constants* $\mathtt{c}/0 \in \mathcal{A}$ the distance is $d_{nc}(\mathtt{c},\mathtt{c}) = 0$; For all atoms within *non-matching predicate* symbols, $d_{nc}(\mathtt{p}(\mathtt{t_1},\ldots,\mathtt{t_n}),\mathtt{q}(\mathtt{s_1},\ldots,\mathtt{s_m})) = 0$; otherwise, if the *predicate symbols match*, we recursively compute $\frac{1}{2\mathtt{n}}\sum_{\mathtt{i}=1}^{\mathtt{n}} d_{nc}(\mathtt{t_i},\mathtt{s_i})$. That is, for different symbols the distance is one; however, when the symbols are identical, the distance linearly decreases with the number of arguments that have different values, and is at most 0.5. The intuition is that *longer tuples are more error-prone* and that *multiple errors in the same tuple are less likely*. At this point the reader may verify that $d_{nc}(\mathtt{st}(\mathtt{pl},\mathtt{sh}),\mathtt{st}(\mathtt{mi},\mathtt{sh})) = \frac{1}{2\cdot 2}\cdot(0+1) = 0.25$. In other words, the distance smoothes the dichotomic identity function of the propositional case, i.e., the $\mathtt{p}/0$ case. Finally, let us note that to solve the corresponding relational alignment problem, we indeed simply set $S_{i,j} = 1 - d_{nc}(\mathtt{x_i},\mathtt{y_i})$ in Eq. (1).

As the second example, consider sequences of more complex logical objects, namely **interpretations**. An interpretation is a set of ground atoms $\{\mathtt{st}(\mathtt{pl},\mathtt{sh}),\mathtt{st}(\mathtt{mi},\mathtt{sh}),\ldots\}$ specifying which ground atoms are true at a certain time. Typically, one applies a closed-world assumption, i.e., everything not listed is assumed to be false. For interpretations, a different similarity function has to be chosen. A simple way to measure the distance of two interpretations (or sets of logical objects) A and B is the so-called Hausdorff metric d_h:

$$d_h(A,B) = \max\left[\max_{a\in A}\min_{b\in B} d(a,b), \max_{b\in B}\min_{a\in A} d(a,b)\right] .$$

As $d(x,y)$ the Nienhuys-Cheng distance measure d_{nc} can be employed. Other, more complex distance measures for interpretations exist, cf. [17].

We have implemented a global alignment algorithm in C and Prolog that can be used with these kind of distance measures. The system, called REAL, allows for affine gap costs g_{open} and g_{extend}. The distances are defined in Prolog and dynamically calculated. REAL also allows two distinct padding gap costs $g_{paddingFront}$ and $g_{paddingBack}$ so that we can assign different gap costs to specific symbols at the beginning or at the end of a sequence.

Finally, let us note that sequence alignment naturally allows one to classify sequences: align the query sequence with all training sequences and perform a k-nearest neighbor search based on the distances computed. In our experiments, we simply use a 1-nearest neighbor search.

4. Boosting the Reward Model

We have seen how to align sequences using DP given a reward model. In real-world applications, the reward model is typically highly complex and domain specific and, in turn, difficult to specify by hand. It is therefore not surprising that techniques for automatically learning propositional or continuous reward models from examples have been developed. Without extensive feature engineering, however, it is difficult – if not impossible – to apply them to relational domains; there are simply to many possible features. Moreover, they typically assume that the training examples are query and target sequences, properly aligned, and assigned a correct score. Consider to classify protein sequences (for more details, see the experimental section). Here, properly aligned sequences are typically not given. Instead, we have clusters of "similar" proteins. Therefore, we consider a more general learning setting: **Given** a set of classified sequences $q_i^{c_i} = \langle q_{i,j}^{c_i}\rangle_{j=1}^{n}, i = 1,2,\ldots,n$, where c_i denotes the class label $c_i \in \{c_1,c_2,\ldots,c_k\}$ of the i-th sequences, **find** a reward model $r : \mathcal{S}\times\mathcal{S} \mapsto \mathcal{R}$ that maximizes the intra-class sequence similarity and minimizes the inter-class sequence similarities as much as possible. We will now derive our boosting approach to solving this problem.

Boosting is a method for incrementally building linear combinations of "weak" models to generate a "strong" model. Given data, a dictionary of weak learners, and a loss function L, a boosting algorithm sequentially finds linear combinations h of weak learners that minimize the cumulative loss $\sum_i L(z_i,h)$. It has been shown that boosting can be described as a (functional) gradient descent algorithm, where the weights in each step of the algorithm corresponds to the gradient of a loss function at the "current" fit. Let us first define our loss function.

Motivated by Parker *et al.* [16], the main idea is to convert Friedman *et al.*'s [4] loss function from LogitBoost to a loss meaningful to our learning problem, then to use a function approximator estimating the gradient function at points unseen in the training data. Specifically, let $F(x,y,a)$ be the total, additive reward for aligning sequences x and y with the series of alignment operations a. We define the margin for a sequence q^c of class c as

$$\max_{t\in c} F(q^c,t^c,a_{q,t}) - \max_{t\notin c} F(q^c,t^k,\hat{a}_{q,t}) . \quad (2)$$

That is, we view the margin as the difference, for a given query sequence q^c, between the best target in the same cluster and the highest scoring target in the incorrect clusters. This then leads to the following LogitBoost-like loss function for training examples q^c

$$L(q^c,F) = \log\left(1 + e^{F(q^c,t^k,\hat{a}_{q,t}) - F(q^c,t^c,a_{q,t})}\right) . \quad (3)$$

Notice that the loss function monotonically decreases with increasing margin as desired.

Because the reward model is additive, we can express our scoring function F as a sum of the rewards for the various events in the alignment. To do this, let us define $r(x,y)$ to represent a combination of the reward functions discussed above for logical objects x and y. Also define

$f(x, y, q^c, t^c, a_{q,t})$ to be the number of times that object x is replaced by y in the alignment $a_{q,t}$ of sequence q with sequence t. Thus we can express the scoring function as:

$$F(q^c, t^c, a_{q,t}) = \sum\nolimits_{x,y} r(x, y)[f(x, y, q^c, t^c, a_{q,t})] \ . \quad (4)$$

Applying gradient boosting means now to iteratively learn the reward model $r(x, y)$ in order to minimize the loss function (3) using Eq. (4) inside. Initially, we set the reward model to some arbitrary (perhaps uninformative) function. Then, we compute r_{k+1}, i.e., the reward model after k iterations by approximating the functional gradient δ_{k+1} of the loss function with respect to r_k and setting $r_{k+1} = r_k - \eta\delta_{k+1}$ (η being the learning rate). This is likely to move r_{k+1} in a direction that decreases the loss function. We iterate until a stopping condition is reached. The key step of this process is to compute the approximate functional gradients, which we now describe.

Define $\Delta f_i(x, y)$ for the i-th training examples to be

$$\Delta f_i(x, y) = f(x, y, q^c, t^c, a_{q,t}) - f(x, y, q^c, t^k, \hat{a}_{q,t}) \ .$$

The cumulative loss over the training set is then given by

$$L = \sum\nolimits_i \log\Big(1 + \exp(-\sum\nolimits_{x,y} r(x, y)[\Delta f_i(x, y)])\Big)$$

and the function gradient at pair (x, y) as $\delta_{k+1}(x, y) =$

$$\frac{\partial L}{\partial r_k(x, y)} = \sum\nolimits_i \frac{\Delta f_i(x, y)}{1 + e^{\sum_{x',y'} r(x',y')\Delta f_i(x',y')}} \ .$$

Putting all together, we iterate over our training examples and compute pairs of (logical) objects weighted by δ_{k+1}. On these regression examples, we induce a regression model that approximates the function gradient, update our current model as described above, and iterate.

In principle, any kind of regression learner can be used. In our experiments, we grow regression trees, which basically works as follows. It starts with the empty tree and repeatedly searches for the best test for a node according to some splitting criterion such as weighted variance. Because we are dealing with logical objects, rather than using attribute-value or threshold tests in a node of the tree, we employ logical queries [1], which are generated by basically choosing a literal, unifying variables, and grounding variables. Next, the examples E in the node are split into Es (success) and Ef (failure) according to the test. For each split, the procedure is recursively applied, obtaining subtrees for the respective splits. As splitting criterion, we use the weighted variance on Es and Ef. We stop splitting if the variance in one node is small enough or a depth limit was reached. In leaves, the average regression value is predicted. We call the resulting algorithm "boosted relational alignment" or short BOOSTEDREAL.

5. Experimental Evaluation

Our general intention is to investigate whether BOOSTEDREAL is useful in real-world applications. More precisely, we investigated the following two questions: **Q1** Is BOOSTEDREAL able to compete with state-of-the-art approaches? **Q2** Is it possible to extract meaningful knowledge when aligning sequences of interpretations?

(Q1) Protein Fold Classification: This is an intrinsically difficult task in molecular biology. The general assumption made is that knowing what class/fold a newly-found protein belongs to, one can infer its function, which in turn helps understanding the processes running in the studied organism so that certain proteins might be identified as potential drug targets.

Here, we consider the task of predicting the actual fold of proteins within the the five most populated folds in the SCOP [6] class *Alpha and beta proteins (a/b)*, i.e., folds c.1, *TIM beta/alpha-barrel*, c.2, *NAD(P)-binding Rossmann-fold domains*, c.23, *Flavodoxin-like*, c.37, *P-loop containing nucleotide triphosphate hydrolases*, and c.55, *Ribonuclease H-like motif*. The examples are sequences of secondary structure elements of proteins which are similar in their three dimensional shape, but in general do not share a common ancestor (i.e. are not homologous). In total there are 2086 sequence distributed over the folds as follows: (c.1: 721), (c.2: 360), (c.23, 274), (c.37, 441), (c.55,290). The data set was generated using the ASTRAL database for the SCOP version 1.63[1] and has been used in a number of publications to evaluate the performance of relational sequence learning algorithms [10, 9, 5, 12]. Each sequence object is represented as a logical atom, either a strand or a helix. The helical objects are further described using three attributes with information about the type of helix (α, π, or *3-10*), its orientation (*right* or *left*), and its discritized length as measured in number of amino acids (*short*, *medium*, or *long*). Similarly, the strand objects contain information about their orientation to other strands (generating a β-sheet), namely *parallel*, *anti-parallel*, or *none*. The length of the strand object is also discretized into three length.

We performed a 10-fold stratified cross validation. As the question of finding the appropriate gap costs in computational biology is commonly answered by a trial and error approach, we have arbitrarily chosen the following gap costs: $g_{open} = 1.0$, $g_{extend} = 0.1$, $g_{paddingFront} = 0.0$, and $g_{paddingBack} = 0.0$ and let the approach investigate the best suited gap costs for this particular application. We have used the same initial gap costs throughout the paper. The learning rate was arbitrarily set to $\eta = 0.1$.

The results shown in Table 1 are quite convincing. Even without boosting, the 1-nearest neighbour classifier based

[1] http://astral.berkeley.edu/scopseq-1.63.html

Method	Year and Ref.	Predicted Accuracy
LoHMMs	2006, [9]	74.00
LoHMMs + FK	2004, [10]	84.00
r-grams	2007, [12]	86.00
TildeCRF	2006, [5]	92.96
REAL	—	93.99 ($\pm$0.35)
BOOSTEDREAL	—	97.82 ($\pm$0.18)

Table 1. Comparison of the performances of REAL and BOOSTEDREAL to several state-of-the-art approaches on the protein dataset. Already REAL using different gap costs is competitive but BOOSTEDREAL significantly outperforms all methods (paired t-test, $p = 0.05$).

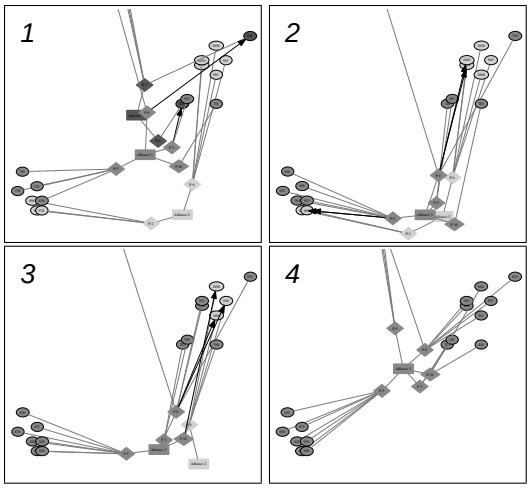

Figure 1. A (partial) sequence of game states in Travian. Circular nodes indicate cities. Rhombical nodes indicate players, and connections denote ownership. Rectangular nodes indicate alliances, and are connected to all players within the alliance. Players and cities are color-coded according to alliance affiliation. Arrows indicate conquest attacks.

on relational alignment outperforms more sophisticated approaches such as relational CRFs [5]. However, boosting increases the performance significantly (paired t-test, $p = 0.05$). This answers question **Q1** affirmatively.

(Q2) Player Interaction in Online Games: As an example for a domain where sequence elements are fully relational state descriptions, we consider a massively multiplayer online game (MMOG). The game considered is *Travian*[2], a commercial, large-scale strategy game supporting approximately 20.000–30.000 interacting players. A game world consists of a large *grid map* on which players own *cities*. During the course of the game, players harvest *resources*, improve their cities by construction of *buildings* or research of *technologies*, found new cities or conquer existing cities of other players. A key aspect of Travian is player interaction in *alliances*, which are dynamically formed groups of players that act jointly. We take a high-level view of the game, focusing on alliances and player interaction only. At this level, a game state is characterized by a graph structure relating cities, players and alliances. As players are acting in the game environment, the graph structure will change, and thereby reflect changes in the social network structure of the game, see Fig. 1.

Data was collected from a "live" Travian server with approximately 25.000 active players. Over a period of three month, the state of the game world (cities and their size and position on the map, players and currently owned cities, alliance structure) was recorded once every 24 hours. As an example for high-level player cooperation, we consider the problem of identifying so-called *meta-alliances*. A meta-alliance is a group of alliances that closely cooperate. We manually identified meta-alliances in the collected game data based on the alliance names (a small free-text field). From all available data, 30 sequences of local game world states were extracted. Each sequence tracks a small set of players from three different alliances, two of which belong to the same meta-alliance (indicated by a fact $meta\_alliance(a1, a2)$). For instance, in Fig. 1 two alliances cooperate and are jointly attacked by a third alliance. On average, sequences consist of 25.8 interpretations, every interpretation contains 16.4 cities and 10.6 players, and there are 17.6 conquest events per sequence. The 30 extracted sequences constitute positive examples. A further 60 negative examples were obtained by giving the wrong meta-alliance information (i.e., $meta\_alliance(a1, a3)$ or $meta\_alliance(a2, a3)$).

We used a 10-fold cross validation to estimate BOOSTEDREAL's performance on the MMOG data. As initial distance, we chose a uniform distance measure. The average accuracy is reported in Fig. 2 and clearly shows that reward models can be learned on such complex data. Moreover the boosted reward model captured useful knowledge. To extract it, we evaluated the reward model on the data, learned a single regression tree, and converted the tree into rules. One interesting rule discovered was:

```
reward(A,B,0.55376) :-
  alliance_partners(A,C,D),
  not_alliance_partners(B,D,C),
  alliance_partners(B,D,E),
  player(A,F,G,E), player(B,H,I,D),
  conquest(A,J,H), conquest(B,K,L),
  city(B,M,H), city(B,N,F), city(A,J,F),
```

[2] www.travian.com; www.traviangames.com

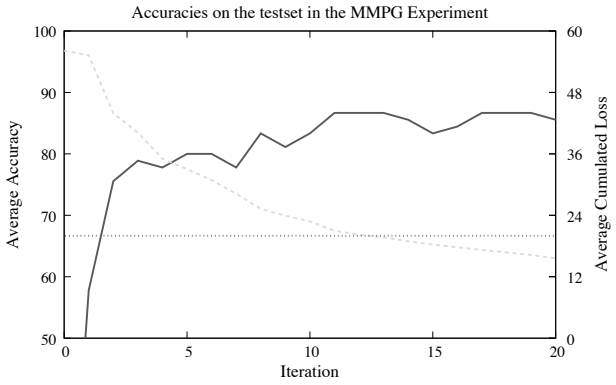

Figure 2. Results on the MMOG data. The dashed line shows the average accumulated loss of the training folds, the solid line the average accuracies on the test folds. The straight dotted line denotes the accuracy of a simple majority vote.

```
K=J, player(B,O,I,E), player(B,P,G,D),!.
```

The rule states that when aligning two sequences, the reward for two interpretations, A and B, is 0.55376, when in interpretation A two alliance partners exist, which are not partners in B but still a player from the same alliance attacks a city belonging to a player in the same alliance in B. Although this fact should be rather obvious, it shows that it is possible to extract meaningful information from the rewards models. This affirmatively answers question **Q2**.

6. Conclusions

We have introduced a novel algorithm, called BOOSTEDREAL, for relational sequence alignment based on LogitBoost and gradient-based boosting. The algorithm is straightforward to implement and highly flexible as it combines the expressive power of relational logic with sequence alignment. In contrast to previous approaches, BOOSTEDREAL can deal with any type of relational sequences even sequences of interpretations. This allows one to elegantly represent challenging dynamics over variable numbers of entities and relations such as (web) communities.

Acknowledgements KK is supported by the Fraunhofer ATTRACT fellowship STREAM and NL by the GOA/08/008 project on "Probabilistic Logic Learning". Björn Bringmann helped to collect the MMOG data.

References

[1] H. Blockeel and L. De Raedt. Top-down Induction of First-order Logical Decision Trees. *Artificial Intelligence*, 101(1–2):285–297, 1998.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[3] J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[5] B. Gutmann and K. Kersting. TildeCRF: Conditional Random Fields for Logical Sequences. In *Proc. of the 15th Europ. Conf. on Machine Learning*, 2006.

[6] T. Hubbard, A. Murzin, S. Brenner, and C. Chotia. *SCOP*: a structural classification of proteins database. *NAR*, 27(1):236–239, 1997.

[7] N. Jacobs. *Relational Sequence Learning and User Modelling*. PhD thesis, CS Dept., K.U. Leuven, 2004.

[8] T. Jiang, L. Wang, and K. Zhang. Alignment of trees: an alternative to tree edit. *TCS*, 143(1), 1995.

[9] K. Kersting, L. De Raedt, and T. Raiko. Logial Hidden Markov Models. *JAIR*, 25:425–456, 2006.

[10] K. Kersting and T. Gärtner. Fisher Kernels for Logical Sequences. In *Proc. of 15th Europ. Conf. on Machine Learning (ECML-04)*, pages 205–216, 2004.

[11] A. Ketterlin. Clustering Sequences of Complex Objects. In *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97)*, pages 215–218, 1997.

[12] N. Landwehr and L. De Raedt. r-grams: Relational grams. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, pages 907–912, 2007.

[13] S. Lee and L. De Raedt. Constraint Based Mining of First Order Sequences in SeqLog. In *Database Support for Data Mining Application*, pages 155–176. Springer, July 2004.

[14] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, 1970.

[15] S. Nienhuys-Cheng. Distance between Herbrand interpretations: A measure for approximations to a target concept. In *Proc. of the 8. Int. Conf. on Inductive Logic Programming (ILP-97)*, pages 250–260, 1997.

[16] C. Parker, A. Fern, and P. Tadepalli. Gradient Boosting for Sequence Alignment. In *Proc. of Nat. Conf. on Artificial Intelligence (AAAI-06)*, 2006.

[17] J. Ramon. *Clustering and instance based learning in first order logic*. PhD thesis, CS Dept., K.U. Leuven, 2002.

[18] K. Sato and Y.Sakakribara. RNA secondary structural alignment with conditional random field. *Bioinformatics*, 25(Suppl. 2):ii237–ii242, 2005.

[19] A. Tobudic and G. Widmer. Relational IBL in Classical Music. *Machine Learning*, 64(1-3):5–24, 2006.

[20] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. *In Proc. of the 21st Int. Conf. on Machine Learning (ICML-04)*, 2004.

[21] N. Weskamp, E. Hllermeier, D. Kuhn, and G. Klebe. Graph Alignments: A New Concept to Detect Conserved Regions in Protein Active Sites. In *Proc. German Conf. on Bioinformatics*, pages 131–140, 2004.