Regularized Least-Squares for Parse Ranking

Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg,
Aleksandr Mylläri, and Tapio Salakoski

Turku Centre for Computer Science (TUCS),
Department of Information Technology, University of Turku,
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
firstname.lastname@it.utu.fi

Abstract. We present an adaptation of the Regularized Least-Squares algorithm for the rank learning problem and an application of the method to reranking of the parses produced by the Link Grammar (LG) dependency parser. We study the use of several grammatically motivated features extracted from parses and evaluate the ranker with individual features and the combination of all features on a set of biomedical sentences annotated for syntactic dependencies. Using a parse goodness function based on the F-score, we demonstrate that our method produces a statistically significant increase in rank correlation from 0.18 to 0.42 compared to the built-in ranking heuristics of the LG parser. Further, we analyze the performance of our ranker with respect to the number of sentences and parses per sentence used for training and illustrate that the method is applicable to sparse datasets, showing improved performance with as few as 100 training sentences.

1 Introduction

Ranking, or ordinal regression, has many applications in Natural Language Processing (NLP) and has recently received significant attention in the context of parse ranking [1]. In this paper, we study parse reranking in the domain of biomedical texts. The Link Grammar (LG) parser [2] used in our research is a full dependency parser based on a broad-coverage hand-written grammar. The LG parser generates all parses allowed by its grammar and applies a set of built-in heuristics to rank the parses. However, the ranking performance of the heuristics has been found to be poor when applied to biomedical text [3]. Therefore, a primary motivation for this work is to present a machine learning approach for the parse reranking task in order to improve the applicability of the parser to the domain. We propose a method based on the Regularized Least-Squares (RLS) algorithm (see e.g. [4]), which is closely related to Support Vector Machines (SVM) (see e.g. [5]). We combine the algorithm and rank correlation measure with grammatically motivated features, which convey the most relevant information about parses.

Several applications of SVM-related machine-learning methods to ranking have been described in literature. Herbrich et al. [6] introduced SVM ordinal

A.F. Famili et al. (Eds.): IDA 2005, LNCS 3646, pp. 464–474, 2005.

regression algorithm based on a loss function between rank pairs. Joachims [7] proposed a related SVM ranking approach for optimizing the retrieval quality of search engines. SVM-based algorithms have also been applied to parse reranking, see, for example, [8]. For a recent evaluation of several parse reranking methods, see [9].

One of the aspects of the method introduced in this paper is its applicability to cases where only small amounts of data are available. The annotation of data for supervised learning is often resource-intensive, and in many domains, large annotated corpora are not available. This is especially true in the biomedical domain. In this study, we use the Biomedical Dependency Bank (BDB) dependency corpus[1] which contains 1100 annotated sentences.

The task of rank learning using the RLS-based regression method, termed here Regularized Least-Squares ranking (RLS ranking), can be applied as an machine learning approach alternative to the built-in heuristics of the LG parser. We address several aspects of parse ranking in the domain. We introduce an F-score based parse goodness function, where parses generated by the LG parser are evaluated by comparing the linkage structure to the annotated data from BDB. For evaluating ranking performance, we apply the commonly used rank correlation coefficient introduced by Kendall [10] and adopt his approach for addressing the issues of tied ranks. An application of the method to the parse rank learning task is presented, and an extensive comparison of the performance of the built-in LG parser heuristics to RLS ranking is undertaken. We demonstrate that our method produces a statistically significant increase in rank correlation from 0.18 to 0.42 compared to the built-in ranking heuristics of the LG parser.

The paper is organized as follows: in Section 2, we describe a set of grammatically motivated features for ranking dependency parses; in Section 3, we introduce a parse goodness function; in Section 4, we discuss the Regularized Least-Squares algorithm; in Section 5, we provide the performance measure applied to parse ranking and discuss the problem of tied ranks; in Section 6, we evaluate the applicability of the ranker to the task and benchmark it with respect to dataset size and the number of parses used in training; we conclude this paper in Section 7.

2 Features for Dependency Parse Ranking

The features used by a learning machine are essential to its performance, and in the problem considered in this paper, particular attention to the extracted features is required due to the sparseness of the data. We propose features that are grammatically relevant and applicable even when relatively few training examples are available. The output of the LG parser contains the following information for each input sentence: the linkage consisting of pairwise dependencies between pairs of words termed links, the link types (the grammatical roles assigned to the links) and the part-of-speech (POS) tags of the words. As LG does not perform any morphological analysis, the POS tagset used by LG is limited, consisting

[1] `http://www.it.utu.fi/~BDB`

mostly of generic verb, noun and adjective categories. Different parses of a single sentence have a different combination of these elements. Each of the features we use are described below.

Grammatical bigram. This feature is defined as a pair of words connected by a link. In the example linkage of Figure 1, the extracted grammatical bigrams are *absence—of*, *of—alpha-syntrophin*, *absence—leads*, etc. These grammatical bigrams can be considered a lower-order model related to the grammatical trigrams proposed as the basis of a probabilistic model of LG in [11]. Grammatical bigram features allow the learning machine to identify words that are commonly linked, such as *leads—to* and *binds—to*. Further, as erroneous parses are provided in training, the learning machine also has the opportunity to learn to avoid links between words that should not be linked.

Word & POS tag. This feature contains the word with the POS tag assigned to the word by LG. In the example, the extracted word & POS features are *absence.n*, *alpha-syntrophin.n*, *leads.v*, etc. Note that if LG does not assign POS to a word, no word & POS feature is extracted for that word. These features allow the ranker to learn preferences for word classes; for example, that "binds" occurs much more frequently as a verb than as a noun in the domain.

Link type. In addition to the linkage structure and POS tags, the parses contain information about the link types used to connect word pairs. The link types present in the example are *Mp*, *Js*, *Ss*, etc. The link types carry information about the grammatical structures used in the sentence and allow the ranker to learn to favor some structures over others.

Word & Link type. This feature combines each word in the sentence with the type of each link connected to the word, for example, *absence—Mp*, *absence—Ss*, *of—Js*, etc. The word & link type feature can be considered as an intermediate between grammatical unigram and bigram features, and offers a possibility for addressing potential sparseness issues of grammatical bigrams while still allowing a distinction between different linkages, unlike unigrams. This feature can also allow the ranker to learn partial selectional preferences of words, for example, that "binds" prefers to link directly to a preposition.

Link length. This feature represents the number of words that a link in the sentence spans. In Figure 1, the extracted features of this type are *1*, *1*, *3*, etc. This feature allows the ranker to learn the distinction between parses, which have different link length. The total summed link length is also used as a part of LG ordering heuristics, on the intuition that linkages with shorter link lengths are preferred [2].

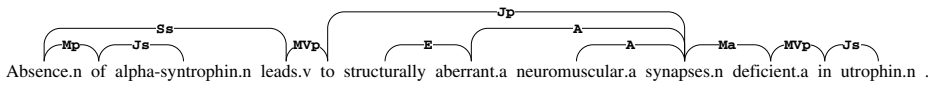

Fig. 1. Example of parsed sentence

Link length & Link type. This feature combines the type of the link in the sentence with the number of words it spans. In Figure 1, the extracted features of this type are *1—Mp*, *1—Js*, *3—Ss*, *1—MVp*, etc. The feature is also related to the total length property applied by the LG parser heuristics, which always favor linkages with shorter total link length. However, the link length & link type feature allows finer distinctions to be made by the ranker, for example, favoring short links overall but not penalizing long links to prepositions as much as other long links.

Link bigram. The link bigram features extracted for each word of the sentence are all the possible combinations of two links connected to the word, ordered left-most link first. In the example, link bigrams are *Mp—Ss*, *Mp—Js*, *Ss—MVp*, etc.

3 F-Score Based Goodness Function for Parses

The corpus BDB is a set of manually annotated sentences, that is, for each sentence of BDB, we have a manually annotated correct parse. Let P be the set of parses produced by the LG parser when applied to the sentences of BDB. We define a parse goodness function as

$$f^* : P \mapsto \mathbb{R}_+$$

which measures the similarity of the parse $p \in P$ with respect to its correct parse p^*. We propose an F-score based goodness function that assigns a goodness value to each parse based on information about the correct linkage structure. This function becomes the target output value that we try to predict with the RLS algorithm.

Let $L(p)$ denote the set of links with link types of a parse p. The functions calculating numbers of true positives (TP), false positives (FP) and false negatives (FN) links with link types are defined as follows:

$$TP(p) =\mid L(p) \cap L(p^*) \mid \tag{1}$$

$$FP(p) =\mid L(p) \setminus L(p^*) \mid \tag{2}$$

$$FN(p) =\mid L(p^*) \setminus L(p) \mid \tag{3}$$

The links are considered to be equal if and only if they have the same link type and the indices of the words connected with the links are the same in the sentence in question. We adopt one exception in (2) because of the characteristics of the corpus annotation. Namely the corpus annotation does not have all links, which the corresponding LG linkage would have: for example, punctuation is not linked in the corpus. As a consequence, links in $L(p)$ having one end connected to a token without links in $L(p^*)$, are not considered in (2). The parse goodness function is defined as an F-score

$$f^*(p) = \frac{2TP(p)}{2TP(p) + FP(p) + FN(p)}. \tag{4}$$

High values of (4) indicate that a parse contains a small number of errors, and therefore, the bigger $f^*(p)$ is, the better is parse p.

Next we consider the Regularized Least-Squares algorithm by which the measure f^* can be predicted.

4 Regularized Least-Squares Algorithm

Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in P, y_i \in \mathbb{R}$, be the set of training examples. We consider the Regularized Least-Squares (RLS) algorithm as a special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive introduction, see e.g. [4]):

$$\min_f \sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \|f\|_k^2, \tag{5}$$

where l is the loss function used by the learning machine, $f : P \to \mathbb{R}$ is a function, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_k$ is a norm in a Reproducing Kernel Hilbert Space defined by a positive definite kernel function k. Here P can be any set, but in our problem, P is a set of parses of the sentences of the BDB corpus. The target output value y_i is calculated by a parse goodness function, that is $y_i = f^*(x_i)$, and is the one which we predict with RLS algorithm. The second term in (5) is called a regularizer. The loss function used with RLS for regression problems is called least squares loss and is defined as

$$l(f(x), y) = (y - f(x))^2.$$

By the Representer Theorem (see e.g. [12]), the minimizer of equation (5) has the following form:

$$f(x) = \sum_{i=1}^{m} a_i k(x, x_i),$$

where $a_i \in \mathbb{R}$ and k is the kernel function associated with the Reproducing Kernel Hilbert Space mentioned above.

Kernel functions are similarity measures of data points in the input space P, and they correspond to the inner product in a feature space H to which the input space data points are mapped. Formally, kernel functions are defined as

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle,$$

where $\Phi : P \to H$.

5 Performance Measure for Ranking

In this section, we present the performance measures used to evaluate the parse ranking methods. We follow Kendall's definition of rank correlation coefficient [10] and measure the degree of correspondence between the true ranking and

the ranking output by an evaluated ranking method. If two rankings are equal, then correlation is +1, and on the other hand, if one ranking is the inverse of the other, correlation is −1.

The problem of the parse ranking can be formalized as follows. Let s be a sentence of BDB, and let $P_s = \{p_1, \ldots, p_n\} \subseteq P$ be the set of all parses of s produced by the LG parser. We apply the parse goodness function f^* to provide the target output variables for the parses by defining the following preference function

$$R_{f^*}(p_i, p_j) = \begin{cases} 1 & \text{if } f^*(p_i) > f^*(p_j) \\ -1 & \text{if } f^*(p_i) < f^*(p_j) \\ 0 & \text{otherwise} \end{cases}$$

which determines the ranking of the parses $p_i, p_j \in P_s$. We also define a preference function $R_f(p_i, p_j)$ in a similar way for the regression function f learned by the RLS algorithm. In order to measure how well the ranking R_f is correlated with the target ranking R_{f^*}, we adopt Kendall's commonly used rank correlation measure τ. Let us define the score S_{ij} of a pair p_i and p_j to be the product

$$S_{ij} = R_f(p_i, p_j) R_{f^*}(p_i, p_j).$$

If score is +1, then the rankings agree on the ordering of p_i and p_j, otherwise score is -1. The total score is defined as

$$S = \sum_{i<j\leq n} S_{ij}.$$

The number of all different pairwise comparisons of the parses of P_s that can be made is $\binom{n}{2} = \frac{1}{2} \cdot n\,(n-1)$. This corresponds to the maximum value of the total score, when agreement between the rankings is perfect. The correlation coefficient τ_a defined by Kendall is:

$$\tau_a = \frac{S}{\frac{1}{2} \cdot n\,(n-1)}.$$

While τ_a is well applicable in many cases, there is an important issue that is not fully addressed by this coefficient—tied ranks, that is, $f^*(p_i) = f^*(p_j)$ or $f(p_i) = f(p_j)$ for some i, j. To take into account possible occurrences of tied ranks, Kendall proposes an alternative correlation coefficient

$$\tau_b = \frac{S}{\frac{1}{2}\sqrt{\sum R_{f^*}(p_i, p_j)^2 \cdot \sum R_f(p_i, p_j)^2}}.$$

With tied ranks the usage of τ_b is more justified than τ_a. For example, if both rankings are tied except the last member, then $\tau_b = 1$ indicating complete agreement between two rankings, while $\tau_a = \frac{2}{n}$. Then for large values of n this measure is very close to 0, and therefore inappropriate. Due to many ties in the data, we use the correlation coefficient τ_b to evaluate performance of our ranker.

6 Experiments

In the experiments, BDB consisting of 1100 sentences was split into two datasets. First 500 were used for parameter estimation and feature evaluation using 10-fold cross-validation, and the rest were reserved for final validation. Each of the sentences has a variable amount of associated parses generated by LG. To address the computational complexity, we limited the number of considered parses per sentence to 5 in the training and to 20 in testing dataset. We also considered the effect of varying the number of parses per sentence used in training (Section 6.3). When more parses than the limit were available, we sampled the desired number of parses from these. When fewer were available, all parses were used.

We conducted several experiments to evaluate the performance of the method with respect to different features and learning ability of the ranker. The RLS algorithm has a regularization parameter λ which controls the tradeoff between the minimization of training errors and the complexity of the regression function. The optimal value of this parameter was determined independently by grid search in each of the experiments.

6.1 Evaluation of Features

In Section 2, we described features that were used to convey information about parses to the ranker. To measure the influence of individual feature, we conducted an experiment where features were introduced to the ranker one by one. Performance is measured using τ_b coefficient with respect to the correct ranking based on the parse goodness function f^*. As a baseline we considered the correlation between LG ranking and the correct ranking, which is 0.16. We observed that most of the features alone perform above or close to the baseline,

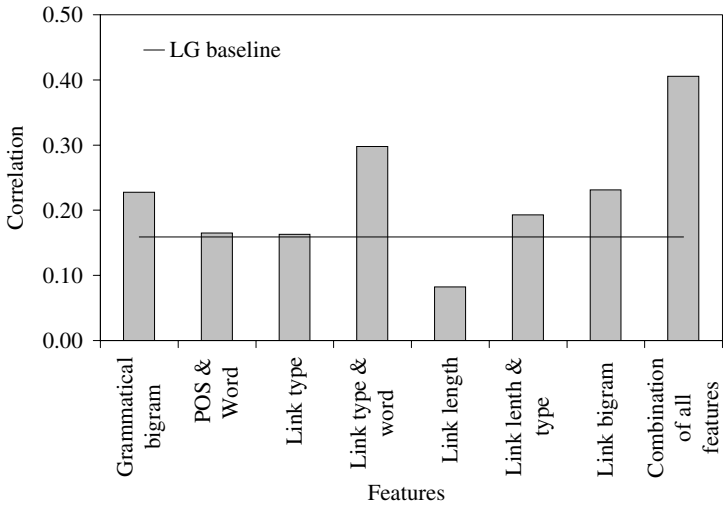

Fig. 2. RLS ranking performance with different features separately and combined

and performance of the RLS ranker with all the seven features combined on evaluation set is 0.42, supporting the relevance of proposed grammatical features. The figure 2 shows the performance of RLS ranking with respect to the different features separately and combined.

6.2 Final Validation

To test the statistical significance of the proposed method, we used the robust 5×2 cross-validation test [13]. The test avoids the problem of dependence between folds in N-fold cross-validation schemes and results in more realistic estimate than, for example, t-test. The performance of RLS ranker with all the seven features combined on validation set is 0.42 and the improvement is statistically significant ($p < 0.01$) when compared to 0.18 obtained by the LG parser. We also measured the performance of our ranker with respect to the parse ranked first. The average F-score of the true best parse in the corpus is 73.2%. The average F-score value obtained by the parses ranked first by the RLS ranker was found to be 67.6%, and the corresponding value for the LG parser heuristics was 64.2%, showing therefore better performance of our method also for this measure. Note that average F-scores of the highest ranked parses were obtained from the downsampled 20 parses per sentence.

6.3 Learning Ability of the Ranker with Respect to the Training Data

To address the issue of applicability of the proposed method to very sparse datasets, we measured performance of the RLS ranker with respect to two main criteria: the number of sentences and the number of parses per sentence used for training. In these experiments all grammatical features were used.

Number of Sentences. The training dataset of 500 sentences was divided into several parts and for testing a separate set of 500 sentences was used. The validation procedure was applied for each of the parts, representing sets of sizes 50, 100,..., 500 sentences. The number of parses used per sentence for training was 5 and for testing 20. We observed that even with a very sparse dataset our method gives a relatively good performance of 0.37 while the learning set size remains as small as 100 sentences. The learning procedure reflected expected tendency of the increased ranker performance with increased number of sentences, reaching 0.42 with 500 sentences.

Number of Parses. We measured performance of the RLS ranker based on the number of parses per sentence used for training with dataset size fixed to 150 sentences. Number of parses per sentence in training was selected to be 5, 10,..., 50 for each validation run. Test dataset consisted of 500 sentences each containing 20 parses. We observed that major improvement in ranker performance occurs while using only 10 or 20 parses per sentence for training corresponding to 0.41 and 0.43 performance respectively. When using 50 parses per sentence,

performance was 0.45, indicating a small positive difference compared to results obtained with less number of parses.

Parse-Sentence Tradeoff. In this experiment, we fixed the number of training examples, representing number of sentences multiplied by number of available parses per sentence, to be approximately one thousand. Datasets of 20, 30, 50, 100, 200, 300, 500 sentences with number of parses per sentence 50, 30, 20, 10, 5, 3, 2, respectively, were validated with 500 sentences each containing 20 parses. The results of these experiments are presented in Table 1. We found that the best performance of ranker was achieved using 100 sentences and 10 parses per each sentence for training, corresponding to 0.41 correlation. The decrease in performance was observed when either having large number of parses with small amount of sentences or vice versa.

Table 1. Ranking performance with different number of sentences and parses

# Sentences	*# Parses*	*Correlation*	*Difference in correlation*
20	50	0.3303	0.0841
30	30	0.3529	0.0615
50	20	0.3788	0.0357
100	***10***	***0.4145***	***0.0000***
200	5	0.3798	0.0347
300	3	0.3809	0.0335
500	2	0.3659	0.0485

7 Discussion and Conclusions

In this study, we proposed a method for parse ranking based on Regularized Least-Squares algorithm coupled with rank correlation measure and grammatically motivated features. We introduce an F-score based parse goodness function. To convey the most important information about parse structure to the ranker, we apply features such as grammatical bigrams, link types, a combination of link length and link type, part-of-speech information, and others. When evaluating the ranker with respect to each feature separately and all features combined, we observed that most of them let the ranker to outperform Link Grammar parser built-in heuristics. For example, grammatical bigram (pair of words connected by a link) and link bigram (pair of links related by words) underline importance of link dependency structure for ranking. Another feature yielding good performance is link type & word, representing an alternative grammatical structure and providing additional information in case of similar parses. We observed that link length feature, which is related to LG heuristics, leads to poor, below the baseline, performance, whereas other features appear to have more positive effect.

We performed several experiments to estimate learning abilities of the ranker, and demonstrate that the method is applicable for sparse datasets. A tradeoff spot between number of parses and sentences used for training demonstrates

that maximum performance is obtained at 100 sentences and 10 parses per sentence supporting our claim for applicability of the ranker to small datasets. Experimental results suggest that for practical reasons the use of 10 to 20 parses per sentence for training is sufficient. We compared RLS ranking to the built-in heuristics of LG parser and a statistically significant improvement in performance from 0.18 to 0.42 was observed.

In the future, we plan to address the issue of RLS algorithm adaptation for ranking by applying and developing kernel functions, which would use domain knowledge about parse structure. Several preliminary experiments with multiple output regression seemed promising and are worth exploring in more detail. In addition, we plan to incorporate RLS ranking into the LG parser as an alternative ranking possibility to its built-in heuristics.

Acknowledgments

This work has been supported by Tekes, the Finnish National Technology Agency and we also thank CSC, the Finnish IT center for science for providing us extensive computing resources.

References

1. Collins, M.: Discriminative reranking for natural language parsing. In Langley, P., ed.: Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, Morgan Kaufmann (2000) 175–182
2. Sleator, D.D., Temperley, D.: Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1991)
3. Pyysalo, S., Ginter, F., Pahikkala, T., Boberg, J., Järvinen, J., Salakoski, T., Koivula, J.: Analysis of link grammar on biomedical dependency corpus targeted at protein-protein interactions. In Collier, N., Ruch, P., Nazarenko, A., eds.: Proceedings of the JNLPBA workshop at COLING'04, Geneva. (2004) 15–21
4. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Amer. Math. Soc. Notice **50** (2003) 537–544
5. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc. (1995)
6. Herbrich, R., Graepel, T., Obermayer, K.: Support vector learning for ordinal regression. In: Proceedings of the Ninth International Conference on Artificial Neural Networks, London, UK, IEE (1999) 97–102
7. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM Press (2002) 133–142
8. Shen, L., Joshi, A.K.: An svm-based voting algorithm with application to parse reranking. In Daelemans, W., Osborne, M., eds.: Proceedings of CoNLL-2003. (2003) 9–16
9. Collins, M., Koo, T.: Discriminative reranking for natural language parsing (2004) To appear in Computational Linguistics, available at http://people.csail.mit.edu/people/mcollins/papers/collinskoo.ps.

10. Kendall, M.G.: Rank Correlation Methods. 4. edn. Griffin, London (1970)
11. Lafferty, J., Sleator, D., Temperley, D.: Grammatical trigrams: A probabilistic model of link grammar. In: Proceedings of the AAAI Conference on Probabilistic Approaches to Natural Language, Menlo Park, CA, AAAI Press (1992) 89–97
12. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In Helmbold, D., Williamson, R., eds.: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory, Berlin, Germany, Springer-Verlag (2001) 416–426
13. Alpaydin, E.: Combined 5×2 cv F-test for comparing supervised classification learning algorithms. Neural Computation **11** (1999) 1885–1892