A Probabilistic Search for the Best Solution Among Partially Completed Candidates

Filip Ginter, Aleksandr Mylläri, and Tapio Salakoski
Turku Centre for Computer Science (TUCS) and
Department of Information Technology
University of Turku
Lemminkäisenkatu 14 A
20520 Turku, Finland
first.last@it.utu.fi

Abstract

We consider the problem of identifying among many candidates a single best solution which jointly maximizes several domain-specific target functions. Assuming that the candidate solutions can be generated incrementally, we model the error in prediction due to the incompleteness of partial solutions as a normally distributed random variable. Using this model, we derive a probabilistic search algorithm that aims at finding the best solution without the necessity to complete and rank all candidate solutions. We do not assume a Viterbi-type decoding, allowing a wider range of target functions.

We evaluate the proposed algorithm on the problem of best parse identification, combining simple heuristic with more complex machine-learning based target functions. We show that the search algorithm is capable of identifying candidates with a very high score without completing a significant proportion of the candidate solutions.

1 Background

Most of the current NLP systems assume a pipeline architecture, where each level of analysis is implemented as a module that produces a single, locally optimal solution that is passed to the next module in the pipeline. There has recently been an increased interest in the application of joint inference, which identifies a solution that is globally optimal throughout the system and avoids some of the problems of the pipeline architecture, such as error propagation.

We assume, at least conceptually, a division of the joint inference problem into two subproblems: that of finding a set of solutions that are structurally compatible with each of the modules, and that of selecting the globally best of these structurally correct solutions. Many of the modules define a target function that scores the solutions by some domain criteria based on local knowledge. The globally best solution maximizes some combination of the target functions, for example a sum.

For illustration, consider a system comprising of two modules: a POS tagger and a parser. The POS tagger generates a set of tag sequences that are compatible with the sentence text. Further, it may implement a target function, based, for instance, on tag n-grams, that scores these sequences according to POS-centric criteria. The parser produces a set of candidate parses and typically also implements a target function that scores the parses based on their structural and lexical features. Each parse that is compatible with both the POS tagger and the parser is structurally correct. The best solution may be defined, for instance, as such a solution that maximizes the sum of the scores of the POS- and parser-centric target functions.

In practice, the set of structurally correct solutions may be computed, for example, through the intersection or composition of finite-state automata as in the formalism of finite-state intersection grammars (Koskenniemi, 1990). Finding the best so-

Workshop on Computationally Hard Problemsand Joint Inference in Speech and Language Processing, pages 33–40,
New York City, New York, June 2006.

lution may be implemented as a best-path search through Viterbi decoding, given a target function that satisfies the Viterbi condition.

Most of the recent approaches to NLP tasks like parse re-ranking make, however, use of feature-based representations and machine-learning induced target functions, which do not allow efficient search strategies that are guaranteed to find the global optimum. In general case, all structurally correct solutions have to be generated and scored by the target functions in order to guarantee that the globally optimal solution is found. Further, each of the various problems in natural language processing is typically approached with a different class of models, ranging from n-gram statistics to complex regressors and classifiers such as the support vector machines. These different approaches need to be combined in order to find the globally optimal solution. Therefore, in our study we aim to develop a search strategy that allows to combine a wider range of target functions.

An alternative approach is that of propagating n best solutions through the pipeline system, where each step re-ranks the solutions by local criteria (Ji et al., 2005). Incorporating a wide range of features representing information from all levels of analysis into a single master classifier is other commonly used method (Kambhatla, 2004; Zelenko et al., 2004).

In this paper, we assume the possibility of generating the structurally correct solutions incrementally, through a sequence of partially completed solutions. We then derive a probabilistic search algorithm that attempts to identify the globally best solution, without fully completing all structurally correct solutions. Further, we do not impose strong restrictions, such as the Viterbi assumption, on the target functions.

To a certain extent, this approach is related to the problem of cost-sensitive learning, where obtaining a feature value is associated with a cost and the objective is to minimize the cost of training data acquisition and the cost of instance classification (Melville et al., 2004). However, the crucial difference is that we do not assume the possibility to influence when advancing a partial solution, which feature will be obtained next.

2 Method

Let us consider a system in which there are N solutions $s_1, \ldots, s_N \in \mathcal{S}$ to a problem and M target functions $f_1, \ldots, f_M$, where $f_k : \mathcal{S} \rightarrow \mathbb{R}$, that assign a score to each of the solutions. The score $f_k(s_i)$ expresses the extent to which the solution s_i satisfies the criterion implemented by the target function f_k. The overall score of a solution s_i

$$f(s_i) = \sum_{k=1}^{M} f_k(s_i) \tag{1}$$

is the sum of the scores given by the individual target functions. The objective is to identify $\hat{s}$, the best among the N possible solutions, that maximizes the overall score:

$$\hat{s} = \arg\max_{s_i} f(s_i) \ . \tag{2}$$

Suppose that the solutions are generated incrementally so that each solution s_i can be reached through a sequence of F *partial solutions* $s_{i,1}, s_{i,2}, \ldots, s_{i,F}$, where $s_{i,F} = s_i$. Let further $u : \mathcal{S} \rightarrow (0, 1]$ be a measure of a *degree of completion* for a particular solution. For a complete solution s_i, $u(s_i) = 1$, and for a partial solution $s_{i,n}$, $u(s_i) < 1$. For instance, when assigning POS tags to the words of a sentence, the degree of completion could be defined as the number of words assigned with a POS tag so far, divided by the total number of words in the sentence.

The score of a partial solution $s_{i,n}$ is, to a certain extent, a prediction of the score of the corresponding complete solution s_i. Intuitively, the accuracy of this prediction depends on the degree of completion. The score of a partial solution with a high degree of completion is generally closer to the final score, compared to a partial solution with a low degree of completion.

Let

$$\delta_k(s_{i,n}) = f_k(s_i) - f_k(s_{i,n}) \tag{3}$$

be the difference between the scores of s_i and $s_{i,n}$. That is, $\delta_k(s_{i,n})$ is the error in score caused by the incompleteness of the partial solution $s_{i,n}$. As the solutions are generated incrementally, the exact value of $\delta_k(s_{i,n})$ is not known at the moment of generating $s_{i,n}$ because the solution s_i has not been completed

yet. However, we can model the error based on the knowledge of $s_{i,n}$. We assume that, for a given $s_{i,n}$, the error $\delta_k(s_{i,n})$ is a random variable distributed according to a probability distribution with a density function Δ_k, denoted as

$$\delta_k(s_{i,n}) \sim \Delta_k(\delta; s_{i,n}) \ . \tag{4}$$

The partial solution $s_{i,n}$ is a parameter to the distribution and, in theory, each partial solution gives rise to a different distribution of the same general shape.

We assume that the error $\delta(s_{i,n})$ is distributed around a mean value and for a 'reasonably behaving' target function, the probability of a small error is higher than the probability of a large error. Ideally, the target function will not exhibit any systematic error, and the mean value would thus be zero[1]. For instance, a positive mean error indicates a systematic bias toward underestimating the score. The mean error should approach 0 as the degree of completion increases and the error of a complete solution is always 0. We have further argued that the reliability of the prediction grows with the degree of completion. That is, the error of a partial solution with a high degree of completion should exhibit a smaller variance, compared to that of a largely incomplete solution. The variance of the error for a complete solution is always 0.

Knowing the distribution Δ_k of the error δ_k, the density of the distribution $d_k(f; s_{i,n})$ of the final score $f_k(s_i)$ is obtained by shifting the density of the error $\delta_k(s_{i,n})$ by $f_k(s_{i,n})$, that is,

$$f_k(s_i) \sim d_k(f; s_{i,n}) \ , \tag{5}$$

where

$$d_k(f; s_{i,n}) = \Delta_k(f - f_k(s_{i,n}) \, ; s_{i,n}) \ . \tag{6}$$

So far, we have discussed the case of a single target function f_k. Let us now consider the general case of M target functions. Knowing the final score density d_k for the individual target functions f_k, it is now necessary to find the density of the overall score $f(s_i)$. By Equation 1, it is distributed as the sum

[1]We will see in our evaluation experiments that this is not the case, and the target functions may exhibit a systematic bias in the error δ.

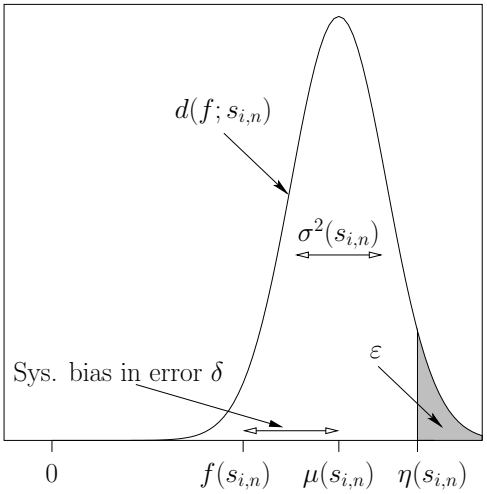

Figure 1: The probability density $d(f; s_{i,n})$ of the distribution of the final score $f(s_i)$, given a partial solution $s_{i,n}$. The density is assumed normally distributed, with mean $\mu(s_{i,n})$ and variance $\sigma^2(s_{i,n})$. With probability $1 - \varepsilon$, the final score is less than $\eta(s_{i,n})$.

of the random variables $f_1(s_i), \ldots, f_M(s_i)$. Therefore, assuming independence, its density is the convolution of the densities of these variables, that is, given $s_{i,n}$,

$$d(f; s_{i,n}) = (d_1 * \ldots * d_M)(f; s_{i,n}) \ , \tag{7}$$

and

$$f(s_i) \sim d(f; s_{i,n}) \ . \tag{8}$$

We have assumed the independence of the target function scores. Further, we will make the assumption that d takes the form of the normal distribution, which is convolution-closed, a property necessary for efficient calculation by Equation 7. We thus have

$$d(f; s_{i,n}) = n\big(f; \mu(s_{i,n}), \sigma^2(s_{i,n})\big) \ , \tag{9}$$

where n is the normal density function. While it is unlikely that independence and normality hold strictly, it is a commonly used approximation, necessary for an analytical solution of (7). The notions introduced so far are illustrated in Figure 1.

2.1 The search algorithm

We will now apply the model introduced in the previous section to derive a probabilistic search algorithm.

Let us consider two partial solutions $s_{i,n}$ and $s_{j,m}$ with the objective of deciding which one of them is 'more promising', that is, more likely to lead to a complete solution with a higher score. The condition of 'more promising' can be defined in several ways. For instance, once again assuming independence, it is possible to directly compute the probability $P(f(s_i) < f(s_j))$:

$$\begin{aligned} &P(f(s_i) < f(s_j)) \\ &= P(f(s_i) - f(s_j) < 0) \\ &= \int_{-\infty}^{0} (d_{s_{i,n}} * (-d_{s_{j,m}}))(f)\, df\,, \end{aligned} \tag{10}$$

where $d_{s_{i,n}}$ refers to the function $d(f; s_{i,n})$. Since d is the convolution-closed normal density, Equation 10 can be directly computed using the normal cumulative distribution. The disadvantage of this definition is that the cumulative distribution needs to be evaluated separately for each pair of partial solutions. Therefore, we assume an alternate definition of 'more promising' in which the cumulative distribution is evaluated only once for each partial solution.

Let $\varepsilon \in [0, 1]$ be a probability value and $\eta(s_{i,n})$ be the score such that $P(f(s_i) > \eta(s_{i,n})) = \varepsilon$. The value of $\eta(s_{i,n})$ can easily be computed from the inverse cumulative distribution function corresponding to the density function $d(f; s_{i,n})$. The interpretation of $\eta(s_{i,n})$ is that with probability of $1 - \varepsilon$, the partial solution $s_{i,n}$, once completed, will lead to a score smaller than $\eta(s_{i,n})$. The constant ε is a parameter, set to an appropriate small value. See Figure 1 for illustration.

We will refer to $\eta(s_{i,n})$ as the *maximal expected score* of $s_{i,n}$. Of the two partial solutions, we consider as 'more promising' the one, whose maximal expected score is higher. As illustrated in Figure 2, it is possible for a partial solution $s_{i,n}$ to be more promising even though its score $f(s_{i,n})$ is lower than that of some other partial solution $s_{j,m}$.

Further, given a complete solution s_i and a partial solution $s_{j,m}$, a related question is whether $s_{j,m}$ is a *promising solution*, that is, whether it is likely that advancing it will lead to a score higher than $f(s_i)$. Using the notion of maximal expected score, we say that a solution is promising if $\eta(s_{j,m}) > f(s_i)$.

With the definitions introduced so far, we are

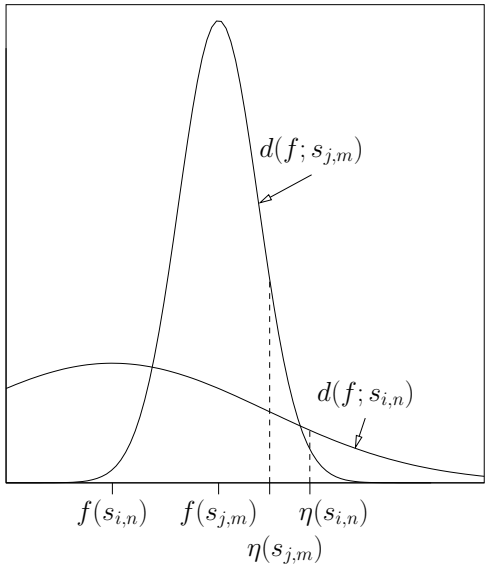

Figure 2: Although the score of $s_{i,n}$ is lower than the score of $s_{j,m}$, the partial solution $s_{i,n}$ is more promising, since $\eta(s_{i,n}) > \eta(s_{j,m})$. Note that for the sake of simplicity, a zero systematic bias of the error δ is assumed, that is, the densities are centered around the partial solution scores.

now able to perform two basic operations: compare two partial solutions, deciding which one of them is more promising, and compare a partial solution with some complete solution, deciding whether the partial solution is still promising or can be disregarded. These two basic operations are sufficient to devise the following search algorithm.

- Maintain a priority queue of partial solutions, ordered by their maximal expected score.
- In each step, remove from the queue the partial solution with the highest maximal expected score, advance it, and enqueue any resulting partial solutions.
- Iterate while the maximal expected score of the most promising partial solution remains higher than the score of the best complete solution discovered so far.

The parameter ε primarily affects how early the algorithm stops, however, it influences the order in which the solutions are considered as well. Low values of ε result in higher maximal expected scores

and therefore partial solutions need to be advanced to a higher degree of completion before they can be disregarded as unpromising.

While there are no particular theoretical restrictions on the target functions, there is an important practical consideration. Since the target function is evaluated every time a partial solution $s_{i,n}$ is advanced into $s_{i,n+1}$, being able to use the information about $s_{i,n}$ to efficiently compute $f_k(s_{i,n+1})$ is necessary.

The algorithm is to a large extent related to the A* search algorithm, which maintains a priority queue of partial solutions, ordered according to a score $g(x) + h(x)$, where $g(x)$ is the score of x and $h(x)$ is a heuristic overestimate[2] of the final score of the goal reached from x. Here, the maximal expected score of a partial solution is an overestimate with the probability of $1 - \varepsilon$ and can be viewed as a probabilistic counterpart of the A* heuristic component $h(x)$. Note that A* only guarantees to find the best solution if $h(x)$ never underestimates, which is not the case here.

2.2 Estimation of $\mu_{\mathbf{k}}(\mathbf{s_{i,n}})$ and $\sigma^{\mathbf{2}}_{\mathbf{k}}(\mathbf{s_{i,n}})$

So far, we have assumed that for each partial solution $s_{i,n}$ and each target function f_k, the density $\Delta_k(\delta; s_{i,n})$ is defined as a normal density specified by the mean $\mu_k(s_{i,n})$ and variance $\sigma^2_k(s_{i,n})$. This density models the error $\delta_k(s_{i,n})$ that arises due to the incompleteness of $s_{i,n}$. The parameters $\mu_k(s_{i,n})$ and $\sigma^2_k(s_{i,n})$ are, in theory, different for each $s_{i,n}$ and reflect the behavior of the target function f_k as well as the degree of completion and possibly other attributes of $s_{i,n}$. It is thus necessary to estimate these two parameters from data.

Let us, for each target function f_k, consider a training set of observations $\mathcal{T}_k \subset \mathcal{S} \times \mathbb{R}$. Each training observation $t_j = \big(s_{j,n_j}, \delta_k\big(s_{j,n_j}\big)\big) \in \mathcal{T}_k$ corresponds to a solution s_{j,n_j} with a known error $\delta_k\big(s_{j,n_j}\big) = f_k(s_j) - f_k\big(s_{j,n_j}\big)$.

Before we introduce the method to estimate the density $\Delta_k(\delta; s_{i,n})$ for a particular $s_{i,n}$, we discuss data normalization. The overall score $f(s_{i,n})$ is defined as the sum of the scores assigned by the individual target functions f_k. Naturally, it is desirable that these scores are of comparable magnitudes. Therefore, we normalize the target functions using the z-normalization

$$z(x) = \frac{x - mean(x)}{stdev(x)}\,. \tag{11}$$

Each target function f_k is normalized separately, based on the data in the training set $\mathcal{T}_k$. Throughout our experiments, the values of the target function are always z-normalized.

Let us now consider the estimation of the mean $\mu_k(s_{i,n})$ and variance $\sigma^2_k(s_{i,n})$ that define the density $\Delta_k(\delta; s_{i,n})$. Naturally, it is not possible to estimate the distribution parameters for each solution $s_{i,n}$ separately. Instead, we approximate the parameters based on two most salient characteristics of each solution: the degree of completion $u(s_{i,n})$ and the score $f_k(s_{i,n})$. Thus,

$$\mu_k(s_{i,n}) \approx \mu_k(u(s_{i,n})\,, f_k(s_{i,n})) \tag{12}$$

$$\sigma^2_k(s_{i,n}) \approx \sigma^2_k(u(s_{i,n})\,, f_k(s_{i,n}))\,. \tag{13}$$

Let us assume the following notation: $u_i = u(s_{i,n})$, $f_i = f_k(s_{i,n})$, $u_j = u\big(s_{j,n_j}\big)$, $f_j = f_k\big(s_{j,n_j}\big)$, and $\delta_j = \delta_k\big(s_{j,n_j}\big)$. The estimate is obtained from $\mathcal{T}_k$ using kernel smoothing (Silverman, 1986):

$$\mu_k(u_i, f_i) = \frac{\sum_{t_j \in \mathcal{T}} \delta_j K}{\sum_{t_j \in \mathcal{T}} K} \tag{14}$$

and

$$\sigma^2_k(u_i, f_i) = \frac{\sum_{t_j \in \mathcal{T}} (\delta_j - \mu_k(u_i, f_i))^2\, K}{\sum_{t_j \in \mathcal{T}} K}\,, \tag{15}$$

where K stands for the kernel value $K_{u_i,f_i}(u_j, f_j)$. The kernel K is the product of two Gaussians, centered at u_i and f_i, respectively.

$$K_{u_i,f_i}(u_j, f_j) = n\big(u_j; u_i, \sigma^2_u\big) \cdot n\big(f_j; f_i, \sigma^2_f\big)\,, \tag{16}$$

where $n\big(x; \mu, \sigma^2\big)$ is the normal density function. The variances σ^2_u and σ^2_f control the degree of smoothing along the u and f axes, respectively. High variance results in stronger smoothing, compared to low variance. In our evaluation, we set the

[2] In the usual application of A* to shortest-path search, $h(x)$ is a heuristic underestimate since the objective is to minimize the score.

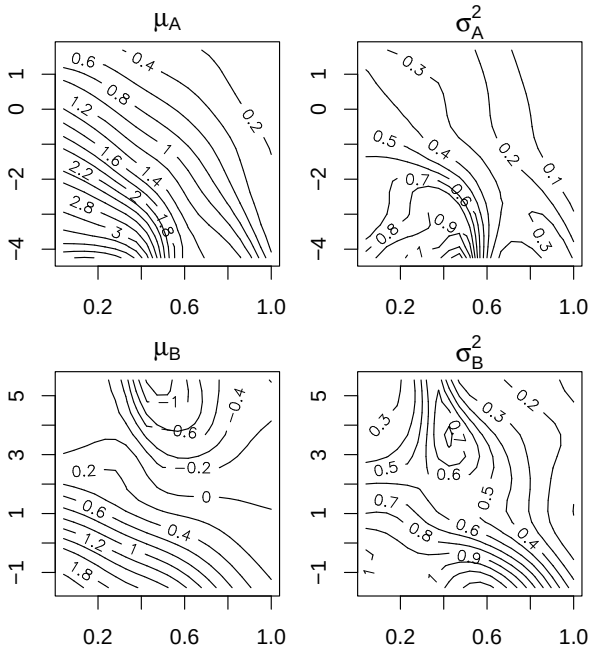

Figure 3: Mean and variance of the error $\delta(s_{i,n})$. By (12) and (13), the error is approximated as a function of the degree of completion $u(s_{i,n})$ and the score $f_k(s_{i,n})$. The degree of completion is on the horizontal and the score on the vertical axis. The estimates (μ_A, σ_A^2) and (μ_B, σ_B^2) correspond to the RLSC regressor and average link length target functions, respectively.

variance such that σ_u and σ_f equal to 10% of the distance from $\min(u_j)$ to $\max(u_j)$ and from $\min(f_j)$ to $\max(f_j)$, respectively.

The kernel-smoothed estimates of μ and σ^2 for two of the four target functions used in the evaluation experiments are illustrated in Figure 3. While both estimates demonstrate the decrease both in mean and variance for u approaching 0, the target functions generally exhibit a different behavior. Note that the values are clearly dependent on both the score and the degree of completion, indicating that the degree of completion alone is not sufficiently representative of the partial solutions. Ideally, the values of both the mean and variance should be strictly 0 for $u = 1$, however, due to the effect of smoothing, they remain non-zero.

3 Evaluation

We test the proposed search algorithm on the problem of dependency parsing. We have previously developed a finite-state implementation (Ginter et al., 2006) of the Link Grammar (LG) parser (Sleator and Temperley, 1991) which generates the parse through the intersection of several finite-state automata. The resulting automaton encodes all candidate parses. The parses are then generated from left to right, proceeding through the automaton from the initial to the final state. A partial parse is a sequence of n words from the beginning of the sentence, together with string encoding of their dependencies. Advancing a partial parse corresponds to appending to it the next word. The degree of completion is then defined as the number of words currently generated in the parse, divided by the total number of words in the sentence.

To evaluate the ability of the proposed method to combine diverse criteria in the search, we use four target functions: a complex state-of-the-art parse reranker based on a regularized least-squares (RLSC) regressor (Tsivtsivadze et al., 2005), and three measures inspired by the simple heuristics applied by the LG parser. The criteria are the average length of a dependency, the average level of nesting of a dependency, and the average number of dependencies linking a word. The RLSC regressor, on the other hand, employs complex features and word n-gram statistics.

The dataset consists of 200 sentences randomly selected from the BioInfer corpus of dependency-parsed sentences extracted from abstracts of biomedical research articles (Pyysalo et al., 2006). For each sentence, we have randomly selected a maximum of 100 parses. For sentences with less than 100 parses, all parses were selected. The average number of parses per sentence is 62. Further, we perform 5×2 cross-validation, that is, in each of five replications, we divide the data randomly to two sets of 100 sentences and use one set to estimate the probability distributions and the other set to measure the performance of the search algorithm. The RLSC regressor is trained once, using a different set of sentences from the BioInfer corpus. The results presented here are averaged over the 10 folds. As a comparative baseline, we use a simple

greedy search algorithm that always advances the partial solution with the highest score until all solutions have been generated.

3.1 Results

For each sentence s with parses $\mathcal{S}\,\{s_1, \ldots, s_N\}$, let $\mathcal{S}_C \subseteq \mathcal{S}$ be the subset of parses fully completed before the algorithm stops and $\mathcal{S}_N = \mathcal{S} \setminus \mathcal{S}_C$ the subset of parses not fully completed. Let further T_C be the number of iterations taken before the algorithm stops, and T be the total number of steps needed to generate all parses in $\mathcal{S}$. Thus, $|\mathcal{S}|$ is the size of the search space measured in the number of parses, and T is the size of the search space measured in the number of steps. For a single parse s_i, $rank(s_i)$ is the number of parses in $\mathcal{S}$ with a score higher than $f(s_i)$ plus 1. Thus, the rank of all solutions with the maximal score is 1. Finally, $ord(s_i)$ corresponds to the order in which the parses were completed by the algorithm (disregarding the stopping criterion). For example, if the parses were completed in the order s_3, s_8, s_1, then $ord(s_3) = 1$, $ord(s_8) = 2$, and $ord(s_1) = 3$. While two solutions have the same rank if their scores are equal, no two solutions have the same order. The best completed solution $\hat{s}_C \in \mathcal{S}_C$ is the solution with the highest rank in $\mathcal{S}_C$ and the lowest order among solutions with the same rank. The best solution $\hat{s}$ is the solution with rank 1 and the lowest order among solutions with rank 1. If $\hat{s} \in \mathcal{S}_C$, then $\hat{s}_C = \hat{s}$ and the objective of the algorithm to find the best solution was fulfilled. We use the following measures of performance: $rank(\hat{s}_C)$, $ord(\hat{s})$, $\frac{|\mathcal{S}_C|}{|\mathcal{S}|}$, and $\frac{T_C}{T}$. The most important criteria are $rank(\hat{s}_C)$ which measures how good the best found solution is, and $\frac{T_C}{T}$ which measures the proportion of steps actually taken by the algorithm of the total number of steps needed to complete all the candidate solutions. Further, $ord(\hat{s})$, the number of parses completed before the global optimum was reached regardless the stopping criterion, is indicative about the ability of the search to reach the global optimum early among the completed parses. Note that all measures except for $ord(\hat{s})$ equal to 1 for the baseline greedy search, since it lacks a stopping criterion.

The average performance values for four settings of the parameter ε are presented in Table 1. Clearly,

ε	$rank(\hat{s}_C)$	$ord(\hat{s})$	$\frac{\lvert\mathcal{S}_C\rvert}{\lvert\mathcal{S}\rvert}$	$\frac{T_C}{T}$
0.01	1.6	8.8	0.78	0.94
0.05	2.8	11.2	0.62	0.85
0.10	4.0	12.2	0.53	0.79
0.20	6.0	13.5	0.41	0.73
Base	1.0	28.7	1.00	1.00

Table 1: Average results over all sentences.

the algorithm behaves as expected with respect to the parameter ε. While with the strictest setting $\varepsilon = 0.01$, 94% of the search space is explored, with the least strict setting of $\varepsilon = 0.2$, 73% is explored, thus pruning one quarter of the search space. The proportion of completed parses is generally considerably lower than the proportion of explored search space. This indicates that the parses are generally advanced to a significant level of completion, but then ruled out. The behavior of the algorithm is thus closer to a breadth-first, rather than depth-first search. We also notice that the average rank of the best completed solution is very low, indicating that although the algorithm does not necessarily identify the best solution, it generally identifies a very good solution. In addition, the order of the best solution is low as well, suggesting that generally good solutions are identified before low-score solutions. Further, compared to the baseline, the globally optimal solution is reached earlier among the completed parses, although this does not imply that it is reached earlier in the number of steps. Apart from the overall averages, we also consider the performance with respect to the number of alternative parses for each sentence (Table 2). Here we see that even with the least strict setting, the search finds a reasonably good solution while being able to reduce the search space to 48%.

4 Conclusions and future work

We have considered the problem of identifying a globally optimal solution among a set of candidate solutions, jointly optimizing several target functions that implement domain criteria. Assuming the solutions are generated incrementally, we have derived a probabilistic search algorithm that aims to identify the globally optimal solution without completing all of the candidate solutions. The algorithm is based on a model of the error in prediction caused by the in-

		$\varepsilon = 0.01$				$\varepsilon = 0.2$				Base
$\lvert\mathcal{S}\rvert$	#	$rank(\hat{s}_C)$	$ord(\hat{s})$	$\frac{\lvert\mathcal{S}_C\rvert}{\lvert\mathcal{S}\rvert}$	$\frac{T_C}{T}$	$rank(\hat{s}_C)$	$ord(\hat{s})$	$\frac{\lvert\mathcal{S}_C\rvert}{\lvert\mathcal{S}\rvert}$	$\frac{T_C}{T}$	$ord(\hat{s})$
1-10	40	1.0	1.6	1.00	1.00	1.2	1.8	0.84	0.95	2.85
11-20	18	1.1	4.4	0.88	0.97	2.8	7.0	0.54	0.79	9.82
21-30	8	1.0	2.9	1.00	1.00	1.0	2.4	0.80	0.98	14.75
31-40	9	1.2	7.8	0.79	0.95	2.6	10.8	0.48	0.74	20.67
41-50	6	1.0	4.4	0.80	0.89	4.9	9.8	0.28	0.61	18.07
51-60	3	1.0	2.3	0.64	0.88	7.1	5.9	0.30	0.59	38.67
61-70	5	1.1	26.9	0.86	0.99	3.4	23.2	0.22	0.68	32.60
71-80	3	1.0	8.7	0.78	0.98	9.2	19.6	0.30	0.71	49.67
81-90	6	2.5	8.2	0.61	0.94	9.3	16.6	0.24	0.76	47.67
91-100	102	5.2	20.9	0.50	0.81	18.9	38.2	0.15	0.48	52.69

Table 2: Average results with respect to the number of alternative parses. The column # contains the number of sentences in the dataset which have the given number of parses.

completeness of a partial solution. Using the model, the order in which partial solutions are explored is defined, as well as a stopping criterion for the algorithm.

We have performed an evaluation using best parse identification as the model problem. The results indicate that the method is capable of combining simple heuristic criteria with a complex regressor, identifying solutions with a very low average rank.

The crucial component of the method is the model of the error δ. Improving the accuracy of the model may potentially further improve the performance of the algorithm, allowing a more accurate stopping criterion and better order in which the parses are completed. We have assumed independence between the scores assigned by the target functions. As a future work, a multivariate model will be considered that takes into account the mutual dependencies of the target functions.

References

Filip Ginter, Sampo Pyysalo, Jorma Boberg, and Tapio Salakoski. 2006. Regular approximation of Link Grammar. Manuscript under review.

Heng Ji, David Westbrook, and Ralph Grishman. 2005. Using semantic relations to refi ne coreference decisions. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP'05), Vancouver, Canada*, pages 17–24. ACL.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *The Companion Volume to the Proceedings of 42st Annual Meeting of the Association for Computational Linguistics (ACL'04), Barcelona, Spain*, pages 178–181. ACL.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90), Helsinki, Finland*, pages 229–232. ACL.

Prem Melville, Maytal Saar-Tsechansky, Foster Provost, and Raymond Mooney. 2004. Active feature-value acquisition for classifi er induction. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 483–486. IEEE Computer Society.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2006. Bio Information Extraction Resource: A corpus for information extraction in the biomedical domain. Manuscript under review.

Bernard W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall.

Daniel D. Sleator and Davy Temperley. 1991. Parsing English with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Evgeni Tsivtsivadze, Tapio Pahikkala, Sampo Pyysalo, Jorma Boberg, Aleksandr Myllär i, and Tapio Salakoski. 2005. Regularized least-squares for parse ranking. In *Proceedings of the 6th International Symposium on Intelligent Data Analysis (IDA'05), Madrid, Spain*, pages 464–474. Springer, Heidelberg.

Dmitry Zelenko, Chinatsu Aone, and Jason Tibbets. 2004. Binary integer programming for information extraction. In *Proceedings of the ACE Evaluation Meeting*.