Statistical Ranking in Tactical Generation

Erik Velldal
University of Oslo (Norway)
erik.velldal@ifi.uio.no

Stephan Oepen
University of Oslo (Norway)
and CSLI Stanford (CA)
oe@csli.stanford.edu

Abstract

In this paper we describe and evaluate several statistical models for the task of *realization ranking*, i.e. the problem of discriminating between competing surface realizations generated for a given input semantics. Three models (and several variants) are trained and tested: an n-gram language model, a discriminative maximum entropy model using structural information (and incorporating the language model as a separate feature), and finally an SVM ranker trained on the same feature set. The resulting hybrid tactical generator is part of a larger, semantic transfer MT system.

1 Introduction

This paper describes the application of several different statistical models for the task of *realization ranking* in tactical generation, i.e. the problem of choosing among multiple paraphrases that are generated for a given meaning representation. The specific realization component we use is the open-source chart generator of the Linguistic Knowledge Builder (LKB; Carroll, Copestake, Flickinger, & Poznanski, 1999; Carroll & Oepen, 2005). Given a meaning representation in the form of Minimal Recursion Semantics (MRS; Copestake, Flickinger, Malouf, Riehemann, & Sag, 1995), the generator outputs English realizations in accordance with the HPSG LinGO English Resource Grammar (ERG; Flickinger, 2002).

As an example of generator output, a sub-set of alternate realizations that are produced for a single input MRS is shown in Figure 1. For the two data sets considered in this paper, the average number of realizations produced by the generator is 85.7 and 102.2 (the maximum numbers are 4176 and 3408, respectively). Thus, there is immediate demand for a principled way of choosing a single output among the generated candidates. For this task we train and test three different statistical models: an n-gram language model, a maximum entropy model (MaxEnt) and a (linear) support vector machine (SVM). These are all models that have proved popular within the NLP community, but it is usually only the first of these three that has been applied to the task of ranking in sentence generation. The latter two models that we present here go beyond the surface information used by the n-gram model, and are trained on a *symmetric treebank* with features defined over the full HPSG analyses of competing realizations. Furthermore, such discriminative models are suitable for 'on-line' use within our generator—adopting the technique of *selective unpacking* from a packed forest (Carroll & Oepen, 2005)—which means our hybrid realizer obviates the need for exhaustive enumeration of candidate outputs. The present results extend our earlier work (Velldal, Oepen, & Flickinger, 2004)—and the related work of Nakanishi, Miyao, & Tsujii (2005)—to an enlarged data set, more feature types, and additional learners.

The rest of this paper is structured as follows. Section 2 first gives a general summary of the various statistical models we will be considering, as well as the measures used for evaluating them. We then go on to define the task we are aiming to solve in terms of treebank data and feature types in Section 3. By looking at different variants of the MaxEnt model we review some results for the relative contribution of individual features and the impact of frequency cutoffs for feature selection. Keeping these parameters constant then, Section 4 provides an array of empirical results on the relative performance of the various approaches.

2 Models

In this section we briefly review the different types of statistical models that we use for ranking the output of the generator. We start by describing the language model, and then go on to review the framework for discriminative MaxEnt models and SVM rankers. In the following we will use s and r to denote semantic inputs and generated realizations respectively.

Remember that dogs must be on a leash.
Remember dogs must be on a leash.
On a leash, remember that dogs must be.
On a leash, remember dogs must be.
A leash, remember that dogs must be on.
A leash, remember dogs must be on.
Dogs, remember must be on a leash.

Table 1: A small example set of generator outputs using the ERG. Where the input semantics is no specified for aspects of information structure (e.g. requesting foregrounding of a specific entity), paraphrases include all grammatically legitimate topicalizations. Other choices involve, for example, the optionality of complementizers and relative pronouns, permutation of (intersective) modifiers, and lexical and orthographic alternations.

2.1 Language Models

The use of n-gram language models is the most common approach to statistical selection in generation (Langkilde & Knight, 1998; and White (2004); inter alios). In order to better assert the relative performance of the discriminative models and the structural features we present below, we also apply a trigram model to the ranking problem. Using the freely available CMU SLM Toolkit (Clarkson & Rosenfeld, 1997), we trained a trigram model on an unannotated version of the British National Corpus (BNC), containing roughly 100 million words (using Witten-Bell discounting and back-off). Given such a model p_n, the score of a realization r_i with surface form $w_{i1}^k = (w_{i1}, \ldots, w_{ik})$ is then computed as

$$(1) \quad F(s, r_i) = \sum_{j=1}^{k} p_n(w_{i,j} | w_{i,j-n}, \ldots, w_{i,j-1})$$

Given the scoring function F, the best realization is selected according to the following decision function:

$$(2) \qquad \hat{r} = \underset{r' \in \mathcal{Y}(s)}{\arg\max}\ F(s, r')$$

Although in this case scoring is not conditioned on the input semantics at all, we still include it to make the function formulation more general as we will be reusing it later.

Note that, as the realizations in our symmetric treebank also include punctuation marks, these are also treated as separate tokens by the language model (in addition to pseudo-tokens marking sentence boundaries).

2.2 Maximum Entropy Models

Maximum entropy modeling provides a very flexible framework that has been widely used for a range of tasks in NLP, including parse selection (e.g. Johnson, Geman, Canon, Chi, & Riezler, 1999; Malouf & Noord, 2004) and reranking for machine translation (e.g. Och et al., 2004). A model is specified by a set of real-valued *feature functions* that describe properties of the data, and an associated set of *learned weights* that determine the contribution of each feature.

Let us first introduce some notation before we go on. Let $\mathcal{Y}(s_i) = \{r_1, \ldots, r_m\}$ be the set of realizations licensed by the grammar for a semantic representation s_i. Now, let our (positive) training data be given as $X_p = \{x_1, \ldots, x_N\}$ where each x_i is a pair (s_i, r_j) for which $r_j \in \mathcal{Y}(s_i)$ and r_j is annotated in the treebank as being a correct realization of s_i. Note that we might have several different members of $\mathcal{Y}(s_i)$ that pair up with s_i in X_p. In our set-up, this is the case where multiple HPSG derivations for the same input semantics project identical surface strings.

Given a set of d features (as further described in Section 3.2), each pair of semantic input s and hypothesized realization r is mapped to a feature vector $\Phi(s, r) \in \Re^d$. The goal is then to find a vector of weights $w \in \Re^d$ that optimize the likelihood of the training data. A conditional MaxEnt model of the probability of a realization r given the semantics s, is defined as

$$(3) \qquad p_w(r|s) = \frac{e^{F_w(s,r)}}{Z_w(s)}$$

where the function F_w is simply the sum of the products of all feature values and feature weights, given by

$$(4) \quad F_w(s, r) = \sum_{i=1}^{d} w_i \Phi_i(s, r) = w \cdot \Phi(s, r)$$

The normalization term Z_w is defined as

$$(5) \qquad Z_w(s) = \sum_{r' \in \mathcal{Y}(s)} e^{F_w(s,r)}$$

When we want to find the best realization for a given input semantics according to a model p_w, it is sufficient to compute the score function as in Equation (4) and then use the decision function previously given in Equation (2) above. When it comes to estimating[1] the parameters w, the procedure seeks to maximize the (log of) a penalized likelihood function as in

$$(6) \quad \hat{w} = \arg\max_{w} \log L(w) - \frac{\sum_{i=1}^{d} w_i^2}{2\sigma^2}$$

where $L(w)$ is the 'conditionalized' likelihood of the training data X_p (Johnson et al., 1999), computed as $L(w) = \prod_{i=1}^{N} p_w(r_i|s_i)$. The second term of the likelihood function in Equation (6) is a penalty term that is commonly used for reducing the tendency of log-linear models to over-fit, especially when training on sparse data using many features (Chen & Rosenfeld, 1999; Johnson et al., 1999; Malouf & Noord, 2004). More specifically it defines a zero-mean Gaussian prior on the feature weights which effectively leads to less extreme values. After empirically tuning the prior on our 'Jotunheimen' treebank (training and testing by 10-fold cross-validation), we ended up using $\sigma^2 = 0.003$ for the MaxEnt models applied in this paper.

2.3 SVM Rankers

In this section we briefly formulate the optimization problem in terms of support vector machines. Our starting point is the SVM approach introduced in Joachims (2002) for learning ranking functions for information retrieval. In our case the aim is to learn a ranking function from a set of preference relations on sentences generated for a given input semantics.

In contrast to the MaxEnt approach, the SVM approach has a geometric rather than probabilistic view on the problem. Similarly to the the MaxEnt set-up, the SVM learner will try to learn a linear scoring function as defined in Equation (4) above. However, instead of maximizing the probability of the preferred or positive realizations, we try to maximize their value for F_w directly.

Recall our definition of the set of positive training examples in Section 2.2. Let us here analogously define $X_n = \{x_1, \ldots, x_Q\}$ to be the negative counterpart, so that for a given pair $x = (s_i, r_j) \in X_n$, we have that $r_j \in \mathcal{Y}(s_i)$ but r_j is not annotated as a preferred realization of s_i. Following Joachims (2002), the goal is to minimize

$$(7) \quad V(w, \xi) = \frac{1}{2} w \cdot w + C \sum \xi_{i,j,k}$$

subject to the following constraints,

$$(8) \quad \forall ijk \text{ s.t. } (s_k, r_i) \in X_p \wedge (s_k, r_j) \in X_n : \\ w \cdot \Phi(s_k, r_i) \geq w \cdot \Phi(s_k, r_j) + 1 - \xi_{i,j,k}$$

$$(9) \quad \forall ijk : \ \xi_{i,j,k} \geq 0$$

Joachims (2002) shows that the preference constraints in Equation (8) can be rewritten as

$$(10) \quad w \cdot (\Phi(s_k, r_i) - \Phi(s_k, r_j)) \geq 1 - \xi_{i,j,k}$$

so that the optimization problem is equivalent to training a classifier on the pairwise difference vectors $\Phi(s_k, r_i) - \Phi(s_k, r_j)$. The (non-negative) slack variables $\xi_{i,j,k}$ are commonly used in SVMs to make it possible to approximate a solution by allowing some error in cases where a separating hyperplane can not be found. The trade-off between maximizing the margin size and minimizing the training error is governed by the constant C. Using the SVM^{light} package by Joachims (1999), we empirically specified $C = 0.005$ for the model described in this paper. Note that, for the experiments reported here, we will only be making binary distinctions of preferred/non-preferred realizations, although the approach presented in (Joachims, 2002) is formulated for the more general case of learning ordinal ranking functions.

Finally, given a linear SVM, we score and select realizations in the same way as we did with the MaxEnt model, according to Equations (4) and (2). Note, however, that it is also possible to use non-linear kernel functions with this set-up, since the ranking function can be represented as a linear combination of the feature vectors as in

$$(11) \quad w \cdot \Phi(s, r_i) = \sum \alpha_{j,k} \, \Phi(s_j, r_k) \Phi(s, r_i)$$

2.4 Evaluation Measures

The models presented in this paper are evaluated with respect to two simple metrics: exact match accuracy and word accuracy. The exact match measure simply counts the number of times that the model assigns the highest score to a string that exactly matches a corresponding 'gold' or reference sentence (i.e. a sentence that is marked as preferred in the treebank). This score is discounted appropriately in the case of ties between preferred and non-preferred candidates.

[1] We use the TADM open-source package (Malouf, 2002) for training the models, using its *limited-memory variable metric* as the optimization method and experimentally determine the optimal convergence threshold and variance of the prior.

if several realizations are given the top rank by the model. We also include the exact match accuracy for the five best candidates according to the models (see the n-best columns of Table 6).

The simple measure of exact match accuracy offers a very intuitive and transparent view on model performance. However, it is also in some respects too harsh as an evaluation measure in our setting since there will often be more than just one of the candidate realizations that provides a reasonable rendering of the input semantics. We therefore also include WA as similarity-based evaluation metric. This measure is based on the *Levensthein distance* between a candidate string and a reference, also known as *edit distance*. This is given by the minimum number of deletions, substitutions and insertions of words that are required to transform one string into another. If we let d, s and i represent the number of necessary deletions, substitutions and insertions respectively, and let l be the length of the reference, then WA is defined as

$$\text{(12)} \qquad \text{WA} = 1 - \frac{d + s + i}{l}$$

The scores produced by similarity measures such as WA are often difficult to interpret, but at least they provide an alternative view on the relative performance of the different models that we want to compare. We could also have used several other similarity measures here, such as the BLEU score which is a well-established evaluation metric within MT, but in our experience the various string similarity measures usually agree on the relative ranking of the different models.

3 Data Sets and Features

The following sections summarize the data sets and the feature types used in the experiments.

3.1 Symmetric Treebanks

Conditional parse selection models are standardly trained on a treebank consisting of strings paired with their optimal analyses. For our discriminative realization ranking experiments we require training corpora that provide the inverse relation. By assuming that the preferences captured in a standard treebank can constitute a *bidirectional* relation, Velldal et al. (2004) propose a notion of symmetric treebanks as the combination of (a) a set of pairings of surface forms and associated semantics; combined with (b) the sets of alternative analyses for each surface form, and (c) sets of alternate realizations of each semantic form. Using the semantics of the preferred analyses in an existing treebank as input to the generator, we can produce all equivalent paraphrases of the original string. Furthermore, assuming that the original surface form is an optimal verbalization of the corresponding semantics, we can automatically label the preferred realization(s) by matching the *yields* of the generated trees against the original strings in the 'source' treebank. The result is what we call a *generation treebank*, which taken together with the original parse-oriented pairings constitute a full symmetrical treebank.

Jotunheimen

Bin	Items	Words	Trees	Gold	Chance
$100 \leq n$	396	21.7	367.4	20.7	0.083
$50 \leq n < 100$	246	18.5	73.7	11.5	0.160
$10 \leq n < 50$	831	14.8	24.2	6.3	0.277
$5 \leq n < 10$	426	10.1	7.0	3.0	0.436
$1 < n < 5$	291	11.2	3.3	1.6	0.486
Total	**2190**	**15.1**	**85.7**	**8.2**	**0.287**

Rondane

Bin	Items	Words	Trees	Gold	Chance
$100 \leq n$	107	21.8	498.4	17.8	0.060
$50 \leq n < 100$	63	19.1	72.9	12.0	0.162
$10 \leq n < 50$	244	15.2	23.4	4.9	0.234
$5 \leq n < 10$	119	11.9	7.2	2.7	0.377
$1 < n < 5$	101	9.3	3.21	1.5	0.476
Total	**634**	**15.1**	**102.2**	**6.8**	**0.263**

Table 2: Some core metrics for the symmetric treebanks *'Jotunheimen'* (top) and *'Rondane'* (bottom). The former data set was used for development and cross-validation testing, the latter for cross-genre held-out testing. The data items are aggregated relative to their number of realizations. The columns are, from left to right, the subdivision of the data according to the number of realizations, total number of items scored (excluding items with only one realization and ones where all realizations are marked as preferred), average string length, average number of realizations, and average number of references. The rightmost column shows a random choice baseline, i.e. the probability of selecting the preferred realization by chance.

We have successfully applied this technique to the tourism segments of the LinGO Redwoods treebank, which in turn is built atop the ERG.[2]

[2]See '`http://www.delph-in.net/erg/`' for fur-

Table 2 summarizes the two resulting data sets, which are both comprised of instructional texts on tourist activities, the application domain of the background MT system.

3.2 Feature Templates

For the purpose of parse selection, Toutanova, Manning, Shieber, Flickinger, & Oepen (2002) and Toutanova & Manning (2002) train a discriminative log-linear model on the Redwoods parse treebank, using features defined over *derivation trees* with non-terminals representing the *construction types* and *lexical types* of the HPSG grammar (see Figure 1). The basic feature set of our MaxEnt realization ranker is defined in the same way (corresponding to the PCFG-S model of Toutanova & Manning, 2002), each feature capturing a sub-tree from the derivation limited to depth one. Table 3 shows example features in our MaxEnt and SVM models, where the feature template # 1 corresponds to local derivation sub-trees. To reduce the effects of data sparseness, feature type # 2 in Table 3 provides a back-off to derivation sub-trees, where the sequence of daughters is reduced, in turn, to just one of the daughters. Conversely, to facilitate sampling of larger contexts than just sub-trees of depth one, feature template # 1 allows optional grandparenting, including the upward chain of dominating nodes in some features. In our experiments, we found that grandparenting of up to three dominating nodes gave the best balance of enlarged context vs. data sparseness.

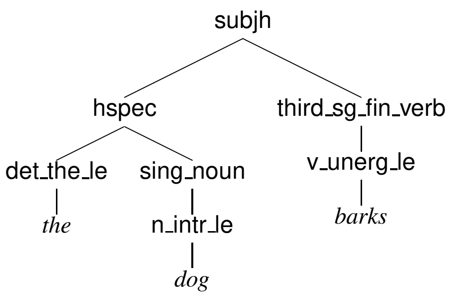

Figure 1: Sample HPSG derivation tree for the sentence *the dog barks*. Phrasal nodes are labeled with identifiers of grammar rules, and (pre-terminal) lexical nodes with class names for types of lexical entries.

In addition to these dominance-oriented features taken from the derivation trees of each realization, our models also include more surface-oriented features, viz. n-grams of lexical types with or without lexicalization. Feature type # 3 in Table 3 defines n-grams of variable size, where (in a loose analogy to part of speech tagging) sequences of lexical types capture syntactic category assignments. Feature templates # 3 and # 4 only differ with regard to lexicalization, as the former includes the surface token associated with the rightmost element of each n-gram. Unless otherwise noted, we used a maximum n-gram size of three in the experiments reported here, again due to its empirically determined best overall performance.

Id	Sample Features
1	⟨0 subjh hspec third_sg_fin_verb⟩
1	⟨1 $\triangle$ subjh hspec third_sg_fin_verb⟩
1	⟨0 hspec det_the_le sing_noun⟩
1	⟨1 subjh hspec det_the_le sing_noun⟩
1	⟨2 $\triangle$ subjh hspec det_the_le sing_noun⟩
2	⟨0 subjh third_sg_fin_verb⟩
2	⟨0 subjh hspce⟩
2	⟨1 subjh hspec det_the_le⟩
2	⟨1 subjh hspec sing_noun⟩
3	⟨1 n_intr_le *dog*⟩
3	⟨2 det_the_le n_intr_le *dog*⟩
3	⟨3 $\triangleleft$ det_the_le n_intr_le *dog*⟩
4	⟨1 n_intr_le⟩
4	⟨2 det_the_le n_intr_le⟩
4	⟨3 $\triangleleft$ det_the_le n_intr_le⟩

Table 3: Examples of structural features extracted from the derivation tree in Figure 1. The first column identifies the feature template corresponding to each example; in the examples, the first integer value is a parameter to feature templates, i.e. the depth of grandparenting (types 1 and 2) or n-gram size (types 3 and 4). The special symbols $\triangle$ and $\triangleleft$ denote the root of the tree and left periphery of the yield, respectively.

The number of instantiated features produced by the feature templates easily grows quite large. For the 'Jotunheimen' data the total number of distinct feature instantiations is 312,650. For the experiments in this paper we implemented a simple frequency based cutoff by removing features that are observed as *relevant* less than c times. We here follow the approach of Malouf & Noord (2004) where *relevance* of a feature is simply defined as taking on a different value for any two competing candidates for the same input. A feature is only included in training if it is relevant for more than c items in the training data. Table 4 shows the effect on the accuracy of the MaxEnt model when varying the cutoff. We see that a model can be

ther information and download pointers.

Cutoff	Features	Accuracy
–	312,650	71.18
1	264,455	71.18
2	112,051	70.03
3	66,069	70.28
4	46,139	69.30
5	35,295	67.93
10	16,036	65.36
20	7,563	63.05
50	2,605	59.10
100	889	54.21
200	261	50.11
500	34	34.70

Table 4: The effects of frequency-based feature selection with respect to model size and accuracy.

model configuration	match	WA
basic model of (Velldal et al., 2004)	63.09	0.904
basic plus partial daughter sequence	64.64	0.910
basic plus grandparenting	67.54	0.923
basic plus lexical type trigrams	68.61	0.921
basic plus all of the above	70.28	0.927
basic plus language model	67.96	0.912
basic plus all of the above	72.28	0.928

Table 5: Performance summaries of best-performing realization rankers using various feature configurations, when compared to the set-up of Velldal et al. (2004). These scores where computed using a relevance cutoff of 3 and optimizing the variance of the prior for individual configurations.

compacted quite aggressively without sacrificing much in performance. For all models presented below we use a cutoff of $c = 3$.

4 Results

In this section we present contrastive results for the models defined in Section 2 above, evaluated against the exact match accuracy and word accuracy as described in Section 2.4.

As can be seen in Table 6, both the MaxEnt and SVM learner does a much better job than the n-gram model at identifying the correct reference strings. The two discriminative models perform very similarly, however, although the MaxEnt model often seems to do slightly better.

When working with a cross-validation set-up the difference between the learners can conveniently be tested using an approach such as the cross-validated paired t-test described by Dietterich (1998). We also tried this approach using the Wilcoxon Matched-Pairs Signed-Ranks test as a non-parametric alternative without the assumption of normality of differences made in the t-test. However, none of the two tests found that the differences between the MaxEnt model and the SVM model were significant for $\alpha = 0.05$ (using two-sided tests).

Note that, due to memory constraints, we only included a random sample of maximum 50 non-preferred realizations per item in the training data used for the SVM ranker. Even so, the SVM trained on the full 'Jotunheimen' data had a total of 66,621 example vectors in its training data, which spawned a total of 639,301 preference constraints with respect to the optimization problem of Equations 8 and 10. We did not try to maximize performance on the development data by repeatedly training with different random samples, but this might be one way to improve the results.

Although we were only able to present results using linear kernels for the SVM ranker in this paper, preliminary experiments using a *polynomial* kernel seem to give promising results. Due to memory constraints and long convergence times, we were only able to train such a model on half of the 'Jotunheimen' data. However, when testing on the remaining half, it achieved an exact match accuracy of 71.03%. This is comparable to the performance achieved by the linear SVM through full 10-fold training and testing. Moreover, there is reason to believe that these results will improve once we manage to train on the full data set.

In order to assess the effect of increasing the size of the training set, Figure 3 presents learning curves for two MaxEnt configurations, viz. the basic configurational model and the one including all features but the language model. Each data point

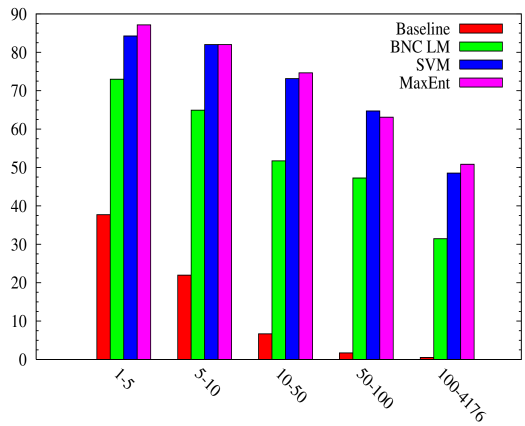

Figure 2: Exact match accuracy scores for the different models. Data items are binned with respect to the number of distinct realizations.

Model	Jotunheimen accuracy	Jotunheimen n-best	Jotunheimen WA	Rondane accuracy	Rondane n-best	Rondane WA
BNC LM	53.24	78.81	0.882	54.19	77.19	0.891
SVM	71.11	84.69	0.922	63.64	83.12	0.906
MaxEnt	72.28	84.59	0.927	64.28	83.60	0.903

Table 6: Performance of the different learners. The results on the 'Jotunheimen' treebank for the discriminative models are averages from 10-fold cross-validation. A model trained on the entire 'Jotunheimen' data was used when testing on 'Rondane'. Note that the training accuracy of the SVM learner on the 'Jotunheimen' training set is 91.69%, while it's 92.99% for the MaxEnt model.

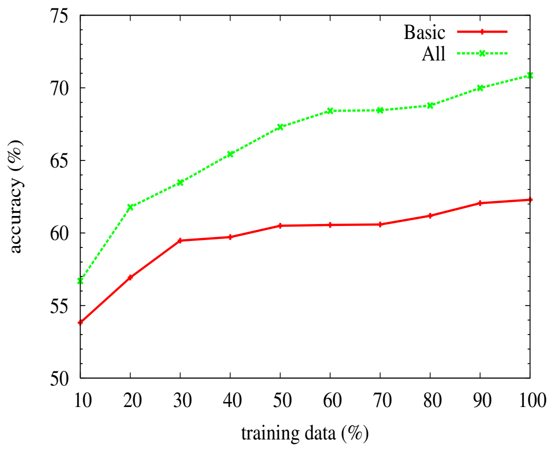

Figure 3: Learning curves for two MaxEnt model configurations (trained without cutoffs). Although there appears to be a saturation effect in model performance with increasing amounts of 'Jotunheimen' training data, for the richer configuration (using all features but the language model) further enlarging the training data still seems attractive.

corresponds to average exact match performance for 10-fold cross-validation on 'Jotunheimen', but restricting the amount of training data presented to the learner to between 10 and 100 per cent of the total. At 60 per cent training data, the two models already perform at 60.6% and 68.4% accuracy, and the learning curves are starting to flatten out. Somewhat remarkably, the richer model including partial daughter back-off, grandparenting, and lexical type trigrams already outperforms the baseline model by a clear margin with just a small fraction of the training data, so the MaxEnt learner appears to make effective use of the greatly enlarged feature space.

When testing against the 'Rondane' held-out set and comparing to performance on the 'Jotunheimen' cross-validation set, we see that the performance of both the MaxEnt model and the SVM degrades quite a bit. Of course, some drop in performance is to be expected as the estimation parameters had been tuned to this development set. Furthermore, as can be seen from Table 2, the baseline is also slightly lower for the 'Rondane' test set as the average number of realizations is higher. Also, while basically from the same domain, the two text collections differ noticeably in style: 'Jotunheimen' is based on edited, high-quality guide books; 'Rondane' has been gathered from a variety of web sites. Note, however, that the performance of the BNC n-gram model seems to be more stable across the different data sets.

In any case we see that, for our realization ranking task, the use of discriminative models in combination with structural features extracted from treebanks, clearly outperforms the surface oriented, generative n-gram model. This is in spite of the relatively modest size of the treebanked training data available to the discriminative models. On the 'Rondane' test set the reduction in error rate for the combined MaxEnt model relative to the n-gram LM, is 22.03%. The error reduction for the SVM over the LM on 'Rondane' is 20.63%.

Another factor that is likely to be important for the differences in performance is the fact that the treebank data is better tuned to the domain of application or the test data. The n-gram language model, on the other hand, was only trained on the general-domain BNC data. Note, however, that when testing on 'Rondane', we also tried to combine this general-domain model with an additional in-domain model trained only on the text that formed the basis of the 'Jotunheimen' treebank, a total of 5024 sentences. The optimal weights for linearly combining these two models were calculated using the interpolation tool in the CMU toolkit (using the expectation maximization (EM) algorithm, minimizing the perplexity on a held out data set of 330 sentences). However, when applied to the 'Rondane' test set, this in-

model	error	ties	correct
BNC LM	253	68	313
MaxEnt (sans LM)	222	63	349
MaxEnt (combined)	225	3	404

Table 7: Exact match error counts for three models, viz. the BNC LM only, the MaxEnt model by itself (using all feature types except the LM probability), and the combined MaxEnt model. The intermediate column corresponds to *ties* or *partial* errors, i.e. the number of items for which multiple candidates were ranked at the top, of which some were actually preferred and some not. Primarily this latter error type is reduced by including the LM feature in the MaxEnt universe.

terpolated model failed to improve on the results achieved by just using the larger general-domain model alone. This is probably due to the small amount of domain specific data that we presently have available for training.

Another observation about our n-gram experiments that is worth a mention is that we found that ranking realizations according to non-normalized log probabilities directly resulted in much better accuracy than using a length normalized score such as the geometric mean.

Finally, Table 7 breaks down per-item exact match errors for three distinct ranking configurations, viz. the BNC LM only, the structural MaxEnt model only, and the combined MaxEnt model, which includes the LM probability as an additional feature; all numbers are for application to the held-out 'Rondane' test set. Further contrasting the first two of these, the BNC LM yields 129 unique errors, in the sense that the structural MaxEnt makes the correct predictions on these items, contrasted to 98 unique errors in the structural MaxEnt model. When compared to the only 124 errors made equally by both rankers, we conclude that the different approaches have partially complementary strengths and weaknesses. This observation is confirmed in the relatively substantial improvement in ranking performance of the combined model on the 'Rondane' test: The exact match accuracies of the n-gram model, the basic MaxEnt model and the combined model are 54.19%, 59.43% and 64.28%, respectively.

5 Summary and Outlook

Applying three alternate statistical models to the realization ranking task, we found that discriminative models with access to structural information substantially outperform the traditional language model approach. Using comparatively small amounts of annotated training data, we were able to boost ranking performance from around 54% to more than 72%, albeit for a limited, reasonably coherent domain and genre. The incremental addition of feature templates into the MaxEnt model suggests a trend of diminishing return, most likely due to increasing overlap in the portion of the problem space captured across templates, and possibly reflecting limitations in the amount of training data. The comparison of the MaxEnt and SVM rankers suggest comparable performance on our task, not showing statistically significant differences. Nevertheless, in terms of scalability when using large data sets, it seems clear that the MaxEnt framework is a more practical and manageable alternative, both in terms of training time and memory requirements.

As further work we would like to try to train an SVM that takes full advantage of the ranking potential of the set-up described in (Joachims, 2002). Instead of just making binary (right/wrong) distinctions, we could grade the realizations in the training data according to their WA scores toward the references and try to learn a similar ranking. So far we have only been able to do preliminary experiments with this set-up on a small sub-set of the data. When evaluated with the accuracy measures used in this paper the results were not as good as those obtained when training with only two ranks, however this might very well look different if we evaluate the full rankings (e.g. number of swapped pairs) instead of just focusing on the top ranked candidates. Note that it is also possible to use such graded training data with the MaxEnt models, by letting the probabilities of the empirical distribution be based on similarity scores such as WA instead of frequencies.

Acknowledgments

The work reported here is part of the Norwegian LOGON project on precision MT, and we are grateful to numerous colleagues; please see '`http://www.emmtee.net`' for background. Furthermore, we warmly acknowledge the support and productive criticism provided by Dan Flickinger (the ERG developer), Francis Bond, John Carrol, and three anonymous reviewers.

References

Carroll, J., Copestake, A., Flickinger, D., & Poznanski, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation.* Toulouse.

Carroll, J., & Oepen, S. (2005). High-efficiency realization for a wide-coverage unification grammar. In R. Dale & K. fai Wong (Eds.), *Proceedings of the 2nd International Joint Conference on Natural Language Processing* (Vol. 3651, pp. 165 – 176). Jeju, Korea: Springer.

Chen, S. F., & Rosenfeld, R. (1999). *A Gaussian prior for smoothing maximum entropy models* (Tech. Rep.). Carnegie Mellon University. (Technical Report CMUCS-CS-99-108)

Clarkson, P., & Rosenfeld, R. (1997). Statistical language modeling using the CMU-Cambridge Toolkit. In *Proceedings of ESCA Eurospeech.*

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., & Sag, I. (1995). Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation.* Leuven, Belgium.

Dietterich, T. G. (1998). Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923.

Flickinger, D. (2002). On building a more efficient grammar by exploiting types. In S. Oepen, D. Flickinger, J. Tsujii, , & H. Uszkoreit (Eds.), *Collaborative language engineering: A case study in efficient grammar-based processing* (pp. 1–17). CSLI Press.

Joachims, T. (1999). Making large-scale svm learning practical. In B. Schölkopf, C. Burges, & A. Smola (Eds.), *Advances in kernel methods - support vector learning.* MIT-Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the ACM conference on knowledge discovery and data mining (KDD).* ACM.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic 'unification-based' grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics* (pp. 535 – 541). College Park, MD.

Langkilde, I., & Knight, K. (1998). The practical value of n-grams in generation. In *International natural language generation workshop.*

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning* (pp. 49–55). Taipei, Taiwan.

Malouf, R., & Noord, G. van. (2004). Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP workshop Beyond Shallow Analysis.* Hainan, China.

Nakanishi, H., Miyao, Y., & Tsujii, J. (2005). Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technologies* (pp. 93 – 102). Vancouver, Canada: Association for Computational Linguistics.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., & Radev, D. (2004). A smorgasbord of features for statistical machine translation. In *Proceedings of the 5th Conference of the North American Chapter of the ACL.* Boston.

Toutanova, K., & Manning, C. D. (2002). Feature selection for a rich HPSG grammar using decision trees. In *Proceedings of the 6th Conference on Natural Language Learning.* Taipei, Taiwan.

Toutanova, K., Manning, C. D., Shieber, S. M., Flickinger, D., & Oepen, S. (2002). Parse disambiguation for a rich hpsg grammar. In *First workshop on treebanks and linguistic theories.* Sozopol, Bulgaria.

Velldal, E., Oepen, S., & Flickinger, D. (2004). Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd workshop on Treebanks and Linguistic Theories.* Tübingen, Germany.

White, M. (2004). Reining in CCG chart realization. In *Proceedings of the 3rd International Conference on Natural Language Generation.* Hampshire, UK.