LOGON

Technical Report # 2007-10

Discriminative Realization Ranking

(Draft of November 30, 2008)

Erik Velldal

Department of Informatics

University of Oslo

Contents

1 Introduction

In this chapter we consider the issue of *realization ranking*, i.e. the problem of ranking the target sentences produced by the generator. Any grammar-based natural language generation (NLG) system with sufficiently wide coverge is deemed to face the problem of *indeterminacy*. This problem concerns the fact that the rules of the grammar often will admitt many possible surface realizations for a given semantic input. For a given target MRS passed on from transfer, the LKB generator (see Chapter **??**) used in LOGON will typically produce hundreds—sometimes even thousands—of different English surface realization. The sources for this indeterminacy are many and include phenomena such as the optionality of complementizers and relative pronouns, permutation of (intersective) modifiers, lexical and orthographic alternations, and (if uncerspecified in the MRS) choices pertaining to topicalization and passivization.

All of the alternative realizations of a given MRS will be what we regard as *paraphrases*, meaning that they are all semantically equivalent, at least in a strict truth-conditional sense. Although all of these paraphrases will also be well-formed according to the underlying grammar (i.e. the LinGO ERG, as described in Chapter **??**), some of them will usually sound much more natural and fluent than others. As pointed out by, among others, Abney (1996), whereas *grammaticality* (at least in computational terms) is an absolute or two-valued property (a given sentence either is or is not grammatical according to a given grammar), *naturalness* is a matter of degree. Moreover, as the coverage and scope of the underlying generation grammar increases, so does typically also the number of realizations that it can produce for a given meaning.

We see that there is clear need for a principled and scalable method for automatically scoring and ranking the competing realizations. In this chapter we describe the type of data-driven model that we employ for this task within LOGON. In short, we show how it is possible train discriminative models for realization ranking in a similar manner as when training models for statistical parse disambiguation. By introducing the notion of a *generation treebank*, we are able to adapt and extend on the methodology of state-of-the-art statistical parsing and statistical unification-based grammars, thus making it applicable to the context of NLG. We also compare the performance of our novel discriminative treebank model to the performance of a more traditional n-gram language model, as well as a model that combines both types of information.

The rest of this chapter is organized as follows. In the next section we first provide some more background on the problem of realization ranking and the types of models that we employ. Section 3 describes the treebank data that we use for trainig our discriminative models, including the notion of a generation treebank.

In Section **??** we then present various evalutation results for the models, using approaches based on both human judges and automatic metrics. We finally summarize our findings and give some concluding remarks in Section 8.

Before we go on, it is important to note that the primary focus of this chapter is restricted to the problem of ranking generator outputs. Although the particular generator we use is embedded in the overal LOGON MT system, our ranking task is restricted to the context of generation. This is an important distinction. It means that we are not concerned with ranking sentences as translations of a foreign source sentence, but rather as realizations of a semantic representation. The problem of finally ranking the output translations (conditioned on the input source) is deferred to Chapter **??**, where we describe a discriminative model for end-to-end reranking. This reranking model of Chapter **??** incoporates the realization ranker described in the current chapter as a separate feature, in addition to several other global features of the target sentences.

2 Background

Figure 1 shows some examples of alternative outputs when generating from a single (underspecified) MRS using the LinGO ERG. We see that, while a linguistic precision grammar goes a long way towards guaranteeing grammaticality of all realizations (to the level of providing the so-called *that* filter on subject extraction, for example), clearly some outputs are far more fluent than others. For the (non-deterministic) items in the test data that we consider in this chapter we get close to ${}_f\text{ix}_m\text{e}_r$ealizations on average, where the maximium is ${}_f\text{ix}_m\text{e}_c$andidates for a single input MRS (this maximum, however, is specific to our data set and could well be larger).

The traditional approach to ranking alternative generation outputs is to score the surface strings using a generative n-gram-based *language model* (LM). An n-gram model factorizes the probability of a sentence into the product of the individual word probabilities, and each word probability is only conditioned on the $n-1$ words preceding it in the sequence. The approach of using an n-gram LM for realization ranking was pioneered in the hybrid NLG system Nitrogen (Knight & Hatzivassiloglou, 1995; Langkilde & Knight, 1998a) and its successor HALOgen (Langkilde, 2002), in which the strings are scored according to a *bigram* model (i.e. $n = 2$). Similar approaches based on n-gram statistics has later been used in many other generator systems, such as those described by Bangalore & Rambow (2000), Ratnaparkhi (2000), White (2004), Habash (2004), and others.

One advantage of using n-gram LMs is that they are relatively easy to estimate, and they can be trained on "raw" unannotated text. However, there are also many

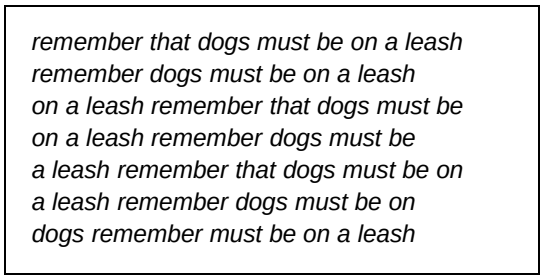


Figure 1: Example sets of generator outputs using the LinGO ERG. Unless the input semantics is specified for aspects of information structure (e.g. requesting foregrounding of a specific entity), paraphrases will include all grammatically legitimate topicalizations. Other sources of generator ambiguity include, for example, the optionality of complementizers and relative pronouns, permutation of (intersective) modifiers, and lexical and orthographic alternations.

limitations inherent to the n-gram approach. The most obvious such limitation, as already pointed out by Langkilde & Knight (1998b), is that an ordinary n-gram language model cannot capture long-range dependencies and dependencies between non-contiguous words. An important part of this problem is, of course, the fact that a purely surface oriented n-gram model will fail to capture dependencies that show a structural rather than sequential regularity. The deeper structures of the strings are ignored entirely. Neither can the model capture dependencies that hold between more than n words. These are some of the reasons why it seems reasonable to assume that the quality of the generator rankings can be improved if we aim to go beyond the abilities of the standard n-gram models, and try to incorporate more information about the linguistic structure of the realizations.

The realization ranking incorporated in LOGON follows a rather different route and draws heavily on previous research on a different but related ranking task; *statistical parse selection*. Compared to the field of NLG, models for statistical ranking have received a lot more attention within the area of natural language understanding (NLU) or *parsing*. Due to its significantly longer history of research, the field of statistical parsing is in many ways much more mature than the field of statistical generation, and statistical parse selection models have proved especially well-suited for capturing soft constraint that are difficult to encode directly in the grammar or to define in terms of explicit rules.

In our case, working with grammar-based generation using a linguistically fine-grained and wide-coverage HPSG grammar such as the ERG, there are certain areas within statistical NLU that immediately stand out as particularly interesting. The work on learning *stochastic unification based grammars* (SUBGs), as pioneered by

Abney (1997) and Johnson, Geman, Canon, Chi, & Riezler (1999), are among these. As any large-scale wide-coverage grammar of a natural language is destined to be massively ambiguous, there is an immediate need to be able to efficiently order the various hypotheses in a systematic way. Johnson et al. (1999) show how *conditional log-linear models* can be used for efficiently estimating statistical parse disambiguation models for large-scale unification-based grammars. As further described in Section 4 below, log-linear models are defined in terms of *feature functions* that can be designed to record arbitrary properties of the structures that we are interested in modeling. Commonly the features are set up to record the grammatical productions in the parse trees, and the estimation of the model parameters is then carried out on the basis of a *treebank*. Generally speaking, a parse treebank is a corpus where strings have been annotated with grammatical structure. In the case of treebanks based on unification grammars, the sets of available parses licensed by the grammar for each string have typically been manually disambiguated in order to indicate which is considered to be preferred or optimal. When estimating a conditional log-linear model parse selection, as in the work by Johnson et al. (1999), the models parameters are chosen to maximize the probability of the preferred parses relative to all the other non-preferred parses.

The tasks of ranking parses and ranking realizations can be seen to closely parallel eachother, and in many ways there is a relation of *inverse similarity* between these two problems. While parsing attempts to recover the underlying meaning and structure of a given surface utterance, generation attempts to express a given meaning as a surface utterance. In both directions of processing, however, the underlying grammar will usually license many possible hypotheses, and correspondingly there is a need for ordering these hypotheses in a principled and systematic manner. We see that the two ranking tasks parallel each other closely, and in both cases the goal is to find the optimal output structure under some set of constraints.

In extension of the research pioneered by Abney (1997) and Johnson et al. (1999), the literature contains many other examples of applying log-linear models for the problem of parse disambiguation, such as the work by Osborne (2000), Malouf (2002), Riezler et al. (2002), Miyao & Tsujii (2002) and Malouf & van Noord (2004), to name a few. However, our starting point for adapting this modeling approach for the purpose of realization ranking, is provdided by the work of Toutanova, Manning, Flickinger, & Oepen (2005) on building discriminative log-linear models for parse disambiguation on the HPSG-based Redwoods treebank (Oepen et al., 2002). One reason why this work forms an especially well-suited starting is that the LOGON treebanks can effectively be viewed as a separate branch of Redwoods. As described in Chapter **??**, Redwoods is annotated in accordance with the ERG, i.e. the same grammar that we use for generation within LOGON, including semantic analysis in the form of MRS. As further detailed in Section 4 below, the core

set of features included in our log-linear realization rankers will also be defined as extensions to the feature set used by Toutanova et al. (2005), viz. various structural features defined over the grammatical derivation of the realizations. However there are a few crucial modifications that need to be done with respect to the treebank data before we are in a position to train discriminative models for realization ranking. The next section describes the notion of a *symmetric treebank*, including the important subset termed a *generation treebank*, as introduced by Velldal, Oepen, & Flickinger (2004).

3 Symmetric Treebanks

As noted above, the discriminative parse selection models are trained by maximizing the probability of all the preferred analyses relative to all the alternative and non-preferred analyses. This gives us a statistical model for the distribution of parses conditioned on a given input string. For the purpose of realization ranking, however, we are interested in modeling a somewhat different distribution, viz. the distribution of strings given the semantics. Estimating such a model would mean maximizing the probability of the preferred realizations relative to all the alternative and non-preferred realizations. However, as there is an implicit directionality inherent to the annotations of traditional parse-oriented treebanks, they do not immediately offer the kind of training data we require. The optimality relations encoded in these treebanks are conceived as mappings *from* strings *to* analyses. This relation is represented in Figure 2(a) below, where the arrow represents the optimality relation and the other arcs correspond to competing (sub-optimal) parses. In other words, the arrow points from the observed string to the disambiguated gold analysis. (To best understand this figure it should be read as if we had zoomed in on a single item in the treebank.)

As argued by Velldal et al. (2004) and (Velldal, 2008), however, for the purposes of training a statistical "generation grammar", it seems reasonable to make the assumption that the treebanked strings can also be treated as optimal realizations of the treebanked semantics. In other words, the suggestion is to view the optimality relations in the treebank as *bidirectional* or *symmetric*. What this effectively means in practise, is that we take the original sentences in the corpus to define the reference realizations for the corresponding treebanked semantics. As described in (Velldal et al., 2004) and Velldal (2008), this assumpiton forms the basis of a fully automated procedure for contsructing the training data required for a discriminative realization rankeron the basis of an existing Redwoods parse treebank (Velldal et al., 2004). We outline this procedure below.

Recall that the MRS component of the HPSG annotations in the Redwoods-style

treebanks can also be used as input to the LKB generator. This means that we can take the semantics of the originally treebanked analysis and exhaustively generate all the possible paraphrases that express this meaning, as licensed by the underlying grammar. Given the assumption of bidirectional optimality, the generated paraphrases matching the original string in the underlying corpus can then be labeled as the optimal or preferred candidates. This results in sets of relations as those illustrated in Figure 2(b). This kind of expanded treebank resource is what Velldal et al. (2004) termed a *symmetric treebank*. A symmetrized treebank consists of

(i) gold-labeled pairs $\langle$*utterance*, *analysis*$\rangle$ that are considered bidirectionally optimal,

(ii) the set of competing analyses for each *utterance*, and

(iii) the set of competing paraphrases for each *analysis*.

Note that the bidirectional optimality pertains to the level of strings and semantics, which are the input to the parser and the generator respectively.

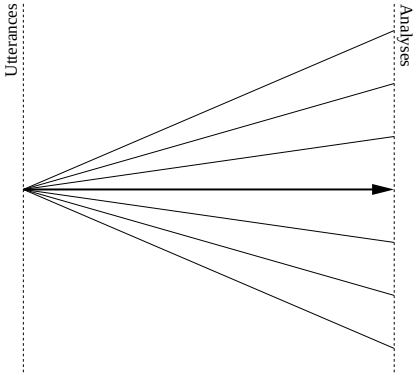
In contexts where we need to be clear about the specific "side" of a symmetric treebank that we are referring to, we sometimes use the terms *parse treebank* and *generation treebank* (Velldal et al., 2004; Velldal & Oepen, 2005). The side corresponding to the generation treebank is isolated in Figure 2(c). This is exactly the data that we need in order to train a realization ranker in a similar way as for discriminative parse selection models.

A more detailed step-by-step description of how we construct the generation treebanks used for training our rankers can be found in Velldal (2008). In addition to the two steps oulined above (i.e. paraphrasing the original treebank items followed by identification of the references), a third step correponds to pruning the resulting data items. This simply amounts to removing items for which there is no interesting indeterminacy and therefore are not relevant for learning (e.g. items with only a single realizations or items where all the realizations have identical yields). In sum then, the procedure for deriving a generation treebank on the basis of an existing parse treebank consists of three main steps:

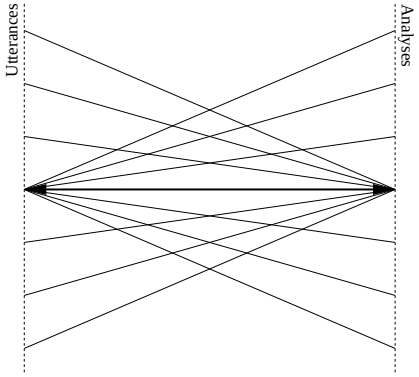
(i) Paraphrasing, i.e. exhaustively generating all the possible realizations for each gold MRS.

(ii) Labeling, i.e. matching the generator output against the original treebanked strings to identify gold realizations.

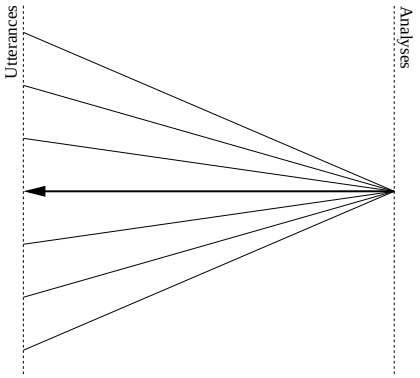
(iii) Pruning, i.e. throwing away items that are not relevant for learning (e.g. items with only one realization).



(a) Parse treebank



(b) Symmetric treebank



(c) Generation treebank

Figure 3: Single-item view of different types of treebanks. Arrows indicate preference or optimality relations. (a) Parse treebank of utterances paired with possible analyses. (b) Symmetric treebank including all paraphrases of the treebanked semantic analyses, and assuming bidirectional optimality relations. (c) Generation treebank of semantic analysis paired with possible realizations.

Before we move on to look at some properties of some actual data sets created with this procedure, it is worth spending a few more comments on the implementation of the second step above, the labeling of references. When it comes to identifying the references on the basis of the originally treebanked strings, the matching could potentially be done on several different levels. One option could be to match the full HPSG signs, or the corresponding derivation tree. Another option is to match the surface strings themselves. For the experiments reported in this chapter, however, the matching is done on the level of the *preterminal yields* of the generated derivations. As further described in relation to the feature functions of

our log-linear models, presented in Section 4, the preterminal yields correspond to lexical identifiers of the ERG. Performing the matching at this level means that all realizations that have the same yield will be treated as equivalent. This furthermore means that we often end up labeling *multiple* realizations as gold for a single input MRS. Finally, note that we migth also "loose" a couple of items in the labeling step, due to the fact that we are not always able to re-generate any realizations that matches the original input in the parse treebank.

Table 1 summarizes the result of "symmetrizing" the treebanked LOGON development data and held-out test data, creating the corresponding generation treebanks. We see that the data items are broken down along several dimensions. The items are first split into bins according to their number of generated realizations (i.e. what would correspond to "ambiguity rate" in parsing terms). For each bin the table then shows the corresponding number of items, average sentence length, average number of realizations, and average number of realizations labeled as gold. As can be seen from the tables, the total average length of the 3921 candidate real-

LOGON Development Data

Aggregate	Items	Words	Trees	Gold	Baseline
$100 \leq n$	369	27.5	360.0	8.0	3.33
$50 \leq n < 100$	230	24.7	73.5	4.9	6.64
$10 \leq n < 50$	1144	20.6	22.5	3.3	17.08
$5 \leq n < 10$	868	15.3	6.9	2.2	32.13
$1 < n < 5$	1310	13.9	3.2	1.4	45.64
Total	**3921**	**18.1**	**47.3**	**3.0**	**28.05**

LOGON Held-Out Test Data

Aggregate	Items	Words	Trees	Gold	Baseline
$100 \leq n$	17	26.0	572.5	7.8	2.20
$50 \leq n < 100$	24	24.1	78.1	5.6	7.55
$10 \leq n < 50$	83	20.0	23.4	3.2	15.86
$5 \leq n < 10$	57	14.6	6.7	2.1	32.21
$1 < n < 5$	88	14.0	3.1	1.4	45.08
Total	**269**	**17.6**	**52.8**	**2.9**	**27.28**

Table 1: Some core metrics for the generation treebanks we use for training and 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, average string length, average number of realizations, average number of references, and finally the baseline for expected accuracy by random choice.

izations in the LOGON generation treebank we use for development is 18.1. For the 269 items in held-out data set the average string length is just slightly shorter, at 17.6. Note that these figures refer to the length of *tokenized* strings, which means that for example punctuation is treated as separate units. In the development data, we see that the average number of generated hypotheses per item is 47.3, while the number is 52.8 for the held-out data. Not surprisingly, we find a higher number of candidate realizations for the items which also have a longer average string length.

Although the degree of non-determinism, i.e. the average number of available candidates, is obviously an important factor characterizing the difficulty of the ranking task, another important factor is the number of candidates labeled as gold, as described in Section 3 above. This figure is around three for both data sets. On the basis of these of these two properties, we can compute a *random choice baseline* to more directly indicate the difficulty of the ranking task. This corresponds to the average exact match accuracy we could expect to obtain if we were to select candidate realizations completely at random. As we see from the final column of Table 1, the baseline figure for the full development treebank is 28.05%, while slightly lower for the held-out data, at 27.28%.

In the following sections we look at the actual models we apply for the realization ranking task. We first present the details of the log-linear model, including the feature types, and look at the corresponding results. In order to also asses the performance of our discriminative treebank model compared to the traditional apporach of using n-gram language models, Section 5 also reports on some experiments of ranking using an LM trained on the British National Corpus (BNC). All models are evaluated according to three automatic metrics:

- Exact match accuracy: The percentage of times that the top-ranked sentence is identical with the reference sentence).

- Word Accuracy (WA): String similarity-measure based on the so-called *edit distance* between a candidate string and a reference.

- NEVA: A reformulation of BLEU (Papineni, Roukos, Ward, & Zhu, 2002) provided by Forsbom (2003) in order to make it well-defined as a sentence-level metric. Computed as the arithmetic mean of the raw n-gram precision scores.

4 Ranking using a Discriminative Log-Linear Model

The flexible framework provided by log-linear modeling has been widely used for a range of tasks in NLP, including parse selection (e.g. Johnson et al., 1999; Malouf

& van 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* (f) that describe properties of the data, and an associated set of *learned weights* (λ) that determine the contribution of each feature. Given a set of d such feature functions, each observed pair of semantics s and realization r is represented as a feature vector $f(s,r) \in \Re^d$, and a vector of weights $\lambda \in \Re^d$ is then fitted to optimize the likelihood of the training data. A conditional log-linear model for the probability of a realization r given a semantics s, has the general parametric form

$$p_\lambda(r|s) = \frac{1}{Z_\lambda(s)} \; exp\left(\sum_{i=1}^{d} \lambda_i f_i(s,r)\right) \tag{1}$$

$$p_\lambda(r|s) = \frac{1}{Z_\lambda(s)} \; exp\left(\sum_{i=1}^{d} \lambda_i f_i(s,r)\right) \tag{2}$$

where Z_λ is a normalization term defined as

$$Z_\lambda(s) = \sum_{r' \in Y(s)} q(r'|s)\, exp\left(\sum_{i=1}^{d} \lambda_i f_i(s,r')\right) \tag{3}$$

and $Y(s)$ gives the set of all possible realizations of s.

The estimation[1] of the λ-parameters seek to maximize the (log of) a penalized likelihood function as in

$$\hat{\lambda} = \arg\max_{\lambda} \log L(\lambda) - \frac{\sum_{i=1}^{d} \lambda_i^2}{2\sigma^2} \tag{4}$$

where $L(\lambda)$ is the *conditionalized* likelihood of the training data, computed as $L(\lambda) = \prod_{i=1}^{N} p_\lambda(r_i|s_i)$ (Johnson et al., 1999). The second term of the likelihood function in Equation (4) 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 & van Noord, 2004). More specifically it defines a zero-mean Gaussian prior on the feature weights which effectively leads to less extreme values. Given a log-linear model p_λ, the scores used for ranking the candidate realizations can be computed simply as $score(r) = \sum_i \lambda_i f_i(r)$ since we are only interested in the relative rank order.

[1] We use the TADM open-source package (Malouf, 2002) for training, using its *limited-memory variable metric* as the optimization method. For more information see '`http://tadm.sourceforge.net/`'.

4.1 Features and Tuning

In the log-linear parse selection models build by Toutanova et al. (2005), the features are defined over HPSG *derivation trees*. This is the fundamental data type of the Redwoods treebanks. The internal nodes of a derivation tree correspond to identifiers of construction types in the underlying ERG grammar, such as the head-complement or head-adjunct schemas, while the preterminal yields correspond to identifiers of the lexical entries. For the purpose of feature extraction, however, these latter preterminal lexical identifiers are first mapped to the more abstract *lexical entry types* (LE-types) within the grammar. A sample derivation tree for the sentence *The dog barks* is shown in Figure 4, where the pre-terminal nodes have first been mapped to the such abstract lexical types.

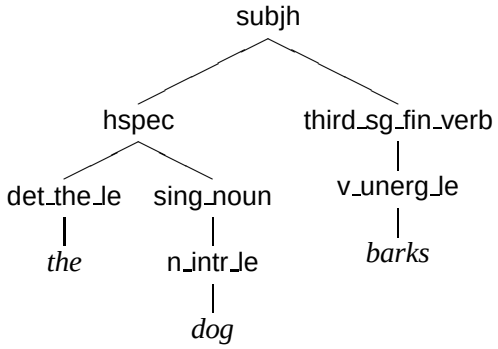


Figure 4: Sample HPSG derivation tree for the sentence *the dog barks*. Phrasal nodes are labeled with identifiers of grammar rules, and (pre-terminal) lexical nodes with class names for types of lexical entries. Note that, while the native derivation tree format has pre-terminals corresponding to lexical identifiers, this figure shows a somewhat modified format where these identifiers are mapped to one of the ERG's abstract lexical types. This is the representation that forms the basis of our treebank features, as exemplified in Table 2.

The basic feature set of our discriminative realization rankers is defined in the same way as for the CPCFG-S model of Toutanova et al. (2005). In this set-up, each feature captures a local sub-tree from the derivation, limited to depth one. In other words, these features record the particular productions observed in a given tree, such as hspec $\rightarrow$ det_the_le sing_noun. In Table 2 these features correspond to feature template # 1. The *value* of a given type # 1 feature corresponds to the number of times a given expansion occurs in the tree.

We have also experimented with several extensions to this feature set. For

example, to reduce the effects of data sparseness, we introduced a feature type #2 as seen in Table 2, which provides a back-off mechanism for the configurational features above. While the type #1 features record the full sequence of daughters in the local sub-trees, the type #2 features reduce this to only one of the daughters in turn. We sometimes refer to these features as *active edge* features, in analogy to the notion of active edges in chart-based generation/parsing. Note that, since the context that gets recorded is reduced (as compared to the type #1 features), there will also be significantly fewer unique features instantiating this template, while the total number of occurrences of each of these features will be correspondingly higher.

Conversely, to facilitate sampling of *larger* contexts than just sub-trees of depth one, feature template #1 also allows for various degrees of *grandparenting*.[2] By specifying an additional parameter to the template, the recorded information can be extended to include various levels of ancestor annotation. This ancestor parameter is available also for the active edge features (type #2), which means that, for a given node, we extract a non-branching path through the tree that includes a single daughter together with an upward chain of dominating nodes. In Table 2, the level of grandparenting is indicated by the first integer in the instantiated type #1 and #2 features. Note that what is the optimal maximum-level of grandparenting is determined empirically through experimentation.

In addition to the dominance-oriented features defined above, our models also include features that are more linearly oriented. The features of type #3 and #4 record n*-grams of lexical types*, extracted from the pre-terminal yields of the derivation trees. In loose analogy to HMM part-of-speech tagging, the n-grams of lexical types capture syntactic category assignments. The difference between templates #3 and #4 only regards *lexicalization*, as the former additionally includes the surface token associated with the rightmost element of each n-gram (again, this can be seen as loosely corresponding to the emission probabilities in an HMM tagger). An additional parameter for both of these features is the *size* of the n-grams. Just as for the level of grandparenting, the optimal value of this parameter is something which must be determined empirically.

After extensive grid search across different feature combinations, we found that using 3-level grandparenting and 3-grams over lexical types, leaving out active edges and constituent weights, resulted in the model with the overall best performance. For a more detailed review of the contribution of the individual feature types, see Velldal (2008). Note that there are in fact other feature configurations that result in similar ranking performance. However, these models include a lot

[2] By *grandparenting* we refer to all use of ancestor information, regardless of whether we for the current node are including a parent, grandparent, great grandparent, etc.

Type Id	Sample Features
1	⟨0 subjh hspec third_sg_fin_verb⟩
1	⟨1 △ 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 △ subjh hspec det_the_le sing_noun⟩
2	⟨0 subjh third_sg_fin_verb⟩
2	⟨0 subjh hspec⟩
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 ◁ det_the_le n_intr_le *dog*⟩
4	⟨1 n_intr_le⟩
4	⟨2 det_the_le n_intr_le⟩
4	⟨3 ◁ det_the_le n_intr_le⟩

Table 2: Examples of structural features extracted from the derivation tree in Figure 4. 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 △ and ◁ denote the root of the tree and left periphery of the yield, respectively.

more active features without being able to show statistically significant improvements. Given that other things are equal, it is good practice to decide by the principle of Occam's Razor. For our final configuration we therefore chose to use the simplest model (i.e. the one with fewer active features) of the ones with equivalent evaluation scores.

There are also other parameters beside the feature specifications that need to be tuned through repeated experimentation. In particular the value of prior can have a major impact on model behavior, both in terms of the number of iterations it takes to reach convergence and in terms of the ranking performance of the resulting model. For the feature configuration used here, we observed the best ranking performance for $\sigma^2 = 8 \times 10^{-4}$ (determined through ten-fold cross-validation on the development set).

Results for the Log-Linear Ranker

	Development		Held-Out	
	top	**5-best**	**top**	**5-best**
Accuracy	72.11%	88.82%	71.31%	89.34%
WA	0.941	0.984	0.941	0.987
NEVA	0.943	0.985	0.939	0.983

Table 3: Results for the best performing log-linear model as described above, tested on the generation treebanks constructed for the LOGON development data and the held-out test data (as summarized in Table 1). The results for the development set are obtained through ten-fold cross-validation. In addition to recording local subtrees of depth one, model features include trigrams across the lexical types of the preterminal yields, as well as three levels of grandparenting (i.e. upward dominating nodes). The variance of the prior is globally set to 8×10^{-4}. The column for 5-best lists reports scores that are maximized over the 5 top-ranked candidates.

4.2 Results

5 Ranking Using an N-gram Language Model

The use of n-gram language models is the most common approach to statistical selection in generation (Langkilde & Knight, 1998b; and White (2004); inter alios). In order to better assess the relative performance of the discriminative treebank model presented above, this section describes the results of applying an n-gram language model our ranking task. Using the freely available CMU SLM Toolkit[3] (Clarkson & Rosenfeld, 1997), we trained a 4-gram LM (using Witten-Bell discounting) on an unannotated version of the British National Corpus[4] (BNC), containing roughly 100 million words. Note while $n = 4$ might seem like a high value—the standard choice being to only consider bigrams, i.e. $n = 2$—the model also includes a back-off mechanism to lower-order models in cases of unseen n-grams. The vocabulary of our model comprises 25,000 words, partly extracted on the basis of the LOGON development data in order to make the model more atuned to the domain. Note that all of the model parameters have been carefully tuned to achieve the best ranking permormance with respect to the LOGON data. For details on the effects of varying the order of n, using different discounting strategies, vocabulary size, etc., see Velldal (2008).

Given an n-gram language model p_n, we compute the score of a given real-

[3]For more information, see '`http://www.speech.cs.cmu.edu/SLM/toolkit.html`'.
[4]For more information, see '`http://www.natcorp.ox.ac.uk/.`'

ization as the (log of the) probability of its surface form $w_{i1}^k = (w_{i1}, \ldots, w_{ik})$, defined as

$$\sum_{j=1}^{k} p_n(w_{i,j}|w_{i,j-n}, \ldots, w_{i,j-1}) \quad (5)$$

Note that, as the realizations in our symmetric treebank also include punctuation marks, these are also treated as separate tokens by the language model. Furthermore, the strings presented to the LM also include additional pseudo-tokens marking the sentence boundaries (*¡s¿* and *¡/s¿*). Although this way of conditioning on the sentence boundaries immediately makes sense intuitively, it is also necessary from a technical point of view, as it ensures that we get a proper probability distribution over all strings independent of length. Other forms of normalization that we applied prior to training include automatic mapping between common English and American spelling variants (as the ERG generates American English, while the BNC, of course, is British), and normalization of numerical expressions.

5.1 Results

The results of applying the LM is summarized in Table 4 below. We see that the exact match accuracy of the LM is 53.75% for the development set and 52.60% for the held-out set. Although the LM was in fact trained on the separate BNC data, some drop in performance when moving from the development set to the held-out set was still to expected as the model vocabularly is at least partly based on the former. Furthermore, although we see that the accuracy of the LM is well above the random choice baseline (28.05% and 27.28% for the dev. and held-set respectively), it is still far below the performance of the conditional log-linear ranker. The p-value computed for these differences using a two-tailed application of the non-parametric Sign-Test show that they are statistically significant at the 0.05 α-level. Comparing the results in Table 3 and Table 4 we see that the log-linear model also outperforms the LM with respect to both the string-similariry metrics. Again we find that the differences are indeed statistically significant for $p < 0.05$, using the (two-tailed) matched pairs Wilcoxon Matched-Pairs Signed-Ranks Test (which is similar to the Sign-Test but additionally takes the *magnitude* of differences into account, not just the *direction*).

6 Manual Evaluation

In assessing the relative performance of the n-gram model and the MaxEnt model, this chapter has so far only referred to the use of automatic and objective metrics such exact-match accuracy, WA, and NEVA. However, we have also performed

Results for the 4-gram LM Ranker

	Development		**Held-Out**	
	top	**5-best**	**top**	**5-best**
Accuracy	53.75%	82.64%	52.60%	81.61%
WA	0.907	0.986	0.904	0.981
NEVA	0.907	0.984	0.887	0.977

Table 4: Results for the best performing language model as described above, tested on the generation treebanks constructed for the LOGON development data and the held-out test data (see Table 1). The LM is a 4-gram model trained on the BNC using Witten-Bell discounting and a vocabulary of 25,000 thousand words (partially extracted from the development data). The column for 5-best lists reports scores that are maximized over the 5 top-ranked candidates.

a round of manual evaluation, carried out by a panel of external and anonymous judges. With the kind assistance of Emily M. Bender, a group of seven MA students within the *the Professional Master's in Computational Linguistics Program* at the University of Washington were recruited to judge the relative quality of alternative generator outputs. The judges—all native speakers of English—were given a questionnaire with a set of test items from the held-out data, and asked to assign a relative rank order to lists of candidate realizations as chosen by different models.

The questionnaire itself was compiled as follows. We started by isolating all test items in the held-out data for which the LM and the MaxEnt model disagree with respect to their top-ranked candidate, recording the candidates chosen by the respective models. For all of these items we then also recorded the corresponding reference from the underlying treebank, in addition to a randomly selected candidate. This resulted in an evaluation set of 73 test items, with up to four distinct alternative realizations for each. For more details on how the questionnaire was compiled and presented, please refer to Velldal (2008, ch. 8).

Now, for each list of alternative strings in the resulting evaluation form, the human judges were asked to rank them relative to each other according to which they considered to sound more natural. The judges where instructed to assign each candidate a rank position on a scale from 1 to 4 (or the number of candidates included in the particular list), also allowing for ties. In other words, the evaluators where not asked to make any absolute assessments of the quality of each realization, but only to assign a relative rank order on the sets of alternatives.

For each of the 73 items in the test set, the judges assigned a relative rank order to the candidates picked by the different models. By summing all of these *per-item* rank values, we can, for each judge, obtain a *system-level* rank order on

the four models themselves. By further summing the system-level rank values, we can obtain a *global* system-level rank order. This is what is shown in Table 5 below, where the models are sorted according to their rank values. For each model the data columns show, from left to right; the total sum of per-item rank values assigned by all judges; the average sum of rank values per judge; and finally the average per-item rank value. Recall that a lower score indicates higher rank (i.e. a higher relative quality according to the evaluators). As is clear from Table 5, the reference sentences (*Gold*) were deemed to have the highest overall quality. Although this particular outcome was what we had anticipated, of course, it is a reassuring result nonetheless, as the reference sentences provide the basis for the automatic and quantitative evaluations that we otherwise rely on, and even provide the basis for how the training data itself is defined for the discriminative learners. In this sense, the fact that the reference sentences clearly stand out as the best candidates according to the human evaluators is something that further validates the assumptions underlying our symmetric treebanking methodology.Also as expected, we see that the random choice baseline receives the lowest average rank. Finally, the two middle rows of the Table 5 hold the most interesting results. We see that the candidates chosen by the MaxEnt treebank model are on average judged to be better than those of the n-gram language model. The candidates selected by the MaxEnt model receives an average rank position of 1.86, while the LM ranker receives an average rank position of 2.38.

Model	**Sum of Ranks**	**Average Sum**	**Average Rank**
Gold	674	96.3	1.319
MaxEnt	951	135.9	1.861
LM	1215	173.6	2.378
Baseline	1304	186.3	2.552

Table 5: Summary of rank values assigned by the human evaluators.

It is also worth noting that the relative rank order of the different systems as assigned by the human judges, is actually the same as the rank order assigned by the automatic metrics we use, such as edit distance and exact match accuracy. In this sense, the results of the manual evaluation effort seen in Table 5 are not only an evaluation of model performance, but indirectly also an assessment of the quality of the automatic evaluation metrics. If the relative ordering resulting from the human evaluation was different from the ordering obtained through the automatic evaluation methods, we would have good reasons to be suspicious of the latter. Fortunately, the outcome of the human evaluation provides us with no such reason

for concern.

An important part of interpreting any human evaluation results is to measure the level of *inter-annotator agreement*. Now, the different rank scores shown in Table 5 seem to be separated by quite solid margins, already giving us good reasons to trust the significance of the result. However, we have also computed several independent measures that also indicate that the level of agreement among the evaluators is indeed quite high. For any candidate sentence given top rank by some judge for a given item, the average number of judges who have given the same candidate top rank is 4.5 out of 7. For the candidate realization that was ranked top by the most judges for a given item, the average number of judges agreeing is 6.0 out of 7. Both of these quite intuitive measures seem to indicate that the evaluators tend to agree quite strongly on the judgments that they make with regards to their top-ranked sentences. However, in order to get a better impression of the inter-annotator agreement on the overall rankings, we also computed *Spearman's rank correlation coefficient*, denoted as ρ. Admittedly, having only 4 different ranking levels is a bit too few for correlation measures to really be meaningful. However, the coefficient still gives some impression of the level of agreements among the judges.

The ρ coefficient is a non-parametric rank statistic for variables that are measured at the ordinal level and is equivalent to Pearson's coefficient applied to ranks instead of raw scores. Let d_i be the difference between the ranks of each candidate for a given test item. Spearman's rank correlation coefficient is then computed as

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{6}$$

where in our case we will have $n = 4$. Note that ties are handled in the same manner as for the Wilcoxon test, i.e. by using the arithmetic average of the corresponding ranks. To use the example from above, where we had $a = 1$, $b = 2$, $c = 2$, and $d = 4$, the normalized ranks would now be $a = 1$, $b = 2.5$, $c = 2.5$, and $d = 4$.

Although the correlation coefficient is based on a pairwise comparison of the judgments of two evaluators, we can easily extend this by computing, for each judge, the average correlation towards all the other judges. Finally, note that the correlation coefficient ranges from -1 for perfect negative correlation, through zero for totally independent judgments, to 1 for perfect positive correlation.

Table 6 shows Spearman's rank correlation coefficient computed for all the MA students participating in the evaluation. The first column simply enumerates the judges. The second column shows the corresponding ρ calculated as the average pairwise correlation for all items in the evaluation set. In the third column the coefficients are computed on the system-level. Using rank values that correspond to the

	Average Rank Correlation	
Judge	**Item Level**	**System Level**
1	0.687	1
2	0.781	1
3	0.727	1
4	0.701	1
5	0.695	1
6	0.747	1
7	0.748	1
All	0.727	1

Table 6: Average Spearman's rank correlation coefficient for each human evaluator.

relative rank ordering seen in Table 5, we here compute the average pairwise correlation coefficients with respect to overall judgments about the models themselves. The bottom row in the table shows the corresponding total averages.

The average Spearman correlation coefficient for all judges over the entire evaluation set is $\rho = 0.73$. Now, although this value would typically be taken to indicate a fairly high degree of agreement (recall the bounds $\rho \in [-1, 1]$), the fact that we only have four levels of ranks involved means that we still cannot say that this figure is statistically significant. However, we also computed ρ on the system-level. This was done by averaging all the pairwise correlations between all judges with respect to the relative rank order of the different models. This resulted in a correlation coefficient of $\rho = 1$. In other words, when looking at the system-level, the inter-annotator agreement could not possibly be any higher, and this time the figure is indeed statistically significant (at the 0.05 level).

As seen in Table 5, the log-linear model seems to outrank the LM by a good margin according to the human judgments. In order to more directly contrast the differences in ranks for these two models, we also applied a statistical significance test to their rank values in isolation. This was carried out as follows. For each item in the questionnaire we assign a score of $+1$ or -1, depending on whether the average rank given to the MaxEnt candidate is higher or lower than that given to the LM candidate. This results in a sequence of 73 Bernoulli trials, one for each item in the evaluation set, to which we then apply a two-tailed Sign-Test. This results in a p-value of $p = 0.012$, showing that the differences in the human rankings for the n-gram-based language model and the log-linear treebank model are statistically significant at $\alpha = 0.05$.

7 A Combined Model

Results for the Log-Linear Ranker w/LM feature

	Development		**Held-Out**	
	top	**5-best**	**top**	**5-best**
Accuracy	74.25%	91.80%	73.98%	92.94%
WA	0.944	0.986	0.944	0.992
NEVA	0.946	0.987	0.941	0.988

Table 7: Results for the best performing log-linear model that includes the LM as a separate feature. Tested on the LOGON generation treebanks of Table 1. The results for the development set are obtained through ten-fold cross-validation. The LM feature corresponds to the model tested in Table 4, while the remaining features are the same as in the log-linear model of Table 3. As before, the scores of the 5-best column are maximized over the 5 top-ranked candidates.

8 Summary

References

Abney, S. (1996). Statistical methods and linguistics. In J. Klavans & P. Resnik (Eds.), *The Balancing Act: Combining Symbolic and Statistical Approaches to Language.* Cambridge, MA: MIT Press.

Abney, S. P. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, *23*, 597 – 618.

Bangalore, S., & Rambow, O. (2000). Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics.* Saarbrücken, Germany.

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

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

Forsbom, E. (2003). Training a super model look-alike: Featuring edit distance, n-gram occurrence, and one reference translation. In *Proceedings of the workshop on machine translation evaluation: Towards systemizing MT evaluation, held in conjunction with MT Summit IX*. New Orleans, USA.

Habash, N. (2004). The use of a structural n-gram language model in generation-heavyhybrid machine translation. In A. Belz, R. Evans, & P. Piwek (Eds.), *Proceedings of Biennial Meeting of the ACL Special Interest Group on Natural Language Generation* (pp. 61 – 69). Brockenhurst, UK: Springer.

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.

Knight, K., & Hatzivassiloglou, V. (1995). Two-level, many paths generation. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics.* Cambridge, MA.

Langkilde, I. (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of Biennial Meeting of the ACL Special Interest Group on Natural Language Generation.* New York, USA.

Langkilde, I., & Knight, K. (1998a). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics.* Montreal, Canada.

Langkilde, I., & Knight, K. (1998b). The practical value of n-grams in generation. In *Proceedings of Biennial Meeting of the ACL Special Interest Group on Natural Language Generation* (pp. 248 – 255). Ontario, Canada.

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

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

Miyao, Y., & Tsujii, J. (2002). Maximum entropy estimation for feature forests. In *Proceedings of the Human Language Technology Conference.* San Diego, CA.

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 2nd Human Language Technology Conference of the North American Chapter of the ACL.* Boston, MA.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., & Brants, T. (2002). The LinGO Redwoods treebank. Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics.* Taipei, Taiwan.

Osborne, M. (2000). Estimation of stochastic attribute-value grammars using an informative sample. In *Proceedings of the 18th International Conference on Computational Linguistics.* Saarbrücken, Germany.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu. A method for automatic evaluation of Machine Translation. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics* (pp. 311 – 318). Philadelphia.

Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. In *Proceedings of the 1st Conference of the North American Chapter of the ACL* (pp. 194 – 201). Seattle, Washington.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell III, J. T., & Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics.* Philadelphia, PA.

Toutanova, K., Manning, C. D., Flickinger, D., & Oepen, S. (2005). Stochastic HPSG parse selection using the Redwoods corpus. *Journal of Research on Language and Computation*, *3*(1), 83 – 105.

Velldal, E. (2008). *Empirical realization ranking.* Unpublished doctoral dissertation, University of Oslo, Institute of Informatics, Oslo.

Velldal, E., & Oepen, S. (2005). Maximum entropy models for realization ranking. In *Proceedings of the 10th Machine Translation Summit* (pp. 109 – 116). Phuket, Thailand.

Velldal, E., Oepen, S., & Flickinger, D. (2004). Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories* (pp. 149 – 160). Tübingen, Germany.

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