Paraphrasing Treebanks for Stochastic Realization Ranking

Erik Velldal♣, Stephan Oepen♣♠, Dan Flickinger♠

♣ Department of Linguistics, University of Oslo (Norway)
♠ Center for the Study of Language and Information, Stanford (USA)

1 Introduction

This paper[1] describes a novel approach to the task of *realization ranking*, i.e. the choice among competing paraphrases for a given input semantics, as produced by a generation system. We also introduce a notion of *symmetric treebanks*, which we define as the combination of (a) a set of pairings of surface forms and associated semantics *plus* (b) the sets of alternative analyses for the surface form and sets of alternate realizations of the semantics. For inclusion of alternate analyses and realizations in the symmetric treebank, we propose to make the underlying linguistic theory *explicit* and *operational*, viz. in the form of a broad-coverage computational grammar. Extending earlier work on grammar-based treebanks in the Redwoods (Oepen et al. [13]) paradigm, we present a fully automated procedure to produce a symmetric treebank from existing resources. To evaluate the utility of an initial (albeit smallish) such 'expanded' treebank, we report on experimental results for training stochastic discriminative models for the realization ranking task.

Our work is set within the context of a Norwegian–English machine translation project (LOGON; Oepen et al. [11]). The LOGON system builds on a relatively conventional semantic transfer architecture—based on Minimal Recursion Semantics (MRS; Copestake et al. [5])—and quite generally aims to combine a 'deep' linguistic backbone with stochastic processes for ambiguity management and improved robustness. In this paper we focus on the isolated subtask of ranking the output of the target language generator.

For target language realization, LOGON uses the LinGO English Resource Grammar (ERG; Flickinger [6]) and LKB generator, a lexically-driven chart generator that accepts MRS-style input semantics (Carroll et al. [2]). Over a representative LOGON data set, the generator already produces an average of 45 English realizations per input MRS; see Figure 1 for an example. As we expect to move to

[1]We would like to thank Mark Johnson (Brown University) and Rob Malouf (San Diego State University) for many fruitful discussions and their comments on earlier drafts and ideas presented in this paper.

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

if you come with the morning boat you can start the trip the same day.
if you come with the morning boat the trip you can start the same day.
if you come with the morning boat the same day you can start the trip.
the trip you can start the same day if you come with the morning boat.
you can start the trip the same day if you come with the morning boat.
the same day you can start the trip if you come with the morning boat.

Figure 1: Example sets of generator outputs using the LinGO ERG. Unless the input semantics is specified for aspects of information structure (e.g. requresting 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.

generation from packed, ambiguous transfer outputs, the degree of generator ambiguity will further increase. It is therefore essential for end-to-end MT to have a scalable means of ranking generator outputs and ultimately selecting one (or a few) preferred realizations.

In this paper we explore the use of discriminative log-linear models, or maximum entropy models, for ranking the realizations and propose an extended and symmetric notion of treebanks for the supervised learning task. This means that we treat the optimality relation encoded in each treebanked $\langle \textit{utterance}, \textit{analysis} \rangle$ pair as being *bidirectional*, and use the underlying grammar to generate all of their possible paraphrases. This provides us with all the admissible realizations for a set of input semantics, each accompanied with an indication of the preferred candidate(s).[2]

The next section further elaborates on the problem of realization ranking as well as the issue of symmetrizing and extending the treebank data. In section 4 we describe the log-linear models that are trained using structural features of the paraphrase data. We also compare the performance of the log-linear models to that of a simple n-gram language model, as well as to a hybrid model that combines the two. The experiments using the language models are described in section 3.

[2]The utility of this kind of resource is by no means restricted to our MT setting, but should prove relevant for other applications that generates from semantic representations. Furthermore, the ability to generate paraphrases of a given input seems potentially beneficial to other tasks too, as, for example, question-answering (QA) and summarization systems.

2 Bidirectionality of Treebank Data

Our perspective on the task of realization ranking is given by recognizing its similarity to the task of *parse selection*. As described below, selecting among the analyses delivered by a parser can be seen as the inverse of task of the realization ranking. The results reported by Oepen et al. [13] on the construction of the HPSG Redwoods treebanks and associated parse selection models provides us with a starting point in this respect, both in terms of the methodology used and available data sets.

When training a model for the task of parse selection (i.e. choosing among competing analyses of a token utterance), the distribution that one is typically interested in is the conditional probability of an analysis given a string. Moreover, this typically requires training data that consists of all possible analyses for a set of strings, where the goal is to estimate a distribution that for each string maximizes the probability of the preferred analysis over all the other competing candidates.

For the task of realization ranking (i.e. choosing among multiple paraphrases of a meaning representation input to the generator), on the other hand, we are interested in a different distribution. In order to select the best realization(s) we need a model that gives us the probability of a string given its semantics.

A treebank is traditionally conceived as a set of utterances (typically strings) paired with their optimal or correct analyses. In this paper we take the optimality relation that these pairs encode to be *bidirectional* or *symmetric*, in the sense that the original utterance can also be treated as an optimal realization of the corresponding semantic analysis (i.e. 'meaning'). The remaining part of this section looks at how we can exploit this bidirectionality or symmetry of the recorded $\langle utterance, analysis \rangle$ pairs to construct all the possible paraphrases for the treebanked items. This will provide us with the necessary training data to learn the discriminative models described in section 4.

The Redwoods treebank[3] is a collection of HPSG analyses derived from the LinGO ERG for various domain corpora (e.g. transcribed scheduling dialogues, ecommerce email, and lately tourism text), with manual annotation to identify the intended parse(s) for each utterance. Since (a) the Redwoods treebank provides a full HPSG sign for each input item and (b) the ERG analyses incorporate an MRS-style semantic component into HPSG, we have the option of using the semantics associated with each preferred analysis for subsequent generation.

Note that the Redwoods approach to treebanking—viz. the construction of the treebank by virtue of selecting among the analyses provided by a broad-coverage computational grammar—already includes alternate ('competing' but dis-preferred) analyses for a token utterance as first class data. While this extension to a conventional conception of treebanks as only providing the 'optimal' $\langle utterance, analysis \rangle$ pairs clearly benefits stochastic parse selection research, it would seem possible in

[3]See '`http://redwoods.stanford.edu/`' for further information on the Redwoods initiative and access to the data available to date. The Redwoods treebank is available under an open-source license and currently comprises some 15,000 annotated utterances.

theory at least that statistical parsing work—aiming to induce a grammar from the treebanked analyses, rather than using the ERG—could derive value from the additional data. Likewise, viewing a treebank as a repository of linguistic information, the availability of dis-preferred analyses might turn out useful to researchers in (formal) grammar or linguistic students. The proposal of the current paper is to further augment the treebank by the inverse correspondence: with regard to the explicit linguistic model underlying the treebank (i.e. the grammar used to build it), the paraphrase step aims to provide the mirror image of the dis-preferred analyses, this time making alternative but sub-optimal realizations first class data to be included in the treebank.

The actual procedure is straightforward. Given a Redwoods treebank, for each analysis that has been manually marked as the correct reading, we exhaustively generate all possible realizations for its semantics. In other words, for each string (and its hand-annotated intended meaning) in the original treebank, all semantically equivalent paraphrases admitted by the grammar are generated.

The next step is to automatically identify and mark the preferred realization(s). This is done by matching the *yields* of the generated trees against the original *strings* in the parse treebank, where all derivations yielding this preferred surface string are counted as equally good candidates. We now have a data set that includes all possible paraphrases for each treebanked semantic representation, with the best realization(s) marked. Note that, the *grammaticality* of all the candidates is guaranteed by the generator with respect to the input grammar. Furthermore, the fact that the Redwoods approach provides a treebank that is built on a grammar (and not the other way around) means that our data can be dynamically updated to reflect developments and improvements as the grammar is revised, i.e. as the grammar evolves there is a semi-automated procedure to (re-)synchronize the treebank with a new set of analyses provided by the grammar (see Oepen et al. [12] for details).

As mentioned initially, the strategy described here for utilizing treebanks comes with the underlying assumption that, without introducing too much distortion, the original string associated with a given reference analysis (in the 'parse treebank'), can also reasonably be taken to be an optimal way of expressing the corresponding meaning. After all, by the fact that a sentence is observed to occur naturally in our semantically annotated corpus (i.e. treebank), we are in some sense already making the assumption that a presumably rational and competent language user chose this very utterance to express the given semantics. Granting real language users some authority when it comes to formulating expressions that in an effective and natural-sounding way convey the meaning that they want to communicate, this does not seem like a too radical proposition.

2.1 Data and Evaluation

In the following we report on a preliminary investigation into the utility of such a set of paraphrases in a symmetric treebank for some 300 sentences from the LOGON domain—hiking instructions in Norway. The (relatively small) symmetric

Aggregate	**items** $\sharp$	**words** ϕ	**readings** ϕ
$100 \leq$ *readings*	19	19.7	422.9
$50 \leq$ *readings* < 100	17	17.8	71.7
$10 \leq$ *readings* < 50	72	13.7	22.6
$1 <$ *readings* < 10	153	10.4	4.8
Total	**261**	**12.4**	**44.5**

Table 1: Some core metrics for the symmetric treebank data used in our initial experiments, broken down by degrees of ambiguity in generation. 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), average string length, and average structural ambiguity.

treebank data that we had available for these initial experiments is summarized in Table 1. Although the total number of items in the treebank is above 300, when reporting results for our realization ranking experiments we exclude items that are unambiguous in generation and hence do not present a realization ranking problem.

Before we in section 4 go into the details of using log-linear models trained using structural features of the paraphrased treebank data, section 3 reports on some experiments that take a purely surface-oriented approach to the ranking task. The simple n-gram model presented here will not only serve as a baseline for experimental results obtained for the initial log-linear model, but will also be incorporated as one of the features in a final combined model.

All models are evaluated according to two different measures; exact match accuracy and the similarity-based BLEU score (Papineni et al. [15]). The exact match measure simply counts the number of top-ranked sentences, according to some model, that exactly matches a corresponding "gold" or reference sentence. In other words, after a model has been applied to all possible paraphrases in the symmetric treebank, we count the number of times that the model assigns the best score to (one of) the string(s) marked as preferred in the symmetric treebank. The similarity-based and less rigid BLEU measure has gained a well-established role as an evaluation metric in MT, and is modeled after the *word error rate* measure used in speech recognition. The score is computed as a weighted average of the n-gram precision of the selected candidate realization with respect to the reference, for all $1 \leq n \leq 4$. When evaluating a models performance on the test data, we report the averaged BLEU scores over all realizations ranked best by the model, which has a constant range in $[0, 1]$. Although it may be hard to intuitively interpret this precision-based measure in isolation, it at least offers an alternative view when comparing the relative performance of the various models that we now turn to describe. For more information on the BLEU scoring metric, see Papineni et al. [15].

3 N-Gram Language Models

As a first shot at ranking the generator outputs, we order the English target strings with respect to the probabilities assigned by a simple n-gram language model.

An n-gram model relies on the Markov assumption that the probability of a given word only depends on the $n-1$ words preceding it, and so the probability of a sequence $(w_1, \ldots, w_k)$ is computed as

$$p_n(w_1, \ldots, w_k) = \prod_{i=1}^{k} p(w_i | w_{i-n}, \ldots, w_{i-1}) \tag{1}$$

In MT applications, the idea of choosing the most fluent string as the best translation is a commonly used technique (see, among others, Langkilde and Knight [8], and Callison-Burch and Flournoy [1]). Using the CMU-SLM Toolkit (Clarkson and Rosenfeld [4]), various n-gram language models were trained on a plain (unannotated) text version of the British National Corpus (BNC), containing approximately 100 million words.

For the experiments reported in this paper, we use 4-gram model trained with Witten-Bell discounting, a vocabulary of 65,000 words, sentence boundary context cues, and using back-off to lower-order models for unobserved n-grams. When applying the language model to the ranking task we obtain close to fifty per cent exact match accuracy (see Table 2). This result improves significantly over a seventeen per cent random choice baseline. The same holds for the similarity-based evaluation in Table 3.

We also tried ranking the realizations by their cross-entropy or perplexity with respect to the language model. Of course, this setting is quite far from the typical model evaluation setting in which perplexity scores are computed (as an asymptotic approximation) over an entire test corpus for assessing the quality of a model. On the sentence level the usual approximation to the perplexity essentially indicates the average log probability of the words:

$$2^{-\frac{1}{k}\log p_n(w_1,\ldots,w_k)} \tag{2}$$

Using these scores, however, gave somewhat inferior performance compared to using the (negative log) probabilities directly. Furthermore, we also saw that increasing the value of n, as well as increasing the vocabulary size, always lead to better performance in our ranking task, although at the expense of larger models.

The basic underlying assumption of the approach described in this section, is that the best realization of the input semantics corresponds to the most fluent string. This implies that we rank outputs as isolated strings rather than as realizations of a given semantic representation. Another obvious limitation inherent to the simple n-gram approach described here is the fact that it cannot capture long-range dependencies.

4 Log-Linear Models

Taking inspiration from contemporary parse selection work, we here describe conditional log-linear models that take into account structural features of competing realizations for a given input MRS. The family of maximum entropy models or log-linear models provides a general framework that allows one to combine disparate and overlapping sources of information in a single model without making unwarranted independence assumptions. A model is given in terms of *specified feature functions* describing the data points, and an associated set of *learned weights* that determine the contribution or importance of each feature. Each event—in our case a realization $r \in \Omega$—is mapped to a feature vector $f(r) \in \Re^d$, and a vector of weights $\lambda \in \Re^d$ is then fitted to optimize some objective function. A conditional log-linear model for the probability of a realization r given the semantics s, has the form

$$p_\lambda(r|s) = \frac{1}{Z(s)} \exp(\lambda \cdot f(r)) \tag{3}$$

where $Z(s)$ is a normalization factor defined as

$$Z(s) = \sum_{r' \in Y(s)} \exp(\lambda \cdot f(r')) \tag{4}$$

When computing the so-called partition function $Z(s)$ as in equation (4) above, $Y(s)$ gives the set of all possible realizations of s. The weight vector λ is chosen as to maximize the (log of) a penalized likelihood function as in

$$\hat{\lambda} = \arg\max_{\lambda} \log L(\lambda) - \frac{\sum_{i=1}^{d} \lambda_i^2}{2\sigma^2} \tag{5}$$

where $L(\lambda)$ is the pseudo-likelihood of the training data (as described by Johnson et al. [7]), computed as

$$L(\lambda) = \prod_{i=1}^{N} p_\lambda(r_i|s_i) \tag{6}$$

In accordance with current best practice, the second term of the objective function in (5) defines a zero mean Gaussian prior on the weight parameters (Chen and Rosenfeld [3], Johnson et al. [7], Malouf and van Noord [10]). By promoting less extreme parameter values this penalty term can reduce the tendency of log-linear models to over-fit the training data. In addition to improving accuracy, this kind of smoothing tends to also reduce the number of iterations needed for convergence during estimation (Malouf and van Noord [10]). We empirically determined a suitable value for the variance σ^2 which is uniformly set to 100 for the results reported here. Note that the value of the variance parameter determines the relative

Aggregate	**random** %	n**-gram** %	**MaxEnt** %	**combined** %
$100 \leq$ *readings*	0.4	10.5	21.1	31.6
$50 \leq$ *readings* < 100	1.5	17.7	22.1	29.4
$10 \leq$ *readings* < 50	5.4	38.2	31.6	37.5
$1 <$ *readings* < 10	27.0	62.8	68.2	75.8
Total	**17.2**	**49.2**	**51.7**	**59.0**

Table 2: Realization ranking accuracies for a random-choice baseline model, 4-gram language model, simple conditional model, and combination of the two. The columns are, from left to right, the subdivision of the data according to degrees of ambiguity, followed by exact match accuracies for the four models.

Aggregate	**random** %	n**-gram** %	**MaxEnt** %	**combined** %
$100 \leq$ *readings*	0.5869	0.7209	0.7023	0.7534
$50 \leq$ *readings* < 100	0.5956	0.7751	0.7939	0.7790
$10 \leq$ *readings* < 50	0.6418	0.7985	0.8293	0.8172
$1 <$ *readings* < 10	0.7382	0.8792	0.9178	0.9316
Total	**0.6913**	**0.8386**	**0.8696**	**0.8771**

Table 3: Realization similarity measures for a random-choice baseline model, 4-gram language model, simple conditional model, and combination of the two. The columns are, from left to right, the subdivision of the data according to degrees of ambiguity, followed by averaged BLEU scores of the realization(s) ranked best by each of the four models.

contribution of the prior and the likelihood function, and thereby the degree of smoothing (Malouf and van Noord [10]).

For the parse selection task, Toutanova and Manning [16] train a discriminative log-linear model with features defined over ERG *derivation trees*, where labels identify specific *construction types* and fine-grained *lexical classes*. For our own initial experiments with the realization ranking we define the feature set in the same way (the basic PCFG-S model of Toutanova and Manning [16]), using the `estimate` open-source package (Malouf [9]) for parameter estimation (using the *limited-memory variable metric*). With only around 300 training sentences in our current generation treebank and ten-fold cross validation (which tends to underestimate model performance), this simplest of log-linear models performs competitively to the language model trained on the BNC (see Tables 2 and 3).

As a third model we augmented the log-linear model with an extra feature corresponding to the sentence probabilities of the language model (described in section 3). The value of the $d + 1$'th feature is the (negative log) probability of the string as given by the n-gram model p_n, i.e. $f_{d+1}(r) = -log\ p_n(y(r))$, where $y(r)$ is the yield of r and $n = 4$ as before. Unsurprisingly, the combined model significantly outperforms both the previously described models.

5 Discussion and Outlook

For the relatively coherent LOGON domain at least, a tiny training set of some 300 automatically 'annotated' paraphrases combined with a discriminative baseline model originally proposed for the parse selection task outperforms a language model trained on all of the BNC. Our results suggest that this use of domain-specific treebanks—and the underlying assumption of relative 'naturalness' of the original, corpus-attested realizations—provide a good handle on ranking generator outputs, and that structural, linguistic information as is available to the log-linear model is of central importance for this task. We are currently extending the size of the available treebank for the LOGON domain (to some 1,500 utterances initially and ultimately to at least 5,000 annotated items) and expect that the larger training set—combined with more systematic experimentation with discriminative models using larger and more specialized feature sets—should allow us to improve exact match accuracy significantly, ideally to around eighty per cent exact match as are the currently best available parse selection results.

Additionally, we plan to formalize a notion of graded acceptability of competing realizations (based on string similarity metrics, e.g. BLEU or string kernels) and refine both model training and evaluation in this respect. Unlike in parse selection—where distinct system outputs typically have distinct semantics—in realization ranking there is more of a graded continuum of more or less natural verbalizations (given available information). All outputs are guaranteed by the grammar to be semantically equivalent and grammatically well-formed. This means that the kind properties we aim at capturing with the discriminative model rather are soft constraints that govern the graded degree preference among the competing paraphrases. The approach described by Osborne [14] and Malouf and van Noord [10] for scoring the training instances (parses in both cases) according to some measure of preference, and defining the empirical distributions based on these weights, seems like a well-suited approach for dealing with generation outputs too, where the notion of correctness may be inherently fleeting.

To investigate the degree of domain-specificity in stochastic models derived from Redwoods-style symmetric treebanks, we plan to automatically paraphrase additional segments of the available Redwoods treebank and perform cross-domain realization ranking experiments.

References

[1] Chris Callison-Burch and Raymond S. Flournoy. A program for automatically selecting the best output from multiple machine translation engines. In *Proceedings of the MT Summit*, Santiago, Spain, September 2001.

[2] John Carroll, Ann Copestake, Daniel Flickinger, and Victor Poznanski. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86 – 95, Toulouse, France, 1999.

[3] Stanley F. Chen and Ronald Rosenfeld. A Gaussian prior for smoothing maximum entropy models, 1999. Technical Report CMUCS-CS-99-108.

[4] Philip Clarkson and Roni Rosenfeld. Statistical language modeling using the CMU-Cambridge Toolkit. In *Proceedings of ESCA Eurospeech*, 1997.

[5] Ann Copestake, Dan Flickinger, Rob Malouf, Susanne Riehemann, and Ivan Sag. Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium, 1995.

[6] Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28, 2000.

[7] Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic 'unification-based' grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 535 – 541, College Park, MD, 1999.

[8] Irene Langkilde and Kevin Knight. The practical value of n-grams in generation. In *International Natural Language Generation Workshop*, 1998.

[9] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan, 2002.

[10] Robert Malouf and Gertjan van Noord. Wide coverage parsing with stochastic attribute value grammars. In *Proceedings of the IJCNLP workshop Beyond Shallow Analysis*, Hainan, China, 2004.

[11] Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. Som å kappete med trollet? Towards MRS-based Norwegian – English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD, 2004.

[12] Stephan Oepen, Dan Flickinger, and Francis Bond. Towards holistic grammar engineering and testing. Grafting treebank maintenance into the grammar revision cycle. In *Proceedings of the IJCNLP workshop Beyond Shallow Analysis*, Hainan, China, 2004.

[13] Stephan Oepen, Kristina Toutanova, Stuart Shieber, Chris Manning, Dan Flickinger, and Thorsten Brants. The LinGO Redwoods treebank. Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

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

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu. A method for automatic evaluation of Machine Translation. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA, 2002.

[16] Kristina Toutanova and Christopher D. Manning. Feature selection for a rich HPSG grammar using decision trees. In *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan, 2002.