Is Bad Structure Better Than No Structure?: Unsupervised Parsing for Realisation Ranking

Yasaman MOTAZEDI Mark DRAS François LAREAU
Centre for Language Technology, Macquarie University
Yasaman.Motazedi@students.mq.edu.au, Mark.Dras@mq.edu.edu,
Francois.Lareau@mq.edu.au

Abstract
In natural language generation using symbolic grammars, state-of-the-art realisation rankers use statistical models incorporating both language model and structural features. The rankers depend on multiple structures produced by the particular large-scale symbolic grammars to rank the output; for languages with smaller resources and in-development grammars, we look at the feasibility of an alternative source of structural features, unsupervised parsers. We show that, in spite of their lower quality of structure, raw sets of unsupervised parse features can be helpful with smaller language models; and that the parses do contain particular elements that can be highly useful, improving performance on our classification task by up to 10% on 60% of the test set leading to an overall improvement under a back-off model.

Title and Abstract in French

Une mauvaise structure est-elle mieux que pas de structure du tout? L'analyse non supervisée pour la sélection des réalisations

Dans plusieurs systèmes récents de génération de texte basés sur des grammaires symboliques, les résultats sont ordonnés selon leur acceptabilité par des modèles statistiques qui incorporent des modèles de Markov et des traits structurels. Ces modules d'ordonnancement dépendent de diverses structures produites par la grammaire, ce qui présuppose une grammaire suffisamment développée. Pour les langues à faibles ressources ou pour les grammaires en cours de développement, nous étudions ici la viabilité d'une source alternative de traits structurels: les analyseurs non supervisés. Nous démontrons que, en dépit de la faible qualité des structures produites, elles contiennent des éléments qui peuvent être très utiles pour les langues peu dotées, permettant d'améliorer de 10% la performance de notre classificateur pour 60% des phrases de notre corpus de test.

Keywords: natural language generation, realization ranking, unsupervised parsing.

Keywords in French: génération de langue naturelle, analyseur statistique non supervisé.

Proceedings of COLING 2012: Technical Papers, pages 1811–1830,
COLING 2012, Mumbai, December 2012.

1 Condensed 2-page version in French

Les systèmes de génération de textes peuvent produire plusieurs textes de qualité variable pour les mêmes données. Cela a mené à la création d'algorithmes de sélection pour choisir le meilleur texte (Langkilde-Geary, 2000; Velldal and Oepen, 2005; Cahill et al., 2007, par exemple). Charniak (2001) a proposé d'utiliser les analyseurs statistiques comme source de structures sur lesquelles baser les algorithmes de sélection. À notre connaissance, seuls des analyseurs supervisés ont été utilisés à cet effet. Cependant, pour les systèmes basés sur des grammaires symboliques de modeste envergure ou en développement, trop peu de données sont disponibles, et les modèles existants s'appliquent mal. Dans cet article, nous étudions donc la possibilité d'utiliser un analyseur statistique non supervisé (Naseem et al., 2010) comme source alternative d'information pour la sélection automatique des textes générés, malgré la faible qualité des structures qu'ils produisent.

Nos travaux sont basés notamment sur ceux de Cahill et al. (2007), qui à la suite de Velldal and Oepen (2006), ont développé un modèle log-linéaire de réalisation pour une grammaire Lexical Functional Grammar (LFG). Comme eux, nous utilisons la plateforme Xerox Linguistics Environment (XLE) et construisons une banque d'arbres symétrique en analysant puis en regénérant les phrases. Ceci nous fournit à la fois des exemples positifs et négatifs. Nous utilisons les sections 2 à 21 du Penn Treebank (38 008 phrases) pour l'entraînement et la section 23 (2245 phrases) pour l'évaluation. Nous les avons analysées en utilisant la grammaire ParGram de l'anglais sur XLE (Butt et al., 2002), puis nous les avons regénérées en renversant la même grammaire pour produire de multiples paraphrases candidates. Nous avons rejeté les cas où la grammaire ne regénérait pas plus d'une phrase, ou plus de 1000, pour obtenir un corpus d'entraînement de 20 613 ensembles de paraphrases et un corpus de test de 1168 ensembles de paraphrases. Souvent, la grammaire n'arrivait pas à reproduire la phrase originale. Nous avons donc calculé la distance d'édition pour chaque paraphrase. Nous utilisons comme cas positif pour l'entraînement et pour le test la phrase regénérée ayant la plus petite distance par rapport à la phrase originale. Comme exemple négatif, nous avons comparé deux alternatives : la phrase ayant la plus grande distance (greatest, dans nos tableaux), et une phrase choisie au hasard parmi celles qui n'avaient pas la plus petite distance (random, dans nos tableaux).

Pour la classification, nous avons utilisé le système à entropie maximale MegaM (cinquième version) de Hal Daumé III. Pour fins de comparaison, nous avons reproduit la même méthode avec deux analyseurs supervisés : celui de Stanford (Klein and Manning, 2003) et celui de Charniak et Johnson (2005) (ci-après, C&J). Nous couplons les analyseurs à des modèles de Markov : un grand, construit avec SRILM (Stolcke, 2002) sur le corpus Gigaword, et un petit, afin d'imiter les conditions de développement pour des petites grammaires. Nous utilisons ces modèles de Markov à la fois comme modèles de base et en combinaison avec les modèles structurels. En outre, parce que le principal obstacle à l'utilisation d'un analyseur non supervisé est la faible qualité des analyses qu'il produit (voir Figure 1 pour une analyse non supervisée, par opposition à l'analyse supervisée de la Figure 2). Nous avons comparé deux approches pour identifier les dépendances les plus fiables à utiliser comme traits : le calcul de précision des dépendances individuelles, et la méthode de gain d'information.

Les résultats de notre expérience montrent que, sans surprise, les analyseurs non supervisés donnent des résultats nettement moins bons que les analyseurs supervisés. Cependant, ils sont utiles s'ils sont couplés à un modèle de Markov, même de taille modeste. L'utilisation de traits d'analyse seuls donne quand même des résultats meilleurs qu'un simple modèle de Markov. En conclusion, donc, les analyseurs non supervisés peuvent être utiles. En général, les traits qu'ils fournissent contribuent à améliorer les résultats obtenus avec un modèle de Markov de taille modeste, du type qu'on trouve normalement pour les langues peu dotées. Ils contribuent aussi très fortement à améliorer

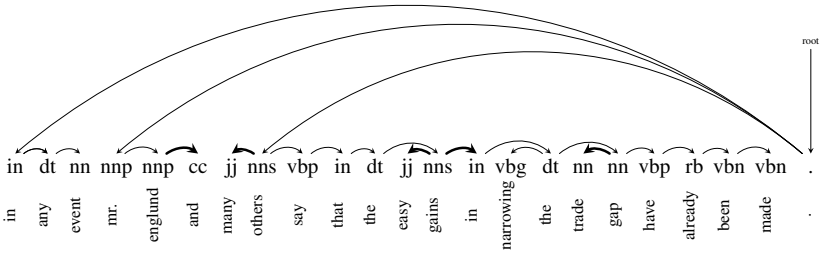

Figure 1 – Unsupervised dependency tree for a sample sentence.//Analyse non supervisée pour une phrase.

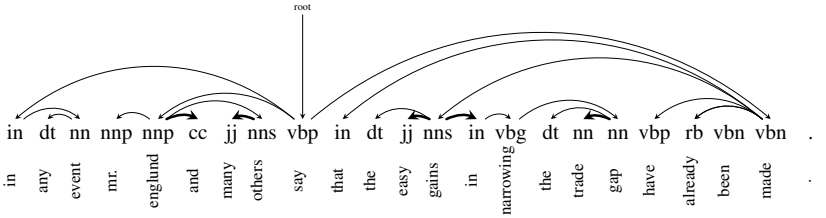

Figure 2 – Supervised dependency tree for sample sentence of Fig 1.//Analyse supervisée pour la même phrase qu'à la Figure 1.

les résultats pour un échantillon respectable du corpus de test, une fois que les traits utiles ont été identifiés par la méthode de gain d'information : nous avons observés des gains de précision de 10% dans 60% des phrases du corpus de test, ce qui rend possible l'utilisation d'un modèle de repli.

2 Introduction

Natural Language Generation (NLG) systems take some machine-oriented input such as databases, knowledge bases or logical forms and produce human-like text. The realization involves making many decisions on the surface representation of a sentence, including lexical selection, use of referring expressions, and word order. In general, such systems can thus generate multiple outputs, some of which are genuine good paraphrases, some of which may sound somewhat marked or odd out of context, and some of which may be (due to system limitations) incorrect.

This observation led to the idea of taking the possible outputs and ranking them, in the earliest work by use of a language model (LM) (Langkilde-Geary, 2000) and later by injecting new features like headwords, constituent category and Part of Speech (POS) tags (Langkilde-Geary, 2002). This idea was developed further for handcrafted symbolic grammars based on linguistic formalisms that have been used for generation. In the Head-driven Phrase Structure Grammar (HPSG) system LOGON, Velldal and Oepen (2005) improve on this by using a maximum entropy model incorporating both a LM and features derived from HPSG structures for the candidate outputs; this drew on earlier developments in the field of statistical parsing that used such features for reranking (Johnson et al., 1999; Riezler et al., 2002, for example). Cahill et al. (2007) and White and Rajkumar (2009) similarly show that ranking for Lexical Functional Grammar (LFG)- and Combinatory Categorial Grammar (CCG)-based systems respectively can be improved by models incorporating a LM and structural features.

All of these rankers depend on multiple structures produced by the respective large-scale symbolic grammars to rank the output. For much smaller symbolic grammars, and those in the process of development — e.g. LFG grammars for Indonesian (Arka et al., 2009) or Arrernte (Dras et al., 2012), or HPSG grammars for Persian (Müller, 2010) or Wambaya (Bender, 2008) — these multiple structures will not be available or will be quite impoverished. The goal of this paper then is to investigate using other sources of structural information that will be available for such languages: specifically — since such languages are also often unlikely to have treebanks — unsupervised statistical parsers. Unsupervised parsers have been motivated by possible application in contexts where large treebanks for training supervised parsers are absent (Klein and Manning, 2004, for example), but given their very modest performance with respect to supervised alternatives it is an open question as to whether the structures they produce have any extrinsic application. We aim to give an answer to this question in the context of realisation ranking.

In Section 3 we look at some previous work on reranking in realization in more depth, and more briefly note work on parser-based models for ranking and on unsupervised parsing. In Section 4 we describe our models based on statistical parsers, and the experimental set-up for investigating them. In Section 5 we discuss our results, and conclude in Section 6.

3 Related Works

Reranking in Realization Using statistical models that can combine a wide range of features to rank candidate outputs is standard across many applications, such as parse selection and Machine Translation (MT). They have also been applied to realisation ranking in NLG, to select the best of a set of candidate generated sentences. Here we review some of these, with a focus on the work of Cahill et al. (2007), as this is the setup closest to the one we use in this paper.

Velldal and Oepen (2006) implemented realisation ranking in the context of an MT system that uses a hand-crafted HPSG grammar to generate sentences from semantic specifications. Their system produced multiple candidate sentences which were ranked under three different models: an n-gram

LM; a discriminative maximum entropy model using (HPSG) structural features; and a combination of the two, which produced the best results.

To provide training data for the model, they constructed a 'symmetric treebank' (Velldal et al., 2004) composed of:

1. a set of pairings of surface forms and associated semantics;
2. a set of alternative analyses for each surface form; and
3. a set of alternative realisations of each semantic form.

The preferred realisations are automatically specified by comparing the yields of the generated sentences with the original strings in the treebank (for them, the Redwoods English treebank); this gives the sets of positive and negative (i.e. all non-original) examples for the maxent learner.

Cahill et al. (2007) drew on this to develop a log-linear model for realisation in an LFG context. They worked with a large-scale hand-crafted grammar of German, as they were investigating applicability of the approach to a language with freer word order than English, and generated from f-structures using the XLE system (Maxwell and Kaplan, 1993). They similarly constructed a symmetric treebank, and also had as their goal to re-generate the strings in the original treebank.

For structural features in their model, the authors defined 186,731 instantiated templates based on Riezler et al. (2002), Rohrer and Forst (2006) and Riezler and Vasserman (2004); 1,471 of these actually occurred in their training data. The feature templates included information from both c-structure (e.g. simple features such as number of times a particular category label occurs, or compound features such as number of times it dominates another) and f-structure (e.g. frequency of relative order of subject and object), sentence length and LM scores. Structural features here contributed more strongly than for English.

Which structural features are effective in realisation ranking is still an open question. For example, in the context of CCG, Rajkumar and White (2010) found that features reflecting animacy agreement between nouns and relative clauses, and number agreement between subject and verb, were helpful. White and Rajkumar (2012) found that explicitly representing dependency lengths led to shorter average dependencies, in line with psycholinguistic evidence for human-produced sentences, and to better generated text. And Filippova and Strube (2009) found that while trigram LMs are appropriate for phrase linearisation in German, at a clause level (longer) dependency features produced better results.

Parser-based Ranking Charniak (2001) proposed the idea of using statistical parsers as a kind of structural language model. The idea has been applied a number of times in MT, for example by Charniak et al. (2003) or Post and Gildea (2008); it has also been applied in NLG, where Mutton et al. (2007) showed that a combination of parser-based metrics correlated with human judgements of the quality of generated text. To our knowledge, all work applying parser-based ranking has used supervised parsers. There is also an interesting piece of work by Cherry and Quirk (2008) where implicit discriminative syntactic LMs are constructed, using a latent Support Vector Machine (SVM) to train an unlexicalised parser to judge sentences produced by an MT system. The authors discuss some similarities to unsupervised parsing, in that both sorts of parsers are trained on sentences without the benefit of annotated parse trees. Using unsupervised parse trees as we do in this paper, however, can benefit from lexicalisation and, potentially, linguistic knowledge embodied in the parser (see below).

Unsupervised Parsers The first unsupervised parser that convincingly beat fairly simple baselines, such as constructing a right-branching tree, was the DMV model of Klein and Manning (2004), which combined constituency and dependency parsing to produce a better unsupervised parser than either separately. There have been various developments in unsupervised parsing since then: Headden III et al. (2009) introduced basic valence frames and lexical information, along with a smoothing technique to handle resulting data sparsity; Berg-Kirkpatrick and Klein (2010) used a phylogeny-structured model of parameter drift; and Naseem et al. (2010) used a single set of manually specified language-independent rules characterising syntactic dependencies across languages, as a soft constraint during the inference of the probabilistic model. We use this last one in our work, as it gives state-of-the-art results, and embodies potentially useful hard-coded universal tendencies. Notwithstanding that state-of-the-art performance, directed dependency results for that system range from only 50.9% (Slovene) to 71.9% (English), and importantly, as in previous work, these cross-linguistic evaluations are only carried out on sentences of 10 or fewer words.

4 Experimental Setup

Our goal is to evaluate the usefulness of the structural information proposed by an unsupervised parser in assessing the quality of sentences generated by a symbolic grammar, given its noticeably lower quality than supervised parsers. Like Cahill et al. (2007), we use the XLE system and construct a symmetric treebank by parsing and re-generating sentences: this gives both positive examples (those that match the original sentence) and negative examples (those that don't). Details of the various aspects of this process follow.

4.1 Data and Evaluation

We took the Penn Treebank sections 2–21 for training, consisting of 38008 sentences, and section 23 for testing, consisting of 2245 sentences. We parsed them using XLE and the large-scale ParGram grammar of English (Butt et al., 2002), and then used XLE's re-generate facility to produce the multiple candidate realisations. As Cahill et al. (2007) did, we found that XLE could not parse and re-generate all sentences; and for the purposes of constructing a training set, only instances that re-generated more than one sentence were useful.[1] In addition, we excluded the few sentences that contained more than 1000 re-generated sentences. This resulted in a training set of 20613 (sets of) sentences and a test set of 1168 (sets of) sentences.

Also as Cahill et al. (2007) did, we found that not all sets of re-generated sentences contained the original sentence in its exact form: this was often because of small differences such as use of abbreviations (e.g. *Nov* for *November*) or use of punctuation. In contrast to their approach, where they discarded these cases, we used minimum edit distance (MED) to determine the re-generated candidate closest to the original.

Following this, we constructed training and test sets of pairs of re-generated sentences by taking as the positive example the one with smallest MED from the original sentence, and as the negative example one of two alternatives: either the one with largest MED (greatest) or a random selection from those candidates that did not have the smallest MED (random). greatest always gave the higher result in the classification experiments below, often by several percentage points (which is not surprising, as the differences are larger and the classification hence easier), so we mostly only report results for random.

[1] There is also a technical issue in re-generating numerals. To get around this problem, we preprocessed the text to change numerals to those that could be re-generated.

We used the maximum entropy learner MegaM (fifth release) by Hal Daumé III.[2] Our evaluation metric is the classification accuracy of predicting the positive example from each pair.

4.2 Parsers and Language Models

Before answering any questions about the usefulness of unsupervised parsers, we address the question, Do *supervised* parsers produce structural information that is useful for realisation ranking? If the answer is yes (and based on the work cited in Section 3 and others, we would think it likely), the supervised parsers and a large LM would constitute an upper bound on the efficacy of using parsers to rank candidate sentences generated by XLE. The supervised parsers we use are the Stanford parser (Klein and Manning, 2003) and the Charniak and Johnson (henceforth C&J) parser (Charniak and Johnson, 2005). These parsers are both quite accurate: the Stanford parser gets a labelled f-score of 85.61 on the WSJ, and the C&J 91.09.

From the Stanford parser we examined both horizontal slices of parse trees, in effect treating them as sets of CFG production rules, and dependencies; the production rules and dependencies were either lexicalised or unlexicalised, and the dependency relations either named or unnamed. This gives two constituency and four dependency feature representations from the Stanford parser: prod-rule-lex and prod-rule-unlex; and dep-lex-named, dep-lex-unnamed, dep-unlex-named and dep-unlex-unnamed.

C&J is a reranking parser; the reranker uses 13 feature schemas such as tuples covering head-to-head dependencies, preterminals together with their closest maximal projection ancestors, and subtrees rooted in the least common ancestor. We took two types of features from C&J: production rules; and the instantiated feature schemas from the parse reranking process, making them do 'double duty' as realisation ranking features. C&J is used as a complement to the Stanford parser here, with respect to production rules, as an easy way of getting something approximating the compound features used in realisation ranking discussed in Section 3.

The large LM was constructed using SRILM (Stolcke, 2002) on the Gigaword corpus.[3] We use the 64K 4-gram and the 64K 2-gram (where the size nK represents the use of the top n words occurring in the training text as vocabularies), which are at the two extremes of size. We refer to these as l-lm4 and l-lm2.

For the unsupervised structure that is the main interest of the paper, we used the parser of Naseem et al. (2010),[4] which constructs dependency trees. Relation names are not inferred, so the two alternative representations are lexicalised (unsuper-lex) and unlexicalised (unsuper-unlex).

For a small LM that would be of a realistic size for the scenario we are interested in — i.e. the development of a symbolic grammar for a language with few resources — we considered the size of corpora produced by *An Crúbadán*[5] (Scannell, 2007). This is a web crawler whose specific goal is the "automatic development of large text corpora for minority languages". As examples, it has produced corpora of ~2M words for Akan (Ghana), ~5M words for Tamil, and ~7M words for Turkmen. Consequently, we use the Penn Treebank (~4M words) as our realistically-sized LM. On this sized corpus, we derive both 4-gram and 2-gram LMs from SRILM using the default settings; we refer to these as s-lm4 and s-lm2.[6]

[2] MegaM software is available on `http://www.cs.utah.edu/~hal/megam/`.

[3] `http://www.keithv.com/software/giga/` was the source for these models. They used interpolated, modified Kneser-Ney smoothing, bigram cutoff 3, trigram cutoff 5.

[4] `http://groups.csail.mit.edu/rbg/code/dependency/`

[5] `http://borel.slu.edu/crubadan/`

[6] We note that these are in-domain LMs, in contrast to the Gigaword-derived ones, and so will have a bit of an advantage.

4.3 Models

Base models, supervised and unsupervised The basic models are then the structural models described above. We used the LMs both as baselines and in combination with the structural models. (Note that the C&J parser already includes a LM in its reranking feature templates that we use, so we do not combine in this case.) In the base cases, we use all structures (production rules, templates or dependencies as appropriate) returned by the relevant parser.[7]

For the unsupervised parser, the major issue is that the parses are generally of lower quality. As an illustration, Figures 1 and 2 represent unsupervised and supervised parses of the same sentence (common edges are indicated in bold). The supervised parse looks relatively reasonable, while the unsupervised parse makes some odd choices (*mr.* being a dependent of the root, the verb having no special status, etc), and contains many adjacent dependencies. Figures 3 and 4 represent another pair of parses, this one with a few longer dependencies than the previous pair of figures.

We take two approaches to finding higher quality individual dependencies for use as features: one is the calculation of fine-grained accuracy rates for dependencies, and the other is Information Gain (IG). In the case of fine-grained accuracy rates, we are assessing which dependencies are likely to be reliable, by comparison with a gold standard,[8] and in the case of IG, we identify which dependencies are particularly strongly associated with positive or negative examples.

Unsupervised with reliability-based selection To measure unsupervised dependency reliability, raw precision / recall scores would capture some of that notion — e.g. `VBD TO`, with 6593 gold-standard instances and 1520 correctly identified (recall = 0.231), is almost certainly more reliable than `NNP IN` with 7031 gold standard instances and only 547 correctly identified (recall = 0.078) — but they would obviously overrepresent low frequency dependency pairs, which are likely to be particularly unreliable. Taking the precision and recall scores as probabilities of correctness, we therefore adopted a prior α to smooth the scores. Given the relevant denominators N_p for precision or N_r for recall for each dependency pair type, our modified scores use $N_p + \alpha$ and $N_r + \alpha$ respectively; this leaves a probability mass $\frac{\alpha}{\alpha+N_p}$ (resp. $\frac{\alpha}{\alpha+N_r}$) for unseen instances of each dependency. By inspection of the actual raw scores on the training set, we chose $\alpha = 20$. We then select features with modified scores above various thresholds.

For lexicalised dependencies, with their much greater data sparsity issues, we based the choice on the corresponding unlexicalised dependency: if the parts of speech of the lexicalised dependencies matched an unlexicalised dependency above a particular threshold, we selected that lexicalised dependency. (So `the man` would deemed reliable if `DT NN` were deemed reliable.)

Unsupervised with IG selection We calculate IG over the training set, using a standard formulation of Yang and Pedersen (1997):

$$\begin{aligned} IG(r) = & -\sum_{i=1}^{m} \Pr(c_i) \log \Pr(c_i) \\ & + \Pr(r) \sum_{i=1}^{m} \Pr(c_i|r) \log \Pr(c_i|r) \\ & + \Pr(\bar{r}) \sum_{i=1}^{m} \Pr(c_i|\bar{r}) \log \Pr(c_i|\bar{r}) \end{aligned}$$

[7] In cases where the feature representation is the same for both positive and negative instances, we make a random choice.

[8] The Penn Treebank dependency gold standard was derived using `http://nlp.cs.lth.se/software/treebank_converter/`.

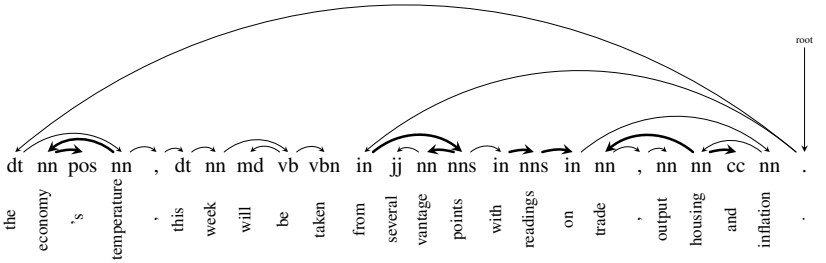

Figure 3: Unsupervised dependency tree for another sample sentence.

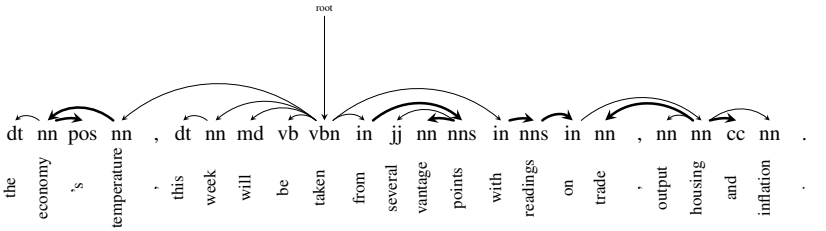

Figure 4: Supervised dependency tree for sample sentence of Fig 3.

with r representing a dependency, c a binary class, and $m = 2$. We then select features with IG scores above various thresholds. The application to unlexicalised features is straightforward: IG is applied to the unlexicalised dependencies, and some subset of these chosen based on the resulting ranking and a particular threshold. For lexicalised dependencies, there are two alternatives. One is to apply IG to the lexicalised dependencies directly (direct); the second is the same as the approach for unsupervised dependencies with reliability selection above, where we extract all the lexicalised dependencies that have corresponding selected unlexicalised dependencies (indirect).

Of course, both of these refinements of unsupervised parse features require some kind of annotation, and we discuss the implications for our scenario later in Section 5; but here we are just interested in the question of the extent to which unsupervied parsers can produce dependencies that are at all useful for realisation ranking.

5 Results

Base models (supervised) Table 1 gives classification accuracy results for the large LMs, and for the supervised parsers, both separately and in combination with the LM, all on the random test set. The key results here are:

1. Using parse features alone almost always outperforms the LM (except in the case of dep-unlex-

model	acc.%	+l-lm4	+l-lm2
l-lm4	68.36	-	-
l-lm2	64.16	-	-
C&J	90.50	-	-
prod-rule-lex	79.48	80.76	80.33
prod-rule-unlex	71.49	72.24	71.64
dep-lex-named	72.45	76.18	74.68
dep-lex-unnamed	71.51	75.54	74.72
dep-unlex-named	69.88	72.92	71.55
dep-unlex-unnamed	63.41	66.75	65.04

Table 1: Classification scores for supervised parse features and large LM on random: accuracy on parse features; parse features plus large l-lm4 LM over these sentences; parse features plus large l-lm2 LM over these sentences

unnamed), and the combination with the LM always improves over just parse features alone. This is broadly in line with the findings discussed in Sec 3 on using structural features from symbolic grammars, and suggests that using external statistical parsers is a valid alternative approach to carrying out realisation ranking.

2. While the l-lm4 LM forms quite a strong baseline, the l-lm2 version is somewhat weaker. It does still, however, contribute to an improvement in performance when added to the dependency feature models, of around 2%. This is not unexpected, as it would be contributing information that is complementary to the dependencies, which (in the cases where the dependencies are not adjacent) give longer-distance information. For the same reason, adding this bigram data to the production rule models produces a much smaller improvement.

3. Production rule features are better than dependency features; presumably one reason for this is that there are more production rule features (i.e. the ones consisting of only non-terminal nodes in the tree).

4. The C&J parser's templates — one particular set of choices for representing compound structural features — outperform just production rules combined with a LM, quite substantially. This also fits with previous work on using compound features.

5. In comparison with the results for the greatest test set (Table 2), as mentioned above, random is always consistently lower, for each comparable cell in the table. This is expected on the assumption that our MED method for choosing positive and negative examples is an accurate reflection of their goodness with respect to the original sentence. There is generally the same pattern for subsequent results as well, so henceforth we only report random, as the more conservative of the two measures and as a more realistic scenario (comparing a reference sentence against some arbitrary one, not one that we know is the 'worst' of a set of candidates).

Base models (unsupervised) Table 3 gives results when unsupervised parse structures are combined with a large LM. Not surprisingly, these results are quite a lot worse than for the supervised: a drop of around 13% for lexicalised dependencies and 8% for unlexicalised. These are also lower than the LMs in Table 1. For a more detailed look, we calculated the classification accuracy over those sentences where the feature representation differed (which we refer to as the *effective test*

model	acc.%	+l-lm4	+l-lm2
l-lm4	71.83	-	-
l-lm2	65.19	-	-
C&J	91.61	-	-
prod-rule-lex	84.25	84.45	84.19
prod-rule-unlex	75.81	76.56	76.01
dep-lex-named	76.78	81.11	78.58
dep-lex-unnamed	75.75	80.42	77.98
dep-unlex-named	73.25	75.75	74.38
dep-unlex-unnamed	66.80	69.84	66.97

Table 2: Classification scores for supervised parse features and large LM on greatest: accuracy on parse features; parse features plus large l-lm4 LM over these sentences; parse features plus large l-lm2 LM over these sentences

model	overall acc.%	#sent	acc.%
unsuper-lex	62.67	941	65.73
unsuper-unlex	58.26	791	62.20

Table 3: Classification scores for unsupervised parse features and large LM on random: accuracy on parse features; number of sentences where feature vectors differ for positive and negative examples; accuracy over effective test set.

model	#sent	l-lm4	+l-lm4	l-lm2	+l-lm2
unsuper-lex	941	75.29	70.03	69.65	68.97
unsuper-unlex	791	75.53	67.95	71.49	67.95

Table 4: Classification scores for unsupervised parse features and large LM on random: number of sentences where feature vectors differ for positive and negative examples; accuracy of language models (individually, or combined with unsupervised model) over effective test set.

model	#sent	s-lm4	+s-lm4	s-lm2	+s-lm2
unsuper-lex	941	67.80	68.76	67.48	68.76
unsuper-unlex	791	67.95	64.48	67.48	63.84

Table 5: Classification scores for unsupervised parse features and small LM on random: number of sentences where feature vectors differ for positive and negative examples; accuracy of language models (individually, or combined with unsupervised model) over effective test set.

set) and hence the model makes a genuine prediction,[9] to see whether a back-off model would be appropriate: if the accuracy for the unsupervised models is higher than for the LM over the effective test set, the decision for that subset could be based on the unsupervised model decision, with a back-off to the LM. Table 3 includes the effective test set, and the unsupervised classification accuracy scores over that set, for each model.

The LMs may also differ over the effective test sets: we present the classification accuracies for these in Table 4 (for the large LMs) and Table 5 (for the small LMs). The tables include both the accuracy of the LMs alone (e.g. the column l-lm4), as well as the accuracy of the LM in combination with the unsupervised model (e.g. +l-lm4). It is apparent that the effective test sets for the unsupervised models are also in fact easier for the large LMs: their scores on these subsets are all higher than the overall LM accuracies. They are also higher than the unsupervised models, and the combinations are lower than for the LMs alone. The story is different for the small LMs: while the LMs are higher than the unsupervised models, the combinations are better than both, by 3% in the lexicalised case. The conclusion here would be that if a very large LM is available, raw unsupervised parse features would not be helpful; but if only a small LM is available, they would still contribute.

Unsupervised with reliability-based selection Table 6 presents the results for our recall-based reliability measure, along with the effective test set sizes for four thresholds across dependencies with a positive reliability score.[10] These are uniformly poor, and in fact marginally worse than the raw unsupervised features in Table 3, so we do not present combinations with the LMs. We looked at the 10 highest- and 10 lowest-ranked features under this measure (Table 7), along with their raw counts in the training corpus. The highest-ranked ones were believable as reliable instances of dependencies: infinitival *to* in VB TO seems likely to be often correct, as does existential *there* in the first three cases. However, it is quite possible that many of the reliable ones such as PRP VBZ are actually poor at distinguishing between positive and negative examples by virtue of their frequency — they may occur equally often with both, in the same way that words like *the* are useless in general text classification.

Another possibility is that our reliability metric is not capturing the right phenomenon: that it is flagging as errors systematic intentional choices that the unsupervised parser is making, just because they happen to disagree with the systematic choices of the gold standard. For example, in Figures 3 and 4, the unsupervised parse contains the dependency MD VB, choosing serial attachment of auxiliaries and modals to the main verb, while the supervised parser contains MD VBN, choosing attachment of all to the main verb instead. Here the unsupervised parser does not necessarily seem incorrect. Indeed, the very smallness of the proportion of correct instances in the lowest-ranking dependency pairs in Table 7 suggests that these are not just random (and almost always incorrect) choices by the unsupervised parser; four of them in fact appear to relate to the sort of verb attachment choices mentioned. The issue of the difficulty of comparing dependency treebanks in general, perhaps because of fairly arbitrary decisions about headedness, is raised in Zeman et al. (2012); this would appear to be relevant to the task here too.

[9] Recall that in cases where the model cannot make a prediction, it chooses at random.

[10] This consequently does not include dependencies with recall-based reliability zero. We do not present the precision-based ones here.

model	cut-off%	acc.%	#sent
unsuper-lex	100	65.19	915
unsuper-lex	75	64.33	897
unsuper-lex	50	64.37	856
unsuper-lex	25	64.55	708
unsuper-unlex	100	61.44	721
unsuper-unlex	75	62.39	686
unsuper-unlex	50	62.00	629
unsuper-unlex	25	62.44	442

Table 6: Classification scores for unsupervised parse features selected by reliability on random: threshold cut-off accuracy on parse features; number of sentences where feature vectors differ for positive and negative examples (effective test set)

Highest			Lowest		
feature	correct	# in gold	feature	correct	# in gold
EX VBZ	72	179	VBZ VBP	1	201
EX VBP	45	110	VBP VBP	1	218
EX VBD	27	72	VBD NNP	1	292
DT VBZ	108	364	VBP VBD	1	335
$ TO	151	541	VBZ VBD	2	696
PRP VBZ	509	1792	IN CD	2	860
RP VBN	60	222	NN $	1	481
PRP VBD	600	2423	CD RB	1	485
VB TO	1520	6593	CD TO	1	487
NNS JJ	28	103	IN MD	1	694

Table 7: Highest- (left) and lowest- (right) ranking dependency features based on reliability measure. For each, columns represent the dependency; the number of times it was correct in an unsupervised parse; and the number of times it occurred in the gold standard.

model	cut-off%	acc.%	#sent	l-lm4
unsuper-lex(direct)	100	71.97	710	80.28
unsuper-lex(direct)	75	82.17	300	81.17
unsuper-lex(direct)	50	83.26	242	80.58
unsuper-lex(direct)	25	90.00	185	82.70
unsuper-lex(indirect)	100	74.57	920	78.80
unsuper-lex(indirect)	75	82.56	894	78.91
unsuper-lex(indirect)	50	85.19	849	79.62
unsuper-lex(indirect)	25	89.53	717	79.50
unsuper-unlex	100	60.84	738	79.20
unsuper-unlex	75	60.59	708	79.45
unsuper-unlex	50	61.08	657	80.29
unsuper-unlex	25	62.06	506	80.90

Table 8: Classification scores for unsupervised parse features selected by IG: threshold cut-off; accuracy on parse features; number of sentences where feature vectors differ for positive and negative examples (effective test set); large LM score over these sentences

rank	features				
1–5	TO VB	VBN VB	VB VBN	VB TO	VBG VB
6–10	VB VBG	NNS TO	IN RBR	VBN TO	DT RP

Table 9: Highest ranking features based on IG.

Unsupervised with IG selection Table 8 shows the results for selecting the top $k\%$ unsupervised parse features ranked by positive IG scores.[11] We see a dramatic improvement in the lexicalised case (direct) over the raw features (Table 3). The effective test set also falls a lot more for the direct lexicalised case; this is not surprising, as the top (say) 25% lexicalised features on the training set are much less likely to occur in the test set than the top 25% unlexicalised features. On the other hand, the indirect lexicalised case — where lexicalised features are instantiated on the basis of IG-ranked *unlexicalised dependencies* — have a sentence coverage that is much greater, as would be expected, but perhaps surprisingly an accuracy that is comparable to the direct method, and in fact even higher for three of the four thresholds presented in Table 8. For the 25% threshold on the indirect method, the model performs around 10% higher on around 60% of the total test set, making a back-off model a very suitable option.

Table 9 presents the top 10 unlexicalised features. It is not immediately apparent why there is a preponderance of paired verbs, such as VB VBG or VB TO (with the infinitival *to* probably indicating another verb following). Perhaps verb sequences indicate poor sentences, although this would require further inspection of the data.

Illustration of Generated Content To see how unsupervised parses, though bad, might contribute useful structure, we present an example where the unsupervised parse model predicted the better candidate, while the other models did not. The original sentence is given in (1), the preferred candidate in (2), and the dispreferred candidate in (3). The preferred candidate is basically the same as the original, slightly odd punctuation notwithstanding, while the dispreferred candidate has obvious problems in terms of ordering.

(1) For example, their selling caused trading halts to be declared in USAir Group, which closed down 3 7/8 to 41 1/2, Delta Air Lines, which fell 7 3/4 to 69 1/4, and Philips Industries, which sank 3 to 21 1/2.

(2) For example their selling, caused trading halts to be declared in USAir Group which closed, down 3 7/8 to 41 1/2 Delta Air Lines which fell 7 3/4 to 69 1/4 and Philips Industries which sank 3 to 21 1/2.

(3) For example to be declared in USAir Group, their selling, caused trading halts which closed, down 3 7/8 to 41 1/2 Delta Air Lines which fell 7 3/4 to 69 1/4 and Philips Industries which sank 3 to 21 1/2.

Figure 5 shows the unsupervised (upper) and supervised (lower) parses for the preferred candidate (2). For the most part, the supervised parser makes sensible linguistic choices: it identifies as the head of e.g. *Delta Air Lines which fell 7 3/4 to 69 1/4* the final noun of the named entity, which in turn is a dependant of the head of the previous parallel clause; the unsupervised parser, by contrast,

[11] That is, those cases with an IG of zero — i.e. entirely non-distinguishing between positive and negative examples — are excluded, which is why the 100% scores are higher than for the raw features of Table 3.

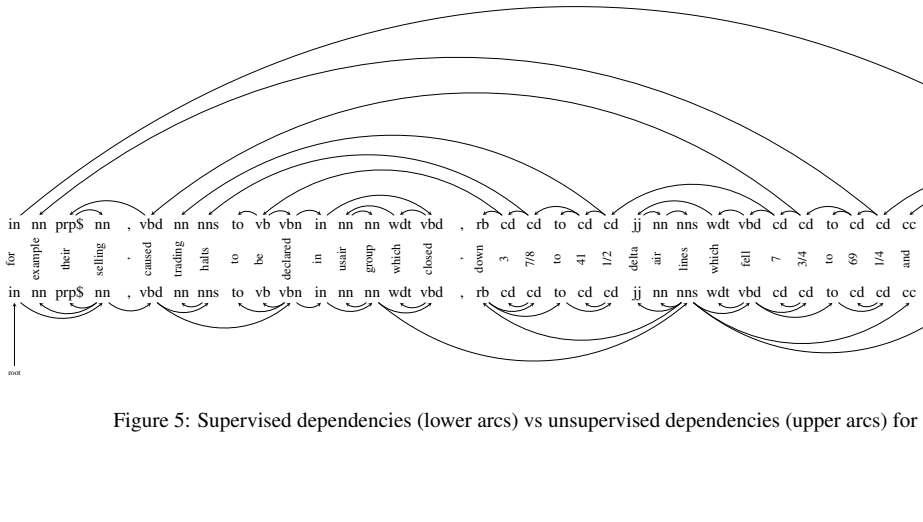

Figure 5: Supervised dependencies (lower arcs) vs unsupervised dependencies (upper arcs) for

has as the ultimate head the final numeral, with odd choices in the intermediate edges as well such as *which* being the head of the named entity in each case. However, the supervised parser makes a fundamental error in grouping two chunks of numerical quantities (i.e. *down 3 7/8 to 41 1/2* and *fell 7 3/4 to 69 1/4*) being associated with the same named entity. The unsupervised parse does not do this: while it makes odd choices, they are consistent odd choices. This consistency of choices may in fact not be accidental, and is further reason to reconsider our approach to assessing the reliability of unsupervised parse edges.

Implications If the reliability-based selection for unsupervised parse features had proved effective, it would have been necessary to find some mechanism to approximate a check against a gold standard, such as constructing a committee of unsupervised parsers and using only those dependencies that received a sufficient number of votes. However, this is not warranted by the results.

For the IG-based selection, what is necessary is a binary annotation of sentence pairs as preferred or dispreferred. This is much less intensive than treebank annotation, and so a reasonable alternative to constructing supervised parsers. As noted, a large LM outperforms raw unsupervised features; but even constructing large corpora (semi-)automatically for many of the world's languages, as per Scannell (2007), is challenging, and the IG-selected features in any case do better than the large LM alone and better still in conjunction with it.

6 Conclusion

For symbolic grammars that are small and/or in development, there may well not be many resources to draw on for building good realisation ranking models for natural language generation. In this paper we have examined unsupervised parsers as a possible source of structural features for such models, notwithstanding their generally much poorer quality of parses than supervised parsers.

We found that they can indeed be useful. In the general case, unsupervised parse features contribute in the case of smaller language models, of the sort that might be available for many less resourced languages. They also contribute very strongly over reasonable sized subsets of the test set once useful features have been identified by Information Gain: there are improvements of up to 10% in classification accuracy over 60% of the test set, making a model that uses unsupervised features and then backs off to a language model an attractive option. This would be feasible in scenarios where it is possible to annotate pairs of sentences as preferred and dispreferred.

Choosing features by a reliability-based measure did not prove useful. However, this may be related to systematic choices made by the unsupervised parser that were different from the gold standard's choices, rather than bad parsing; an option for aligning systematic choices is the HamleDT approach and software of Zeman et al. (2012). Another option for improving performance is the use of compound features, as is used in much of the work discussed in Section 3. The results in this paper for the supervised parsers showed that the model with compound (or 'higher order') features dramatically outperformed the ones with simple features; it could be promising to extend these to the dependency models, as for example in their use in parse reranking, such as by Hall (2007) and Wang and Zong (2011). In addition to evaluating the approach by measures other than binary classification accuracy (e.g. bleu or the ranking score of Cahill et al. (2007)), and examining other potential comparators (e.g. the treebank-trained generator of Belz (2005)), the next major step would be applying it to one of these smaller languages that has motivated the work in this paper.

Acknowledgements

The authors acknowledge the support of ARC Discovery grant DP1095443.

References

Arka, I. W., Andrews, A., Dalrymple, M., Mistica, M., and Simpson, J. (2009). A linguistic and computational morphosyntactic analysis for the applicative -i in Indonesian. In *Proceedings of LFG 2009*, pages 85–105, Cambridge, UK.

Belz, A. (2005). Statistical Generation: Three Methods Compared and Evaluated. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG 2005)*, pages 15–23, Edinburgh, UK.

Bender, E. (2008). Radical Non-Configurationality without Shuffle Operators: An Analysis of Wambaya. In *Proceedings of the Fifteenth Annual Conference on Head-Driven Phrase Structure Grammar (HPSG08)*.

Berg-Kirkpatrick, T. and Klein, D. (2010). Phylogenetic Grammar Induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden.

Butt, M., Dyvik, H., King, T. H., Masuichi, H., and Rohrer, C. (2002). The Parallel Grammar Project. In *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pages 1–7.

Cahill, A., Forst, M., and Rohrer, C. (2007). Stochastic realisation ranking for a free word order language. In *Proceedings of the Eleventh European Workshop on Natural Language Generation*, ENLG '07, pages 17—-24, Morristown, NJ, USA. Association for Computational Linguistics.

Charniak, E. (2001). Immediate-Head Parsing for Language Models. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics (ACL'01)*, pages 124–131, Toulouse, France.

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180, Ann Arbor, Michigan.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based Language Models for Machine Translation. In *Proceedings of MT Summit IX*.

Cherry, C. and Quirk, C. (2008). Discriminative, Syntactic Language Modeling through Latent SVMs. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA'08)*.

Dras, M., Lareau, F., Börschinger, B., Dale, R., Motazedi, Y., Rambow, O., Turpin, M., and Ulinski, M. (2012). Complex Predicates in Arrernte. In *Proceedings of LFG 2012*, Bali, Indonesia.

Filippova, K. and Strube, M. (2009). Tree Linearization in English: Improving Language Model Based Approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'09)*, pages 225–228, Boulder, Colorado.

Hall, K. (2007). K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 392–399, Prague, Czech Republic. Association for Computational Linguistics.

Headden III, W. P., Johnson, M., and McClosky, D. (2009). Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'09)*, pages 101–109, Boulder, Colorado.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. (1999). Estimators for Stochastic "Unification-Based" Grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 535–541, College Park, MD, US.

Klein, D. and Manning, C. (2004). Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 478–485, Barcelona, Spain.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Langkilde-Geary, I. (2000). Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Langkilde-Geary, I. (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 12th International Natural Language Generation Workshop*, pages 17–24. Citeseer.

Maxwell, J. T. and Kaplan, R. M. (1993). The Interface between Phrasal and Functional Constraints. *Computational Linguistics*, 19(4):571–590.

Müller, S. (2010). Persian Complex Predicates and the Limits of Inheritance-Based Analyses. *Journal of Linguistics*, 46(3):601–655.

Mutton, A., Dras, M., Wan, S., and Dale, R. (2007). GLEU: Automatic Evaluation of Sentence-Level Fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 344–351, Prague, Czech Republic.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using Universal Linguistic Knowledge to Guide Grammar Induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP'10)*, pages 1234–1244, Cambridge, MA.

Post, M. and Gildea, D. (2008). Parsers as language models for statistical machine translation. In *Eighth Conference of the Association for Machine Translation in the Americas (AMTA 2008)*, Honolulu, HI.

Rajkumar, R. and White, M. (2010). Designing Agreement Features for Realization Ranking. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1032–1040, Beijing, China.

Riezler, S., King, T. H., Kaplan, R. M., Crouch, R., Maxwell III, J. T., and Johnson, M. (2002). Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of the 40th Annual Meeting of the ACL*, number July, pages 271–278. Association for Computational Linguistics.

Riezler, S. and Vasserman, A. (2004). Gradient feature testing and l1 regularization for maximum entropy parsing. In *Proceedings of EMNLP'04*, Barcelona, Spain.

Rohrer, C. and Forst, M. (2006). Improving coverage and parsing quality of a large-scale LFG for German. In *LREC-2006*, Genoa, Italy.

Scannell, K. (2007). The Crúbadán Project: Corpus building for under-resourced languages. In Fairon, C., Naets, H., Kilgarriff, A., and de Schryver, G.-M., editors, *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop*, volume 4 of *Cahiers du Cental*, pages 5–15. Louvain-la-Neuve, Belgium.

Stolcke, A. (2002). SRILM – An Extensible Language Modeling Toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, pages 901–904, Denver, CO, US.

Velldal, E. and Oepen, S. (2005). Maximum entropy models for realization ranking. In *Proceedings of the 10th MTSummit X Phuket Thailand*. Citeseer.

Velldal, E. and Oepen, S. (2006). Statistical Ranking in Tactical Generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP'06)*, pages 517–525, Sydney, Australia.

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

Wang, Z. and Zong, C. (2011). Parse reranking based on higher-order lexical dependencies. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP'11)*, pages 1251–1259, Chiang Mai, Thailand.

White, M. and Rajkumar, R. (2009). Perceptron Reranking for CCG Realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, number August, pages 410–419. Association for Computational Linguistics.

White, M. and Rajkumar, R. (2012). Minimal Dependency Length in Realization Ranking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP'12)*, pages 244–255, Jeju, Korea.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)*, pages 412–420.

Zeman, D., Mareček, D., Popel, M., Ramasamy, L., Štěpánek, J., Žabokrtský, Z., and Hajič, J. (2012). Hamledt: To parse or not to parse? In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.