Feature selection for fluency ranking

Daniël de Kok
University of Groningen
d.j.a.de.kok@rug.nl

Abstract

Fluency rankers are used in modern sentence generation systems to pick sentences that are not just grammatical, but also fluent. It has been shown that feature-based models, such as maximum entropy models, work well for this task.

Since maximum entropy models allow for incorporation of arbitrary real-valued features, it is often attractive to create very general feature templates, that create a huge number of features. To select the most discriminative features, feature selection can be applied. In this paper we compare three feature selection methods: frequency-based selection, a generalization of maximum entropy feature selection for ranking tasks with real-valued features, and a new selection method based on feature value correlation. We show that the often-used frequency-based selection performs badly compared to maximum entropy feature selection, and that models with a few hundred well-picked features are competitive to models with no feature selection applied. In the experiments described in this paper, we compressed a model of approximately 490.000 features to 1.000 features.

1 Introduction

As shown previously, maximum entropy models have proven to be viable for fluency ranking (Nakanishi et al., 2005; Velldal and Oepen, 2006; Velldal, 2008). The basic principle of maximum entropy models is to minimize assumptions, while imposing constraints such that the expected feature value is equal to the observed feature value in the training data. In its canonical form, the probability of a certain event (y) occurring in the context (x) is a log-linear combination of features and feature weights, where $Z(x)$ is a normalization over all events in context x (Berger et al., 1996):

$$p(y|x) = \frac{1}{Z(x)} exp \sum_{i=1}^{n} \lambda_i f_i \qquad (1)$$

The training process estimates optimal feature weights, given the constraints and the principle of maximum entropy. In fluency ranking the input (e.g. a dependency structure) is a context, and a realization of that input is an event within that context.

Features can be hand-crafted or generated automatically using very general feature templates. For example, if we apply a template *rule* that enumerates the rules used to construct a derivation tree to the partial tree in figure 1 the *rule(max_xp(np))* and *rule(np_det_n)* features will be created.

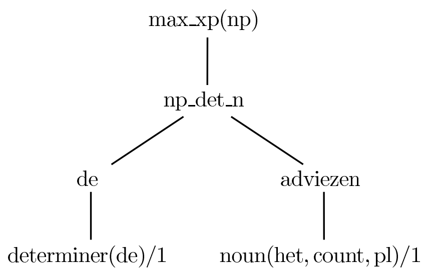

Figure 1: Partial derivation tree for the noun phrase *de adviezen* (*the advices*).

To achieve high accuracy in fluency ranking quickly, it is attractive to capture as much of the language generation process as possible. For instance, in sentence realization, one could extract nearly every aspect of a derivation tree as a feature using very general templates. This path is followed in recent work, such as Velldal (2008). The advantage of this approach is that it requires little human labor, and generally gives good ranking performance. However, the generality of templates leads to huge models in terms of number of features. For instance, the model that we will discuss contains about 490,000 features when no feature selection is applied. Such models are very opaque, giving very little understanding of good discriminators for fluency ranking, and the size of the models may also be inconvenient. To make such models more compact and transparent, feature selection can be applied.

In this paper we make the following contributions: we modify a maximum entropy feature selection method for ranking tasks; we introduce a new feature selection method based on statistical correlation of features; we compare the performance of the preceding feature selection methods, plus a commonly used frequency-based method; and we give an analysis of the most effective features for fluency ranking.

2 Feature selection

2.1 Introduction

Feature selection is a process that tries to extract $S \subset F$ from a set of features F, such that the model using S performs comparably to the model using F. Such a compression of a feature set can be obtained if there are features: that occur sporadically; that correlate strongly with other features (features that show the same behavior within events and contexts); or have values with little or no correlation to the classification or ranking.

Features that do have no correlation to the classification can be removed from the model. For a set of highly-correlating features, one feature can be selected to represent the whole group.

Initially it may seem attractive to perform fluency selection by training a model on all features, selecting features with relatively high weights. However, if features overlap, weight mass will usually be divided over these features. For instance, suppose that f_1 alone has a weight of 0.5 in a given model. If we retrain the model, after adding the features $f_2..f_5$ that behave identically to f_1, the weight may be distributed evenly between $f_1..f_5$, giving each feature the weight 0.1.

In the following sections, we will give a short overview of previous research in feature selection, and will then proceed to give a more detailed description of three feature selection methods.

2.2 Background

Feature selection can be seen as model selection, where the best model of all models that can be formed using a set of features should be selected. Madigan and Raftery (1994) propose an method for model selection aptly named *Occam's window*. This method excludes models that do not perform competitively to other models or that do not perform better than one of its submodels. Although this method is conceptually firm, it is nearly infeasable to apply it with the number of features used in fluency ranking. Berger et al. (1996) propose a selection method that iteratively builds a maximum entropy model, adding features that improve the model. We modify this method for ranking tasks in section 2.5. Ratnaparkhi (1999) uses a simple frequency-based cutoff, where features that occur infrequently are excluded. We discuss a variant of this selection criterium in section 2.3. Perkins et al. (2003) describe an approach where feature selection is applied as a part of model parameter estimation. They rely on the fact that ℓ_1 regularizers have a tendency to force a subset of weights to zero. However, such integrated approaches rely on parameter tuning to get the requested number of features.

In the fluency ranking literature, the use of a frequency cut-off (Velldal and Oepen, 2006) and ℓ_1 regularization (Cahill et al., 2007) is prevalent. We are not aware of any detailed studies that compare feature selection methods for fluency ranking.

2.3 Frequency-based selection

In frequency-based selection we follow Malouf and Van Noord (2004), and count for each feature f the number of inputs where there are at least two realizations y_1, y_2, such that $f(y_1) \neq f(y_2)$. We then use the first N features with the most frequent changes from the resulting feature frequency list.

Veldall (2008) also experiments with this selection method, and suggests to apply frequency-based selection to fluency ranking models that will be distributed to the public (for compactness' sake). In the variant he and Malouf and Van Noord (2004) discuss, all features that change within more than n contexts are included in the model.

2.4 Correlation-based selection

While frequency-based selection helps selecting features that are discriminative, it cannot account for feature overlap. Discriminative features that have a strong correlation to features that were selected previously may still be added.

To detect overlap, we calculate the correlation of a candidate feature and exclude the feature if it shows a high correlation with features selected previously. To estimate Pearson's correlation of two features, we calculate the sample correlation coefficient,

$$r_{f_1,f_2} = \frac{\sum_{x \in X, y \in Y}(f_1(x,y) - \bar{f}_1)(f_2(x,y) - \bar{f}_2)}{(n-1)s_{f_1}s_{f_2}} \quad (2)$$

where $\bar{f}_x$ is the average feature value of f_x, and s_{f_x} is the sample standard deviation of f_x.

Of course, correlation can only indicate overlap, and is in itself not enough to find effective features. In our experiments with correlation-based selection we used frequency-based selection as described in 2.3, to make an initial ranking of feature effectiveness.

2.5 Maximum entropy feature selection

Correlation-based selection can detect overlap, however, there is yet another spurious type of feature that may reduce its effectiveness. Features with relatively noisy values may contribute less than their frequency of change may seem to indicate. For instance, consider a feature that returns a completely random value for every context. Not only does this feature change very often, its correlation with other features will also be weak. Such a feature may seem attractive from the point of view of a frequency or correlation-based method, but is useless in practice.

To account for both problems, we have to measure the effectiveness of features in terms of how much their addition to the model can improve prediction of the training sample. Or in other words: does the log-likelihood of the training data increase?

We have modified the Selective Gain Computation (SGC) algorithm described by Zhou et al. (2003) for ranking tasks rather than classification tasks. This method builds upon the maximum entropy feature selection method described by Berger et al. (1996). In this method features are added iteratively to a model that is initially uniform. During each step, the feature that provides the highest gain as a result of being added to the model, is selected and added to the model.

In maximum entropy modeling, the weights of the features in a model are optimized simultaneously. However, optimizing the weights of the features in model $p_{S,f}$ for every candidate feature f is computationally intractable. As a simplification, it is assumed that the weights of features that are already in the model are not affected by the addition of a feature f. As a result, the optimal weight α of f can be found using a simple line search method.

However, as Zhou et al. (2003) note, there is still an inefficiency in that the weight of every candidate feature is recalculated during every selection step. They observe that gains of remaining candidate features rarely increase as the result of adding a feature. If it is assumed that this never happens, a list of candidate features ordered by gain can be kept. To account for the fact that the topmost feature in that list may have lost its effectiveness as the result of a previous addition of a feature to the model, the gain of the topmost feature is recalculated and reinserted into the list according to its new gain. When the topmost feature retains its position, it is selected and added to the model.

Since we use feature selection with features that are not binary, and for a ranking task, we modified the recursive forms of the model to:

$$\begin{aligned} sum^{\alpha}_{S \cup f}(y|x) &= sum_S(y|x) \cdot e^{\alpha} f(y) && (3) \\ Z^{\alpha}_{S \cup f}(x) &= Z_S(x) - \sum_y sum_S(y|x) \\ &\quad + \sum_y sum_{S \cup f}(y|x) && (4) \end{aligned}$$

Another issue that needs to be dealt with is the calculation of context and event probabilities. In the literature two approaches are prevalent. The first approach divides the probability mass uniformly over contexts, and the probability of events within a context is proportional to the event score (Osborne, 2000):

$$p(x) = \frac{1}{|X|} \tag{5}$$

$$p(y|x) = \frac{p(x)}{\left(\frac{score(x,y)}{\sum_y score(x,y)}\right)} \tag{6}$$

where $|X|$ is the number of contexts. The second approach puts more emphasis on the contexts that contain relatively many events with high scores, by making the context probability dependent on the scores of events within that context (Malouf and van Noord, 2004):

$$p(x) = \frac{\sum_y score(x,y)}{\sum_{y \in X} score(x,y)} \tag{7}$$

In our experiments, the second definition of context probability outperformed the first by such a wide margin, that we only used the second definition in the experiments described in this paper.

2.6 A note on overlap detection

Although maximum-entropy based feature-selection may be worthwhile in itself, the technique can also be used during feature engineering to find overlapping features. In the selection method of Berger et al. (1996), the weight and gain of each candidate feature is re-estimated during each selection step. We can exploit the changes in gains to detect overlap between a selected feature f_n, and the candidates for f_{n+1}. If the gain of a feature changed drastically in the selection of f_{n+1} compared to that of f_n, this feature has overlap with f_n.

To determine which features had a drastic change in gain, we determine whether the change has a significance with a confidence interval of 99% after normalization. The normalized gain change is calculated in the following manner as described in algorithm 1.

Algorithm 1 Calculation of the normalized gain delta

```
$\Delta G_f \leftarrow G_{f,n} - G_{f,n-1}$
**if** $\Delta G_f \geq 0.0$ **then**
    $\Delta G_{f,norm} \leftarrow \frac{\Delta G_f}{G_{f,n}}$
**else**
    $\Delta G_{f,norm} \leftarrow \frac{\Delta G_f}{G_{f,n-1}}$
**end if**
```

3 Experimental setup

3.1 Task

We evaluated the feature selection methods in conjunction with a sentence realizer for Dutch. Sentences are realized with a chart generator for the Alpino wide-coverage grammar and lexicon (Bouma et al., 2001). As the input of the chart generator, we use abstract dependency structures, which are dependency structures leaving out information such as word order. During generation, we store the compressed derivation trees and associated (HPSG-inspired) attribute-value structures for every realization of an abstract dependency structure. We then use feature templates to extract features from the derivation trees. Two classes of features (and templates) can be distinguished *output features* that model the output of a process and *construction features* that model the process that constructs the output.

3.1.1 Output features

Currently, there are two output features, both representing auxiliary distributions (Johnson and Riezler, 2000): a word trigram model and a part-of-speech trigram model. The part-of-speech tag set consists of the Alpino part of speech tags. Both models are trained on newspaper articles, consisting of 110 million words, from the Twente Nieuws Corpus[1].

The probability of unknown trigrams is estimated using linear interpolation smoothing (Brants, 2000). Unknown word probabilities are determined with Laplacian smoothing.

[1] http://wwwhome.cs.utwente.nl/druid/TwNC/TwNC-main.html

3.1.2 Construction features

The construction feature templates consist of templates that are used for parse disambiguation, and templates that are specifically targeted at generation. The parse disambiguation features are used in the Alpino parser for Dutch, and model various linguistic phenomena that can indicate preferred readings. The following aspects of a realization are described by parse disambiguation features:

- Topicalization of (non-)NPs and subjects.
- Use of long-distance/local dependencies.
- Orderings in the middle field.
- Identifiers of grammar rules used to build the derivation tree.
- Parent-daughter combinations.

Output features for parse disambiguation, such as features describing dependency triples, were not used. Additionally, we use most of the templates described by Velldal (2008):

- Local derivation subtrees with optional grandparenting, with a maximum of three parents.
- Local derivation subtrees with back-off and optional grand-parenting, with a maximum of three parents.
- Binned word domination frequencies of the daughters of a node.
- Binned standard deviation of word domination of node daughters.

3.2 Data

The training and evaluation data was constructed by parsing 11764 sentences of 5-25 tokens, that were randomly selected from the (unannotated) Dutch Wikipedia of August 2008[2], with the wide-coverage Alpino parser. For every sentence, the best parse according to the disambiguation component was extracted and considered to be correct. The Alpino system achieves a concept accuracy of around 90% on common Dutch corpora (Van Noord, 2007). The original sentence is considered to be the best realization of the abstract dependency structure of the best parse.

We then used the Alpino chart generator to construct derivation trees that realize the abstract dependency structure of the best parse. The resulting derivation trees, including attribute-value structures associated with each node, are compressed and stored in a derivation treebank. Training and testing data was then obtained by extracting features from derivation trees stored in the derivation treebank. At this time, the realizations are also scored using the General Text Matcher method (GTM) (Melamed et al., 2003), by comparing them to the original sentence. We have previously experimented with ROUGE-N scores, which gave rise to similar results. However, it is shown that GTM shows the highest correlation with human judgments (Cahill, 2009).

3.3 Methodology

To evaluate the feature selection methods, we first train models for each selection method in three steps: 1. For each abstract dependency structure in the training data 100 realizations (and corresponding features) are randomly selected. 2. Feature selection is applied, and the N-best features according to the selection method are extracted. 3. A maximum entropy model is trained using the TADM[3] software, with a ℓ_2 prior of 0.001, and using the N-best features.

We used 5884 training instances (abstract dependency trees, and scored realizations) to train the model. The maximum entropy selection method was used with a weight convergence threshold of $1e^{-6}$. Correlation is considered to be strong enough for overlap in the correlation-based method when two features have a correlation coefficient of $r_{f_1,f_2} \geq 0.9$

Each model is then evaluated using 5880 held-out evaluation instances, where we select only instances with 5 or more realizations (4184 instances), to avoid trivial ranking cases. For every instance, we select the realization that is the closest to the original sentence to be the correct realization[4]. We then calculate the fraction of instances for which the

[2] `http://ilps.science.uva.nl/WikiXML/`

[3] `http://tadm.sourceforge.net/`

[4] We follow this approach, because the original sentence is not always exactly reproduced by the generator.

model picked the correct sentence. Of course, this is a fairly strict evaluation, since there may be multiple equally fluent sentences.

4 Results

4.1 Comparing the candidates

Since each feature selection method that we evaluated gives us a ranked list of features, we can train models for an increasing number of features. We have followed this approach, and created models for each method, using 100 to 5000 features with a step size of 100 features. Figure 2 shows the accuracy for all selection methods after N features. We have also added the line that indicates the accuracy that is obtained when a model is trained with all features (490667 features).

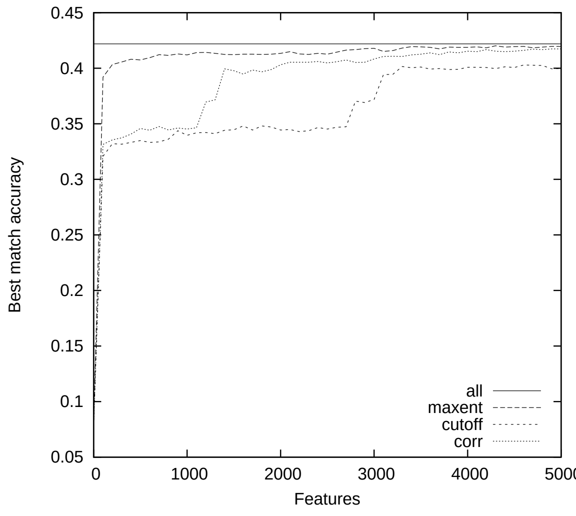

Figure 2: Accuracy of maximum entropy, correlation-based, and frequency-based selection methods after selecting N features ($N \leq 5000$), with increments of 100 features.

In this graph we can see two interesting phenomena. First of all, only a very small number of features is required to perform this task almost as well as a model with all extracted features. Secondly, the maximum entropy feature selection model is able to select the most effective features quickly - fewer than 1000 features are necessary to achieve a relatively high accuracy.

As expected, the frequency-based method fared worse than maximum entropy selection. Initially some very useful features, such as the n-gram models are selected, but improvement of accuracy quickly stagnates. We expect this to be caused by overlap of newly selected features with features that were initially selected. Even after selecting 5000 features, this method does not reach the same accuracy as the maximum entropy selection method had after selecting only a few hundred features.

The correlation-based selection method fares better than the frequency-based method without overlap detection. This clearly shows that feature overlap is a problem. However, the correlation-based method does not achieve good accuracy as quickly as the maximum entropy selection method. There are three possible explanations for this. First, there may be noisy features that are frequent, and since they show no overlap with selected features they are good candidates according to the correlation-based method. Second, less frequent features that overlap with a frequent feature in a subset of contexts may show a low correlation. Third, some less frequent features may still be very discriminative for the contexts where they appear, while more frequent features may just be a small indicator for a sentence to be fluent or non-fluent. It is possible to refine the correlation-based method to deal with the second class of problems. However, the lack of performance of the correlation-based method makes this unattractive - during every selection step a candidate feature needs to be compared with all previously selected features, rather than some abstraction of them.

Table 1 shows the peak accuracies when selecting up to 5000 features with the feature selection methods described. Accuracy scores of the random selection baseline, the n-gram models, and a model trained on all features are included for comparison. The random selection baseline picks a realizations randomly. The n-gram models are the very same n-gram models that were used as auxiliary distributions in the feature-based models. The combined word/tag n-gram model was created by training a model with both n-gram models as the only features. We also list a variation of the frequency-based method often used in other work (such as Velldal (2008) and Malouf and Van Noord (2004)), where there is a fixed frequency threshold (here 4), rather than using the first N most frequently changing features.

Besides confirming the observation that feature selection can compress models very well, this table shows that the popular method of using a frequency cutoff, still gives a lot of opportunity for compressing the model further. In practice, it seems best to plot a graph as shown in figure 2, choose an acceptable accuracy, and to use the (number of) features that can provide that accuracy.

Method	Features	Accuracy
Random	0	0.0778
Tag n-gram	1	0.2039
Word n-gram	1	0.2799
Word/tag n-gram	2	0.2908
All	490667	0.4220
Fixed cutoff (4)	90103	0.4181
Frequency	4600	0.4029
Correlation	4700	0.4172
Maxent	4300	0.4201

Table 1: Peak accuracies for the maximum entropy, correlation-based, and frequency-based selection methods when selecting up to 5000 features. Accuracies for random, n-gram and full models are included for comparison.

4.2 Overlap in frequency-based selection

As we argued in section 2.5, the primary disadvantage of the frequency-based selection is that it cannot account for correlation between features. In the extreme case, we could have two very distinctive features f_1 and f_2 that behave exactly the same in any event. While adding f_2 after adding f_1 does not improve the model, frequency-based selection cannot detect this. To support this argumentation empirically, we analyzed the first 100 selected features to find good examples of this overlap.

Initially, the frequency-based selection chooses three distinctive features that are also selected by the maximum entropy selection method: the two n-gram language models, and a preference for topicalized NP subjects. After that, features that indicate whether the *vp_arg_v(np)* rule was used change very frequently within a context. However, this aspect of the parse tree is embodied in 13 successively selected features. Due to the generality of the feature templates, there are multiple templates to capture the use of this grammar rule: through local derivation trees (with optional grandparenting), back-off for local derivation trees, and the features that calculate lexical node dominance.

Another example of such overlap in the first 100 features is in features modeling the use of the *non_wh_topicalization(np)* rule. Features containing this rule identifier are used 30 times in sequence, where it occurs in local derivation subtrees (with varying amounts of context), back-off local derivation subtrees, lexical node domination, or as a grandparent of another local derivation subtree.

In the first 100 features, there were many overlapping features, and we expect that this also is the case for more infrequent features.

4.3 Effective features

The maximum entropy selection method shows that only a small number of features is necessary to perform fluency ranking (section 4.1). The first features that were selected in maximum entropy selection can give us good insight of what features are important for fluency ranking. Table 2 shows the 10 topmost features as returned by the maximum entropy selection. The weights shown in this table, are those given by the selection method, and their sign indicates whether the feature was characteristic of a fluent sentence (+) or a non-fluent sentence (−).

As expected (see table 1) the n-gram models are a very important predictor for fluency. The only surprise here may be that the overlap between both n-gram models is small enough to have both models as a prominent feature. While the tag n-gram model is a worse predictor than the word n-gram model, we expect that the tag n-gram model is especially useful for estimating fluency of phrases with word sequences that are unknown to the word n-gram model.

The next feature that was selected, *r2(vp_arg_v(pred),2,vproj_vc)*, indicates that the rule *vp_arg_v(pred)* was used with a *vproj_vc* node as its second daughter. This combination occurs when the predicative complement is placed after the copula, for instance as in *Amsterdam is de hoofdstad van Nederland* (*Amsterdam is the capital of The Netherlands*), rather than *De hoofdstad van Nederland is Amsterdam* (*The capital of The Netherlands is Amsterdam*).

The feature *s1(non_subj_np_topic)* and its negative weight indicates that realizations with non-topicalized NP subjects are dispreferred. In Dutch, non-topicalized NP subjects arise in the OVS word-order, such as in *de soep eet Jan* (*the soup eats Jan*). While this is legal, SVO word-order is clearly preferred (*Jan eet de soep*).

The next selected feature (*ldsb(vc_vb,vb_v, [vproj_vc,vp_arg_v(pp)])*) is also related to topicalization: it usually indicates a preference for prepositional complements that are not topicalized. For instance, *dit zorgde voor veel verdeeldheid* (*this caused lots of discord*) is preferred over the PP-topicalized *voor veel verdeeldheid zorgde dit* (*lots of discord caused this*).

ldsb(n_n_pps,pp_p_arg(np),[]) gives preference PP-ordering in conjuncts where the PP modifier follows the head. For instance, the conjunct *groepen van bestaan of khandas* (*planes of existance or khandas*) is preferred by this feature over *van bestaan groepen of khandas* (*of existence planes or khandas*).

The next feature (*lds_dl(mod2,[pp_p_arg(np)], [1],[non_wh_topicalization(modifier)])*) forms an exception to the dispreference of topicalization of PPs. If we have a PP that modifies a copula in a subject-predicate structure, topicalization of the PP can make the realization more fluent. For instance, *volgens Williamson is dit de synthese* (*according to Williamson is this the synthesis*) is considered more fluent than *dit is de synthese volgens Williamson* (*this is the synthesis according to Williamson*).

The final three features deal with punctuation. Since punctuation is very prevalent in Wikipedia texts due to the amount of definitions and clarifications, punctuation-related features are common. Note that the last two *lds_dl* features may seem to be overlapping, they are not: they use different frequency bins for word domination.

Weight	**Name**
0.012	ngram_lm
0.009	ngram_tag
0.087	r2(vp_arg_v(pred),2,vproj_vc)
-0.094	s1(non_subj_np_topic)
0.090	ldsb(vc_vb,vb_v, [vproj_vc,vp_arg_v(pp)])
0.083	ldsb(n_n_pps,pp_p_arg(np),[])
0.067	lds_dl(mod2,[pp_p_arg(np)],[1], [non_wh_topicalization(modifier)])
0.251	lds_dl(start_start_ligg_streep, [top_start_xp,punct(ligg_streep), top_start_xp],[0,0,1],[top_start])
0.186	lds_dl(start_start_ligg_streep, [top_start_xp,punct(ligg_streep), top_start_xp],[0,0,2],[top_start])
0.132	r2(n_n_modroot(haak),5,l)

Table 2: The first 10 features returned by maximum entropy feature selection, including the weights estimated by this feature selection method.

5 Conclusions and future work

Our conclusion after performing experiments with feature selection is twofold. First, fluency models can be compressed enormously by applying feature selection, without losing much in terms of accuracy. Second, we only need a small number of targeted features to perform fluency ranking.

The maximum entropy feature selection method shows a high accuracy after selecting just a few features. The commonly used frequency-based selection method fares far worse, and requires addition of many more features to achieve the same performance as the maximum entropy method. By experimenting with a correlation-based selection method that uses the frequency method to make an initial ordering of features, but skips features that show a high correlation with previously selected features, we have shown that the ineffectiveness of frequency-based selection can be attributed partly to feature overlap. However, the maximum entropy method was still more effective in our experiments.

In the future, we hope to evaluate the same techniques to parse disambiguation. We also plan to compare the feature selection methods described in this paper to selection by imposing a ℓ_1 prior.

The feature selection methods described in this paper are usable for feature sets devised for ranking and classification tasks, especially when huge sets of automatically extracted features are used. An open source implementation of the methods described in this paper is available[5], and is optimized to work on large data and feature sets.

[5]http://danieldk.eu/Code/FeatureSqueeze/

References

A.L. Berger, V.J.D. Pietra, and S.A.D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):71.

G. Bouma, G. Van Noord, and R. Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. In *Computational Linguistics in the Netherlands 2000. Selected Papers from the 11th CLIN Meeting*.

T. Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, Seattle, WA.

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

A. Cahill. 2009. Correlating Human and Automatic Evaluation of a German Surface Realiser. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 97–100.

M. Johnson and S. Riezler. 2000. Exploiting auxiliary distributions in stochastic unification-based grammars. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 154–161, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

D. Madigan and A.E. Raftery. 1994. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546.

R. Malouf and G. van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop: Beyond shallow analyses - Formalisms and statistical modeling for deep analyses.* JST CREST, March.

I. D. Melamed, R. Green, and J. P. Turian. 2003. Precision and recall of machine translation. In *HLT-NAACL.*

H. Nakanishi, Y. Miyao, and J. Tsujii. 2005. Probabilistic models for disambiguation of an hpsg-based chart generator. In *Parsing '05: Proceedings of the Ninth International Workshop on Parsing Technology*, pages 93–102, Morristown, NJ, USA. Association for Computational Linguistics.

M. Osborne. 2000. Estimation of stochastic attribute-value grammars using an informative sample. In *Proceedings of the 18th conference on Computational linguistics*, pages 586–592, Morristown, NJ, USA. Association for Computational Linguistics.

S. Perkins, K. Lacker, and J. Theiler. 2003. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1):151–175.

G. Van Noord. 2007. Using self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 1–10. Association for Computational Linguistics.

E. Velldal and S. Oepen. 2006. Statistical ranking in tactical generation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 517–525. Association for Computational Linguistics.

E. Velldal. 2008. *Empirical Realization Ranking*. Ph.D. thesis, University of Oslo, Department of Informatics.

Y. Zhou, F. Weng, L. Wu, and H. Schmidt. 2003. A fast algorithm for feature selection in conditional maximum entropy modeling.