Feature Subset Selection in Conditional Random Fields for Named Entity Recognition

Roman Klinger and Christoph M. Friedrich
Department of Bioinformatics
Fraunhofer Institute for Algorithms and Scientific Computing (SCAI)
53754 Sankt Augustin, Germany
{roman.klinger,christoph.friedrich}@scai.fraunhofer.de

Abstract

In the application of Conditional Random Fields (CRF), a huge number of features is typically taken into account. These models can deal with inter-dependent and correlated data with an enormous complexity. The application of feature subset selection is important to improve performance, speed and explainability.

We present and compare filtering methods using information gain or χ^2 as well as an iterative approach for pruning features with low weights.

The evaluation shows that with only 3 % of the original number of features a 60 % inference speed-up is possible. The F_1 measure decreases only slightly.

1 Introduction

Feature selection is well established for many machine learning methods, for instance for feed-forward neural networks [2] or decision trees [17]. The main advantages are an improvement of prediction performance, faster training and prediction as well as a better understanding of the models [4]. Methods can be distinguished between filters not using the learning algorithm and wrappers using the learning algorithm as a black box [8]. An overview of approaches for classification tasks is given by Liu and Motoda [11], more specifically for text classification by Yang and Pederson [25].

Such feature selection methods are not well established for Conditional Random Fields [9], a Maximum Entropy-based method [1] for structured data. We propose methods coping with the demanding task of handling sequential data represented by a huge number of features. Reported numbers are for instance 1,686,456 for a gene name tagger [5]. Due to this high complexity, training and inference times can explode. These high numbers of features are generated by automated methods, i. e., for every token in the training set, features are generated. Then, all other tokens are tested for these features. This method is typically applied to determine the identity of words as well as for prefixes or suffixes of different length or for learning schemata of regular expression-like patterns [20].

Only a few approaches dealing with feature handling for CRFs are published. The work by McCallum [12] demonstrates a method for iteratively constructing feature conjunctions that would increase conditional log-likelihood if added to the model. An analysis of different penalty terms for regularization is shown by Peng and McCallum [15]. Goodman [3] presented a related analysis of exponential priors for Maximum Entropy models. The very recent work of Vail et al. [23, 22] shows feature selection in Conditional Random Fields by L_1-norm regularization in the robotics domain, a work which is related to the selection in Maximum Entropy models proposed by Koh et al. [7].

These methods incorporate the training procedure in the selection process. In contrast, we present different filter methods for feature selection to limit the complexity before starting the training. It is demonstrated how the sequential structure of text can be respected by filtering approaches originally developed for classification problems (especially in Section 3.1.2 and 3.1.3). These filter methods are compared to an iterative approach.

The paper is organized as follows. A short description of Conditional Random Fields is given in Section 2. We introduce different approaches for feature selection in Section 3, namely the adaption of classification methods to filter features and an iterative approach to remove features with low weights. In Section 4, an evaluation of the feature selection methods is given. The results show a reduction of complexity leading to improved speed and better explainability.

2 Conditional Random Fields and Sequential Data

Conditional Random Fields [9, 13] are a family of probabilistic, undirected graphical models for computing the probability $P_{\vec{\lambda}}(\vec{y}|\vec{x})$ of a possible label sequence $\vec{y} = (y_0, \ldots, y_n)$ given the input sequence $\vec{x} = (x_0, \ldots, x_n)$. In the context of Named Entity Recognition, this observation sequence $\vec{x}$ corresponds to the tokenized text. The label sequence is encoded in a label alphabet $\mathcal{L} = \{I\text{-<}entity\text{>}, O, B\text{-<}entity\text{>}\}$ where $y_i = O$ means that x_i is outside an entity, $y_i = B$-<*entity*> means that x_i is the beginning and $y_i = I$-<*entity*> means that x_i is inside an entity.

In general, a CRF is given by

$$P(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}) \qquad (1)$$

International Conference RANLP 2009 - Borovets, Bulgaria, pages 185–191

with normalization $Z(\vec{x}) = \sum_{\vec{y} \in \mathcal{Y}} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y})$, where Ψ_j are the different factors and $\mathcal{Y}$ is the set of all possible label sequences. These factor functions combine different features f_i of the considered part of the text and label sequence and usually correspond to maximal cliques on the independence graph.

For simplicity, we focus on the linear-chain CRF as a special case of the general CRF. The factors are given in the form

$$\Psi_j(\vec{x}, \vec{y}) = \exp\left(\sum_{i=1}^{m} \lambda_i f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big)\right). \quad (2)$$

Each feature $f_i(\cdot)$ is weighted by $\lambda_i \in \mathbb{R}$. These weights are the parameters to be learned in the model and later used in the iterative approach for feature subset selection (in Section 3.2). An example formulation of features $f_i(\cdot)$ is

$$f_i\Big(y_{j-1}, y_j, \vec{x}, j\Big) = \begin{cases} 1, \text{if } y_{j-1} = s_i' \text{ and } y_j = s_i'' \text{ and } \varphi_k(x_j) \\ 0, \text{otherwise,} \end{cases} \quad (3)$$

where $s_i', s_i'' \in \mathcal{L}$.

There are two kinds of features. First, the ones representing the token sequence as a feature vector sequence[1] (these features are referred to as $\varphi_k \in \mathfrak{F}$). An example is that $\varphi_k(x_j)$ holds if and only if x_j is a capital word. Second, the feature functions representing $\varphi_k(x_j)$ with label transitions, to be referred to as $f_i \in \mathcal{F}$.

Optimization of the parameters λ_i is often performed with the limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS, Nocedal, [14]) on the convex function $L(\mathcal{T})$ with the training data $\mathcal{T}$, including a penalty term:

$$L(\mathcal{T}) = \log P_{\vec{\lambda}}(\vec{y}|\vec{x})) - \sum_{i=1}^{m} \frac{\lambda_i^2}{2\sigma^2}. \quad (4)$$

3 Feature Selection Methods for CRFs

An exhaustive search to find the optimal feature subset is not possible due to the large number of features and comparatively long training times. Hence, a wrapper approach considering the CRF as a black box is impractical. We compare different approaches for feature subset selection of sequential data in CRFs, i. e., filtering methods (in Section 3.1) and an iterative method (in Section 3.2).

3.1 Filter

To apply filter methods for classification tasks, the sequence data have to be represented as classification instances. For a pair of sequences $(\vec{y}, \vec{x})$ this is done with respect to the incorporated factors in the CRF. For every factor $\Psi_j(\vec{y}, \vec{x})$, an instance is built. The labels are all dependencies on $\vec{y}$, the features have the values at the corresponding position for $\vec{x}$.

[1]For simplicity, we only consider boolean features.

For the factors in a linear-chain CRF as shown in Equation 2, the instance $\mathcal{I}_j = (\mathfrak{L}_j, \vec{\varphi}_j)$ at position j $(0 < j \leq n)$ has the label $\mathfrak{L}_j = (y_{j-1}, y_j)$ from a set of all possible transitions[2] $\mathfrak{L}_j \in \mathcal{L}^2$. The feature values are $\varphi_k(x_j)$. Instances are built for all positions in all training examples from $\mathcal{T}$.

The features are ranked by measures presented below. The best p_{filter} features (where p_{filter} is a parameter specifying the percentage of kept features) are selected to represent the text data.

In the following, the number of generated instances is denoted with h, the number of instances with feature $\varphi(x_j)$ with value 1 with h_j^1 and with value 0 with h_j^0. The number of instances with label $\mathfrak{L}_\ell$ is $h(\ell)$, with feature values 1 or 0 of those with $h_j^1(\ell)$ and $h_j^0(\ell)$ respectively.

3.1.1 Simple Information Gain

Our first approach for measuring the quality of a feature is the use of information gain of a feature $IG(\varphi(x_j))$ to differentiate between all possible labels $\mathfrak{L}_\ell$. It is defined as

$$IG(\varphi(x_j)) = I\left(\frac{h_j^1}{h}, \frac{h_j^0}{h}\right) - R(\varphi(x_j)) \quad (5)$$

where $I(\cdot)$ is the information content

$$I(p_1, p_2) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 \quad (6)$$

with probabilities p_1 and p_2. $R(\varphi(x_j))$ is the remainder of bits of information after testing feature $\varphi(x_j)$:

$$R(\varphi(x_j)) = \sum_{\ell=1}^{|\mathcal{L}^2|} \left(\frac{h(\ell)}{h} I\left(\frac{h_j^1(\ell)}{h(\ell)}, \frac{h_j^0(\ell)}{h(\ell)}\right)\right). \quad (7)$$

For comparing the features it is sufficient to compute $R(\cdot)$ because $I(\cdot)$ is constant for one feature [18]. Determining this measure for m features with $|\mathcal{L}|$ different labels lasts $\mathcal{O}(m|\mathcal{L}^2|)$.

Ranking the features with this approach cannot lead to differences between transitions in the CRF. Therefore, it is referred to as *Simple IG*.

3.1.2 Information Gain One-Against-All

The limitation of *Simple IG* is the disregard of differences between transitions in CRF. To cope with that, we assign a list of the best p_{filter} features to every transition $\mathfrak{L}_\ell$. In general, every clique in the graph has its own evaluation of features $\varphi(\cdot)$.

The remainder, the measure for the quality of a feature in Equation 7, changes slightly to

$$R_{OAA}(\varphi(x_j), \mathfrak{L}_\ell) = \left(\frac{h(\ell)}{h} I\left(\frac{h_j^1(\ell)}{h(\ell)}, \frac{h_j^0(\ell)}{h(\ell)}\right)\right) + \left(\frac{\bar{h}(\ell)}{h} I\left(\frac{\bar{h}_j^1(\ell)}{\bar{h}(\ell)}, \frac{\bar{h}_j^0(\ell)}{\bar{h}(\ell)}\right)\right) \quad (8)$$

[2]For the linear-chain CRF of order 1. In general, $\mathfrak{L}_j \in \mathcal{L}^{c+1}$ with order c holds.

	$\varphi(x_j)=1$	$\varphi(x_j)=0$	$\sum$
$\mathfrak{L}_\ell$	$h_j^1(\ell)$	$h_j^0(\ell)$	$h(\ell)$
$\mathfrak{L}_{\neq\ell}$	$\bar{h}_j^1(\ell)$	$\bar{h}_j^0(\ell)$	$\bar{h}(\ell)$
$\sum$	h_j^1	h_j^0	h

Table 1: The 2×2 contingency table.

where

$$\bar{h}(\ell) = \sum_{l \in \{1,\dots,|\mathcal{L}|^2\} \setminus \ell} h(l)\,,$$

and $\bar{h}_j^1(l)$ and $\bar{h}_j^0(l)$ analogous. This approach is applied to rank the features $\varphi(x_j)$ for every transition $\mathfrak{L}_\ell$ separately. It is referred to as *Information Gain One-Against-All (IG-OAA)*. The runtime is $\mathcal{O}(m|\mathcal{L}|)$ and therefore less than *Simple IG*.

3.1.3 χ^2-Statistics

Another well-known and often incorporated ranking method are χ^2-*statistics* [16]. The 2×2 contingency table is defined for each feature $\varphi(x_j)$ and each transition $\mathfrak{L}_\ell$ compared to all other transitions (cf. Table 1). The χ^2-statistic is then computed by

$$\chi^2(\varphi(x_j), \mathfrak{L}_j) = \frac{\left(h_j^1(\ell) \cdot \bar{h}_j^0(\ell) - h_j^0(\ell) \cdot \bar{h}_j^1(\ell)\right)^2 \cdot h}{h(\ell) \cdot \bar{h}(\ell) \cdot h_j^0 \cdot h_j^1} \quad (9)$$

Similar to *IG OAA*, this is performed to rank the features $\varphi(x_j)$ for every transition $\mathfrak{L}_\ell$ separately. Therefore, the runtime is also $\mathcal{O}(m|\mathcal{L}|)$. We refer to this method as χ^2 *OAA*.

3.1.4 Random

The most simple method is a *random* ranking and selection of features. This is used as a baseline to evaluate the other measures presented in the previous sections.

3.2 Iterative Feature Pruning

Training a CRF is commonly performed by the iterative algorithm L-BFGS to assign weights λ_i to all feature functions $f_i \in \mathcal{F}$ such that $\mathcal{L}(T)$ is maximized (compare to Equation 4). Typically, many weights are close to 0. The idea of Iterative Feature Pruning (IFP) is that feature functions with low absolute weight value have a low impact on the output sequence.

Based on this assumption, the algorithm (see pseudo code in Figure 1) starts with a fully optimized CRF using all features representing the training data (Line 2). The next step is the removal of features with lowest absolute value (3). The parameter $p = 1 - \frac{|S|}{|\mathcal{F}|}$ specifies the percentage of features to be removed in each iteration (13–15). S is the set of remaining features after one iteration of IFP.

In each step, a retraining with L-BFGS is performed to allow an adaption of the model by adjusting the weights for the remaining features (16). After that, the pruning is repeated (17) until no features are left (11).

During this process, at each iteration of pruning, the current performance of the model is evaluated and stored (8). This information can be used to select the final feature set (Line 4, an heuristic is shown in Section 4.3) and to train a full model with the identified feature subset (Line 5).

To illustrate the process of IFP, it is shown exemplarily by means of extreme examples for one data set[3] in Figure 2. On the horizontal axis, all L-BFGS training iterations are shown consecutively with the intermediate pruning steps. The blue dotted line shows the decrease of the number of features, the red solid line the F_1 measure for the training data, the green dashed line the F_1 measure for one validation data set. Removing 40 % of the features in each step ($p = 0.4$) clearly depicts the process of removing and retraining of the model. Experiments[4] have shown that in general lower numbers of features can be achieved with comparable F_1 measures if smaller values of p are used. The drawback is the higher number of iterations needed. We focus on $p = 0.1$ which leads to good results as shown in Section 4.

```
1:  function IFP(crf,trainData, valData, p)
2:      crf = BFGS(crf, trainData, valData)
3:      log = PRUNING-STEP(crf, trainData, valData, p)
4:      crf = SELECTFEATURESET(log)
5:      crf = BFGS(crf, trainData+valData,null)
6:  end function
7:  function PRUNING-STEP(crf, trainData, valData, p)
8:      log ← EVALUATE(trainData, valData, crf)
9:      F = GETFEATURESET(crf)
10:     if F = ∅ then
11:         return log
12:     end if
13:     S = features with lowest weights such that
                p = 1 − |S|/|F|
14:     F = F\S
15:     SETFEATURESET(crf,F)
16:     crf = BFGS(crf, trainingData)
17:     return PRUNING-STEP(crf, trainData, valData)
18: end function
```

Figure 1: Iterative Feature Pruning Algorithm (starting with method IFP(·) in Line 1)

4 Results

In this section, the methods proposed in Section 3 are evaluated on the data sets and configurations of the CRFs described in Section 4.1. The hypotheses to be analyzed are:

Selecting a reasonable subset of features:

A1 Improves explainability of the CRF model,

A2 Improves training time and tagging time which is beneficial for developing as well as applying the model,

A3 Improves performance in F_1 measure or does not decrease it dramatically.

Additionally, we assume that one method is superior to all others:

B One method outperforms the others.

[3] CoNLL data set introduced in Section 4.1

[4] Results not shown here due to page limitation.

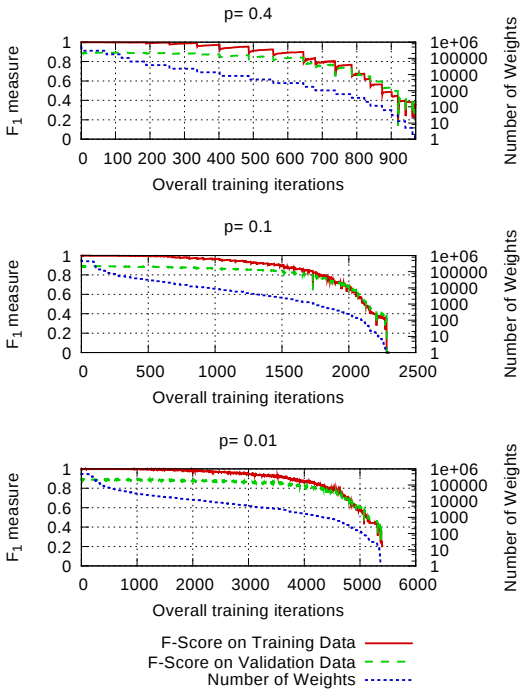

Figure 2: Iterative Feature Pruning for CRF trained on CoNLL data with different percentages of features pruned in each iteration.

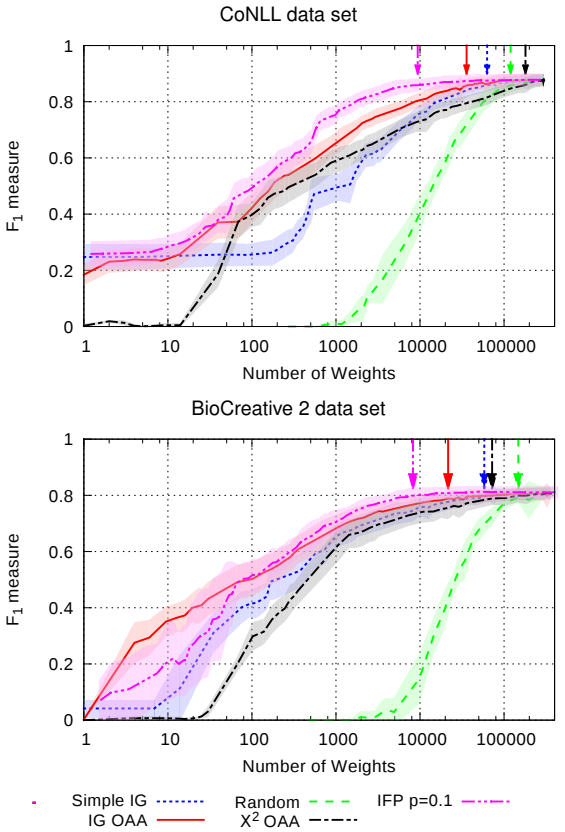

Figure 3: Comparison of the average F_1 measure using 10-fold cross-validation. Used features are determined with methods described in Section 3. The transparent band shows the standard deviation. The arrows show a possible selection of the model features ($g = 2 \cdot 10^{-6}$, $\Delta = 0.02$, cf. Section 4.3).

The evaluations in the following form the basis for the discussion of these hypotheses in Section 4.4.

4.1 Data Sets used for Evaluation

The results and evaluations are shown on the basis of two data sets with slightly different configurations of the CRF. Quantities of entities are given in Table 2.

The BioCreative 2 Gene Mention Task data (BC2) contains entities of the class *Gene/Protein* with the specialty of acceptance of several boundaries for entities [24]. We incorporate the configuration of the CRF as described in a participating system using only the shortest possible annotation as exact true positive per entity [6, 21].

Name	Training Set		
	B-	I-	O
BC2	18165	15017	382983
CoNLL	29466	13180	213499
	Test Set		
	B-	I-	O
BC2	6290	4801	128915
CoNLL	5654	2458	38554

Table 2: Numbers of labels in data sets, the different entity classes are added in CoNLL.

The CoNLL data [19] is an annotation of the Reuters corpus [10] containing the classes *person*, *organization*, *locations* and *misc*. We use an order-one CRF with offset conjunction combining features of one preceding and succeeding token for each position in the text sequence. The feature set is fairly standard with Word-As-Class, prefix and suffix generation of length two, three and four as well as regular expressions detecting capital letters, numbers, dashes and dots separately and as parts of tokens. The combination of the provided sets "train" and "testa" is used for training and "testb" for testing.

For evaluating the inference time on a larger set, a uniform sample from the Medline[5] database of 10,000 entries is used additionally. Each one comprises titles, author names, and abstracts. The number of tokens is 958869 for BC2 and 960744 for CoNLL (additionally to BC2 tokenization, splitting on all dots is performed for CoNLL data).

4.2 Cross-Validation on the Training Sets

As a basis for parameter selection (presented in Section 4.3) and to evaluate the impact of feature selection, 10-fold cross

[5] http://www.nlm.nih.gov/databases/databases_medline.html

validation is performed on the training sets (Section 4.1). The results are shown in Figure 3. The curves depict the average F_1 measure[6] of the 10 partitions. The different numbers of features are detected with different parameters specified for the respective selection method. The transparent band around the line depicts the standard deviation for the according number of features.[7] The significance of the difference of the methods is tested regarding the area under the curves in Figure 3 via Welch's t-test with a significance level of $\alpha = 0.05$.

Comparing the results on the two data sets, the methods lead to similar results whereas the differences are clearer on CoNLL data. All approaches outperform the random selection significantly. The approach of χ^2 *OAA* is worse than the conceptionally similar *IG OAA* and the more naive *Simple IG* on BC2 data.

IG OAA outperforms all other filtering approaches. Assuming the goal to reach the highest possible F_1 measure, *IFP* leads to better results than *IG OAA* on both data sets. Only if an extremely small number of features remains (fewer than about 50), *IG OAA* leads to better results. The superiority of *IFP* to χ^2 *OAA* is significant ($p = 0.02$) on the CoNLL data set.

4.3 Results on independent test sets

We need to find the parameter assignment to determine the feature subset in the final model. For the filter approaches, the parameter is p_{filter}. For *IFP*, the meaningful number of features at which the pruning is stopped has to be detected. Based on the smoothed[8] values of F_1 measure in 10-fold cross-validation (see Figure 3), we define two measures to automatically detect these parameters: The maximally accepted loss in F_1 measure is denoted with Δ, the threshold for the gradient is g. The detection of the feature subset is performed via backward selection starting with high numbers of features. The first position on the curve for which the gradient is smaller than g or the F_1 measure is smaller than Δ is selected. The values $g = 2 \cdot 10^{-6}$ and $\Delta = 0.02$ lead to the positions denoted by arrows in Figure 3. The results for these values are evaluated in the following. The advantage of backward selection to forward selection is that it may capture interacting features more easily [8].

A model is built on the full training set applying *IFP* or filtering with the detected parameters. In Figure 4 the results on independent test sets mentioned in Section 4.1 are depicted. The smallest numbers of features are achieved by *IFP* followed by *IG OAA*.

The F_1 measures decrease up to the accepted $\Delta = 0.02$ on BC2 data and to a lower amount for the CoNLL data. The best trade-off between F_1 measure and the number of features (depicted in third bar chart) is always achieved by *IFP* followed by *IG OAA*.

A smaller number of features should induce a faster model in training and inference. The time of evaluating $P_{\vec{\lambda}}(\vec{y}|\vec{x})$ (compare to Equation 1) on all training examples is shown in the fourth bar charts. This computation is crucial for training durations as it has to be performed many times. These numbers correspond roughly to the number of features, the durations are smaller than for the original model. The ratio is not linear due to the necessary forward-backward algorithm calls (whose runtime is quadratic in the number of possible labels) and dynamic memory allocation times.

These numbers naturally lead to dramatically reduced training times. Training the full CoNLL model lasts 5788 seconds, $2397s$ for BC2 respectively. With the feature set detected by *IFP*, these numbers reduce to $2156s$ and $670s$. This improvement is not helpful in practice, as the *IFP* procedure incorporates training the model. However, filtering via *IG OAA* improves overall training time as it is computationally inexpensive. It leads to $5230s$ and $1002s$ for training a model.[9]

Reducing the number of features also leads to a faster inference[10]: The fifth bar charts show durations for tagging 10000 sampled abstracts from Medline. Best results are achieved by *IFP* (CoNLL: $10.52s$ instead of $17.56s$, BC2: $2.42s$ instead of $4.56s$), followed by the filtering methods which do not differ remarkably (*IG OAA*: CoNLL: $14s$, BC2: $2.65s$).

Summarizing, a reduced training iteration time of $76\,\%$ of original time evaluating $P_{\vec{\lambda}}(\vec{y}|\vec{x})$ with a loss of only $1.7\,\%$ F_1 (absolute value) on the BC2 data is possible with *IFP*. The tagging time is reduced to $53\,\%$. On the CoNLL data, a loss of only $0.57\,\%$ in F_1 occurs with savings of even $54\,\%$ of original computing time. Tagging time is reduced to $60\,\%$.

4.4 Discussion

Comparing the methods, the results are similar for the data sets: In 10-fold cross-validation, the random method is dominated by all other methods. *Simple IG* or χ^2 *OAA* are second worst, depending on the data set. *IFP* is the best method, closely followed by *IG OAA*.

The *Simple IG* lacks the representation of different transitions in the features which is especially important for the CoNLL data with 4 entity classes of interest. No dictionaries with members of these classes have been used in the presented setting, so all classes are memorized with automatically generated features, hence, a large number of different features is needed for the different transitions in the CRF.

The method χ^2 *OAA* always leads to worse results compared to *IG OAA* on the 10-fold cross-validation although it is a systematically similar approach. The reason is presumably the unbalancedness of the labels in the generated classification instances[11] which is taken into account in the

[6] $F_\beta = \frac{(1+\beta^2)\cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$, where $\beta = 1$

[7] In iterative feature pruning, different numbers of features can occur at the same iteration of pruning. In that case, the closest detected number of features is used to compute average and standard deviation.

[8] Smoothing via computation of median in a running window.

[9] The relation to the numbers of features and the iteration durations is not linear as the needed numbers of training iterations differs.

[10] Measuring only the computation, not the time to read the data from hard disk and to extract the features.

[11] Transitions of intermediate terms (like (*O*,*O*)) are for instance much more frequent than those of beginnings of entities (like (*B*,*B*)).

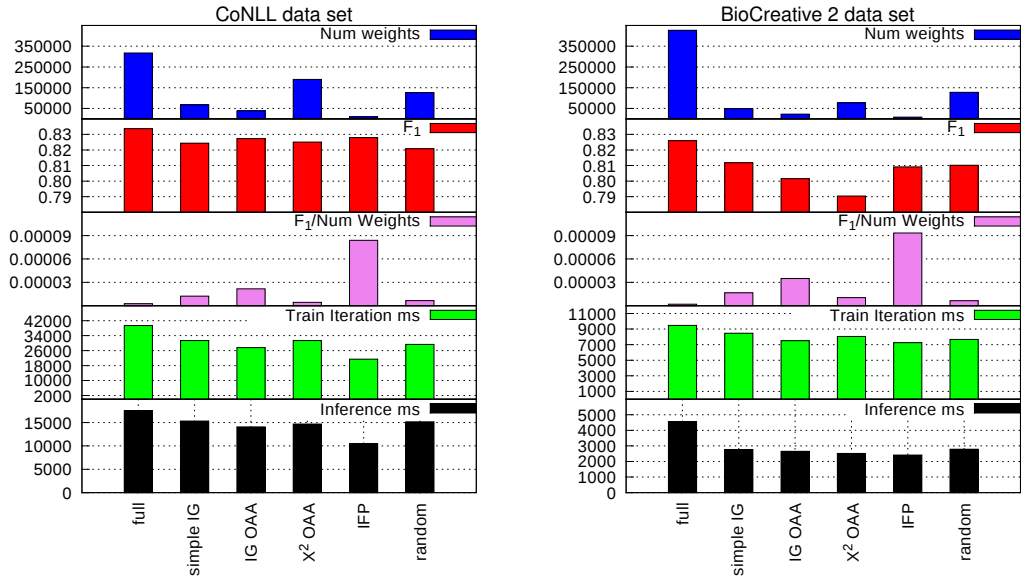

Figure 4: Results on independent test sets (Note the different scales for some of the histograms.)

information remainder (Equation 7 and 8) but not in χ^2 (Equation 9).

IFP leads to better results than *IG OAA* for high F_1 measures. The reason is the limitation of *IG OAA* to use the same number of features (but not the same set) for each transition (specified by p_{filter}). This does not hold for *IFP* as it only relies on the model structure itself. The drawback is the higher computational cost due to the incorporated L-BFGS optimization.

The *random* method does not lead to good results, but it should be noted, that even this approach can remove 30 %–40 % without a dramatic decrease in F_1 measure. The reason are unessential redundancies in the full feature set.

Mainly all these results are reflected on the independent test sets. *IFP* or *IG OAA* has the best trade-off between F_1 measure and the number of features. Good inference speed-ups can be achieved with both methods, corresponding to the low numbers of features. However, *IFP* cannot be used to speed up training as it incorporates the training procedure itself. Hence, *IG OAA* is proposed for this instance. For reducing tagging time as well as for understanding the model, *IFP* should be used as it leads to the smallest numbers of features.

In Table 3, the percentages and numbers of remaining features accepting maximally 0.01 loss in F_1 measure are depicted. Much more features can be ignored in the BC2 than in the CoNLL setting. This can be an indicator for the need for better generalizing features: as the classes have to be memorized by automatically generated features, less features can be eliminated. In BC2, features with better generalization characteristics are implemented.

We conclude the results with an investigation of the hypotheses:

A1 The explainability of models applying feature selection is improved by the lower complexity. The need for many automatically generated features can be an indicator for a possible improvement by implementation of features with better generalization properties. Detected relevant features representing words or important pre- and suffixes help understand the different entity classes of interest. The fact that noisy features are removed allows for an investigation of the remaining features which can be assumed as meaningful[12]

A2 Training and tagging time are decreased by the lower complexity of the CRF. To improve training time, the use of a filtering approach like *IG OAA* is proposed as these methods are computationally inexpensive. To improve tagging time, *IFP* should be used as it leads to the lowest numbers of features.

A3 A small decrease in F_1 measure has to be accepted for the benefit of a model with a considerable lower number of features.

B The recommendation for a method depends on the application: For improving training time, a filtering method should be used, preferably *IG OAA* as it shows best results. For improving tagging time, the computationally more expensive *IFP* can be applied.

	Number of Features		
Data Set	Original	Remaining	%
CoNLL	269506	17377	6.45
BC 2	492611	11096	2.25

Table 3: Minimal numbers of original features needed to lose maximally 0.01 of F_1 measure applying IFP.

[12] Lists of all features of the models are available online to demonstrate the explainability comprehensively. `http://www.scai.fraunhofer.de/ranlp-crf-fs.html`

5 Conclusion and Future Work

A huge number of features is typically used to represent input text in CRFs. We presented different approaches for feature subset selection, novel adaptations of filtering to the sequential structure of text as well as an iterative method. The methods have been evaluated on two domains, showing a decrease of computing time and complexity of the model. The F_1 measure varies slightly.

Summarizing, *IG OAA* is the best filter approach, a lower number of features can only be achieved with Iterative Feature Pruning (*IFP*) with a similar F_1 measure. *IFP* relies only on the CRF structure itself, so it is able to deal with different numbers of features per transition in contrast to the One-Against-All methods. Its main disadvantage is its higher computing cost due to the incorporated training process. It is notable that *IFP* and *IG OAA* are methods taking the sequential structure of the text into account, *IFP* via using the model itself, *IG OAA* via different feature sets for different transitions. The method *Simple IG*, which does not select features with respect to transitions, leads to worse results.

Building a new Named Entity Recognizer often includes annotation of a corpus. It has to be investigated, how the need for features changes during the process of enriching the training set with examples (e. g. via active learning).

Another point is that the proposed methods allow for more complex feature generations (e.g. with more context information). It has to be studied if new features, which could not be implemented before due to an exhaustive memory consumption or run-time demand, could improve the state-of-the-art results.

6 Acknowledgments

We thank especially Günter Rudolph, Katrin Tomanek, Juliane Fluck, Corinna Kolářik and Martin Hofmann-Apitius for fruitful discussions. Many thanks to the reviewers for comprehensive comments. This work has been partially funded by the Max-Planck-Society–Fraunhofer-Society Machine Learning Collaboration (`http://lip.fml.tuebingen.mpg.de/`).

References

[1] A. L. Berger, S. D. Pietra, and V. J. D. Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[3] J. Goodman. Exponential Priors for Maximum Entropy Models. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 305–312, 2004.

[4] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[5] C.-N. Hsu, Y.-M. Chang, C.-J. Kuo, Y.-S. Lin, H.-S. Huang, and I.-F. Chung. Integrating high dimensional bi-directional parsing models for gene mention tagging. *Bioinformatics*, 24(13):i286–i294, Jul 2008.

[6] R. Klinger, C. M. Friedrich, J. Fluck, and M. Hofmann-Apitius. Named Entity Recognition with Combinations of Conditional Random Fields. In *Proc. of the Second BioCreative Challenge Evaluation Workshop*, pages 89–91, 2007.

[7] K. Koh, S.-J. Kim, and S. Boyd. An Interior-Point Method for Large-Scale l_1-Regularized Logistic Regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

[8] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. *Artifical Intelligence – Special Issue on relevance*, 97(1-2):273–324, 1997.

[9] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann Publishers, 2001.

[10] D. D. Lewis, Y. Yang, T. Rose, and F. Li. A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[11] H. Liu and H. Motoda. *Computational Methods of Feature Selection*. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC, 2008.

[12] A. McCallum. Efficiently Inducing Features of Conditional Random Fields. In *Proc. of the 19th Conference in Uncertainty in Artificial Intelligence (UAI-2003)*, pages 403–410, 2003.

[13] R. McDonald and F. Pereira. Identifying Gene and Protein Mentions in Text Using Conditional Random Fields. *BMC Bioinformatics*, 6 Suppl 1:S6, 2005.

[14] J. Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, July 1980.

[15] F. Peng and A. McCallum. Accurate Information Extraction from Research Papers using Conditional Random Fields. In *Proc. of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 329–336, 2004.

[16] R. L. Plackett. Karl Pearson and the Chi-Squared Test. *International Statistical Review*, 51(1):59–72, 1983.

[17] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[18] S. Russell and P. Norvig. *Artificial Intelligence – A Modern Approach*. Prentice Hall, 2003.

[19] E. F. T. K. Sang and F. De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In W. Daelemans and M. Osborne, editors, *Proc. of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[20] B. Settles. ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

[21] L. Smith, L. K. Tanabe, R. J. nee Ando, C.-J. Kuo, I.-F. Chung, C.-N. Hsu, Y.-S. Lin, R. Klinger, C. M. Friedrich, K. Ganchev, M. Torii, H. Liu, B. Haddow, C. A. Struble, R. J. Povinelli, A. Vlachos, W. A. Baumgartner, L. Hunter, B. Carpenter, R. T.-H. Tsai, H.-J. Dai, F. Liu, Y. Chen, C. Sun, S. Katrenko, P. Adriaans, C. Blaschke, R. Torres, M. Neves, P. Nakov, A. Divoli, M. Maa-Lpez, J. Mata, and W. J. Wilbur. Overview of BioCreative II gene mention recognition. *Genome Biol*, 9 Suppl 2:S2, 2008.

[22] D. L. Vail. *Conditional Random Fields for Activity Recognition*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, USA, 2008.

[23] D. L. Vail, J. D. Lafferty, and M. M. Veloso. Feature Selection in Conditional Random Fields for Activity Recognition. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3379–3384, 2007.

[24] J. Wilbur, L. Smith, and L. Tanabe. BioCreative 2. Gene Mention Task. In *Proc. of the Second BioCreative Challenge Evaluation Workshop*, pages 7–9, 2007.

[25] Y. Yang and J. O. Pederson. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the International Conference on Machine Learning*, 1997.