A feature induction algorithm with application to named entity disambiguation

Abstract

The performance of NLP classifiers largely depends on the quality of the features considered for prediction (feature engineering). However, as the number of features increases, the more likely overfitting becomes and performance decreases. Also, due to the very large number of features, only slimple linear classifiers are considered, thus disregarding potentially predictive non-linear combinations of features. Here we propose an automated method for feature induction, which selects and includes in the model features and feature combinations which are likely to be useful for the prediction.The resulting model relies on a smaller feature set, is non-linear and is more accurate than the baseline, which is the model trained on the entire feature set. The method uses a greedy filtering approach based on various univariate measures of feature relevance and it is very fast in practice. Also, our feature induction method is independent of the classifier used: we applied it together with Naïve Bayes and Perceptron models.

1 Introduction

NLP classification tasks are characterized by a very large number of features. When the number of available samples is smaller (for example several orders of magnitude less samples), overfitting can occur, leading to poor performance. In order to avoid overfitting, *feature selection* is commonly applied. In (Guyon and Elisseeff, 2003), the main approaches to feature selection are summarized: *filter*, *wrapper* and *embedded* methods. Filters use some scoring measure to quantify the predictivity of each feature independently. Then, features are ranked and only the top scoring ones are kept in the final model. The most popular measures for feature predictivity are Mutual Information (Lewis, 1992; Taira and Haruno, 1999), Information Gain (Uguz, 2011; Yang and Pedersen, 1997), Kullback-Leibler divergence (Lee and Lee, 2006; Schneider, 2004; Lee et al., 2011), Chi-squared statistics (Yang and Pedersen, 1997; Mesleh, 2007), Fisher statistics, Pearson correlation, etc. In (Yang and Pedersen, 1997) and (Forman, 2003), comparisons of the most popular methods are presented. Filter methods are computationally fast, but the univariate scoring can lead to the elimination of features that are useful only in combinations (Guyon and Elisseeff, 2003). Wrapper methods (Kohavi and John, 1997) can score subsets of features directly, by evaluating the performance of the classifier on the respective subset. A strategy of iteratively updating the subset of features is used, with the goal of finding a (close to) optimal subset. Forward selection, backward elimination, branch-and-bound (Narendra and Fukunaga, 1977), simulated annealing (Ekbal et al., 2011), genetic algorithms (Yang and Honavar, 1998) are among the most popular strategies. Wrapper methods tend to be slow in practice, because a classifier needs to be trained at each iteration. Embedded methods are explicitly optimizing an objective function that incorporates feature selection. In general, the objective is an expression of the trade-off between the goodness of fit and the number of variables that participate in the model. For example, l_1 penalties (Haffner et al., 2005) are combined with the likelihood objective in maximum entropy models in order to keep the number of predictors small.

For most NLP classification tasks, the number of features is very large. If no experts are available for selecting the most promising features for a specific task, the choice is really vast. In (Kamolvilassatian, 2002), the authors systemati-

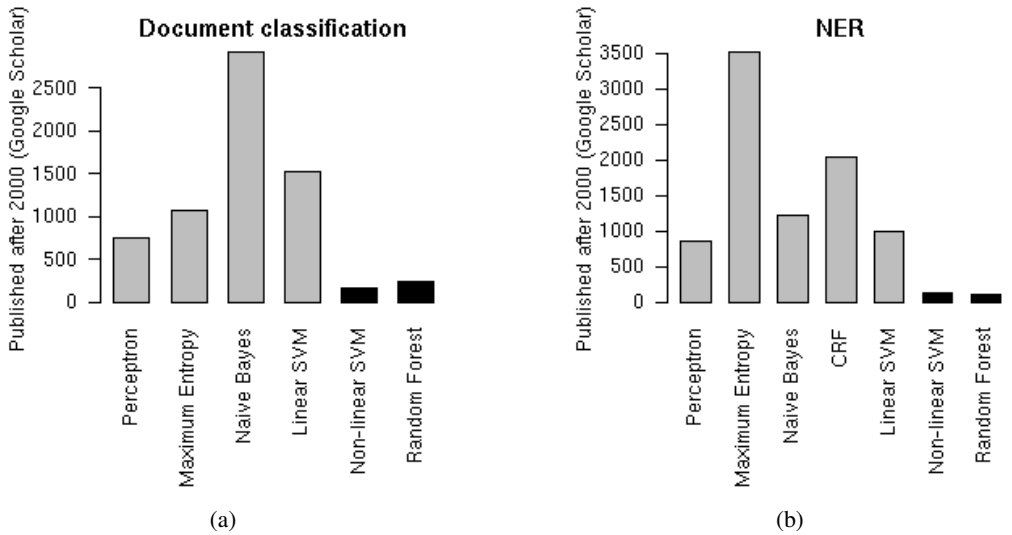

Figure 1: Classifier models used for a) document classification and b) named entity recognition. Linear models are represented with gray bars and non-linear models with black.

cally list of all features (with parameters), such as for example n-grams (n is a parameter), context of words, part of speech, lemmas, stems, etc. Owing to the very large number of features, linear (or log-linear) classifiers are preferred, because they are robust and can be trained fast. Simple search in Google Scholar shows that most frequently used models for document classification are Naive Bayes, linear SVMs (Cortes and Vapnik, 1995), Perceptrons (Rosenblatt, 1957) and Maximum Entropy (Berger et al., 1996) (Figure 1a). In contrast, non-linear models such as non-linear SVMs and Random Forest (Breiman, 2001) and Classification trees (Breiman et al., 1984) are significantly under-represented. For Named Entity Recognition, Maximum Entropy and CRFs (Lafferty, 2001) are mostly used, but other linear models like Perceptron, Naive Bayes and linear SVMs are employed (Figure 1b). Non-linear models are significantly less frequent. Feature induction can be used to efficiently introduce non-linearity in large models, in the form of feature conjunctions. As the space of all conjunctions of arbitrary length is very large ($2^{\#\text{features}}$), a greedy search approach is applied for selecting the most promising conjunctions with reasonable computational cost. In (McCallum, 2003), a method for inducing features and conjunctions especially tailored to CRF models is proposed. Iteratively, the most promising feature or conjunction to be added to the model is identified. To this end, a gain function is defined, for evaluating the improvement of the likelihood target upon the addition of the feature. Conjunctions are considered only among the top scoring feature candidates and the features already included in the model. In (Vens and Costa, 2011), the authors use random forests to form feature conjunctions, by traversing the trees from the root to the leaves.

In this article we present a method for feature selection and feature induction. The advantages of our method are fast running time and generality, in the sense that any classifier can be used. In the Methods section we introduce the algorithm, in the Data section the datasets that we used for validation are presented. The Results sections overviews our results, we comment on the errors in the section Error analysis and conclude the paper with a Discussion.

2 Methods

Given are N pairs of observations and labels $(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)$. The observations X_i are over a set of p binary predicates (or terms) $T_1, ..., T_p$, which we call *atomic features*. For example, an atomic feature is an indicator of presence or absence of a particular word in a document. The class label can take the values from the set $\{c_1, c_2, ..., c_K\}$. In this article. we use the notion of 'features' to denote predicates, and not the classical feature functions $f(X_i, c_i)$ commonly used in NLP tasks. The reason is that we wish to be consistent with the established notions of 'feature selection' and 'feature induction',

which otherwise would have to be called 'predicate selection' and 'predicate induction'.

The purpose of our method is to find a set of features consisting of atomic features or conjunctions of atomic features which can predict the class variable with high accuracy. The classification model is the user's choice.

Our algorithm is entitled Fast ITerative Selection and Induction (FITSI) and is a greedy heuristic search through the space of the atomic features and conjunctions. The main steps are described in Algorithm 1. A *feature selection* step and a *feature induction* step alternate in an iterative process. At each iteration, we first rank the features according to some score σ that measures the univariate relevance of each feature w.r.t. the class variable. The choices for σ are described in Section 2.1. We keep only the top k scoring features, where k is a generic parameter of our algorithm which can be either a percentage of the total number of features, or the number of features with scores larger than a threshold. Then, we add to the features set conjunctions between atomic features larger than a certain rank l and features or conjunctions from the entire list. The algorithm runs for m iterations, which allows for conjunctions of length up to m to be generated.

Below we discuss in detail the two key ingredients of our algorithm: measures of feature relevance and the algorithm for generation of conjunctions.

2.1 Feature ranking and filtering

For ranking and filtering features (σ parameter in Algorithm 1), we implemented several measures: mutual information (MI), information gain (IG), symmetrical uncertainty (SU) and Fisher tests (FT). Each measure returns a score, which is an estimate of the predictive power of each feature w.r.t. the class variable. In a multi-class setting, there are several ways to compute scores: either globally, trying to capture the overall association of the feature with the class variable, or separately, computing a relevance score w.r.t. each class and then summing the scores. We prefer the later approach because it is equally fair to small and large classes. We summarize the class-specific scores by taking either the sum or their maximum value.

In what follows, we denote with Y_i the indicator variable of class c_i: $Y_i[k] = 1$, if $Y[k] = c_i$ and 0, otherwise.

Algorithm 1 Fast Iterative Selection and Induction

Require: $\{T_1, ..., T_p\}$, σ, k, l, m
Initialize feature list $\Phi \leftarrow [T_1, ..., T_p]$
for $i \in \{1, ..., m-1\}$ **do**
  Feature selection:
  $\Phi \leftarrow$ sort(Φ, σ) ▷ Sort the list according to σ scores
  $\Phi \leftarrow \Phi[1..k]$ ▷ Keep only the top k features
  Feature induction:
  $\Gamma \leftarrow$ generateConjunctions(Φ, l)
  if $\Phi == \Phi \bigcup \Gamma$ **then** ▷ No new conjunctions
    break
  else
    $\Phi \leftarrow \Phi \bigcup \Gamma$ ▷ Append Γ to Φ
  end if
end for
return Φ

Algorithm 2 generateConjunctions(Φ, l)

$\Gamma \leftarrow \emptyset$ ▷ Initialize with empty list
for $i \in \{1, ..., l\}$ **do**
  if $\Phi[i]$ is atomic **then**
    for $j \in \{i+1, ..., \text{length}(\Phi)\}$ **do**
      $\Gamma \leftarrow \Gamma \bigcup \{\Phi[i] \& \Phi[j]\}$ ▷ Add conjunction
    end for
  end if
end for
return Γ

Mutual information

Mutual information (Hamming, 1986) between a feature T and an indicator variable Y_i of class c_i is a quantity $\text{MI}(T, Y_i)$ that measures the dependence between the two variables. It is calculated as:

$$\text{MI}(T, Y_i) = \sum_{t \in \{0,1\}} \sum_{y \in \{0,1\}} \Pr(t, y) \log \left(\frac{\Pr(t, y)}{\Pr(t) \Pr(y)} \right)$$

where the joint probability $\Pr(t, y)$ and the marginals $\Pr(t)$ and $\Pr(y)$ are estimated using relative frequencies. $\text{MI}(T, Y_i)$ is a positive quantity, with a value of zero if T and Y_i are independent. A large $\text{MI}(T, Y_i)$ score indicates that T is predictive for class c_i. We define:

$$\text{MI}_{sum}(T, Y) = \sum_{1 \le i \le k} \text{MI}(T, Y_i) \text{and}$$
$$\text{MI}_{max}(T, Y) = \max_{1 \le i \le k} \text{MI}(T, Y_i)$$

Information gain

The information gain of feature T with respect to Y_i measures the decrease in entropy when the feature T is present versus absent from the set of features. We evaluate the information gain of feature T with respect to class c_i as in (Yang and Pedersen, 1997):

$$\begin{aligned}\mathrm{IG}(T, Y_i) = &- \Pr(Y_i = 1) \log(\Pr(Y_i = 1)) \\ &+ \Pr(T) \Pr(Y_i = 1|T) \log(\Pr(Y_i = 1|T)) \\ &+ \Pr(\bar{T}) \Pr(Y_i = 1|\bar{T}) \log(\Pr(Y_i = 1|\bar{T}))\end{aligned}$$

Let:

$$\mathrm{IG}_{sum}(T, Y) = \sum_{1 \le i \le k} \mathrm{IG}(T, Y_i) \text{ and}$$
$$\mathrm{IG}_{max}(T, Y) = \max_{1 \le i \le k} \mathrm{IG}(T, Y_i)$$

Symmetric uncertainty

Symmetric uncertainty between term T and class indicator variable Y_i is a normalized mutual information score, computed as follows:

$$\mathrm{SU}(T, Y_i) = \frac{2\mathrm{MI}(T, Y_i)}{\mathrm{H}(T) + \mathrm{H}(Y_i)}$$

where by $\mathrm{H}(X)$ we denote the entropy of variable X, computed in practice as $\mathrm{H}(X) = \sum_{x \in X} -\Pr(x) \log(\Pr(x))$. The overall measure $\mathrm{SU}(T, Y)$ is given by:

$$\mathrm{SU}_{sum}(T, Y) = \sum_{1 \le i \le k} \mathrm{SU}(T, Y_i) \text{ and}$$
$$\mathrm{SU}_{max}(T, Y) = \max_{1 \le i \le k} \mathrm{SU}(T, Y_i)$$

Fisher test

Fisher's exact test (Fisher, 1928) is used to examine the significance of association between two binary variables. We apply it to the contingency table between feature T and class indicator Y_i and retrieve the significance p-value, which expresses the probability of the observed values of the table under the assumption of independence between the variables. If the probability is very small (i.e. p-value is small), then the independence assumption is rejected. We define the Fisher test score for feature selection as: $\mathrm{FT}(T, Y_i) = 1 - p$-value. $\mathrm{FT}(T, Y_i)$ always has values between 0 and 1. A typical threshold for significance is p-value < 0.05 or, more conservatively, p-value < 0.01.

The overall measure of relevance based on Fisher is:

$$\mathrm{FT}_{sum}(T, Y) = \sum_{1 \le i \le k} \mathrm{FT}(T, Y_i) \text{ and}$$
$$\mathrm{FT}_{max}(T, Y) = \max_{1 \le i \le k} \mathrm{FT}(T, Y_i)$$

2.2 Feature induction (generating conjunctions)

We include in the model feature conjunctions of maximum length m, which is a parameter of our method. At each iteration, the conjunctions are formed between atomic features that exceed some relevance threshold and any other feature or conjunction still present in the list of features.

Before adding a conjunction $T_i \& T_j$ to the the set Φ, we check if it has not been introduced already, for example as $T_j \& T_i$. If at a certain step all conjunctions that are generated are already in Φ, the algorithm stops (see step 8, Algorithm 1).

In typical applications, it is unlikely that very long conjunctions have a great impact on the classification performance. Therefore we suggest that m is kept small in practice, a value up to 3 should be sufficient for most applications.

2.3 Complexity of the feature induction algorithm

As we already argued in the introduction, computational complexity is one important bottleneck of feature selection and induction algorithms. Despite this, our method is fast. We run once through all samples in order to build the necessary data structures for evaluation of MI, IG, SU and FT scores, which takes $\mathcal{O}(pKN)$ time. The data structures esentially store the counts of samples in each class, for each feature. Thereafter, the complexity of scoring the set of p features and the sorting take place in $\mathcal{O}(pK)$ time, which is run m times. The overall time is thus $\mathcal{O}(pKm + pKN)$, which is $\mathcal{O}(pKN)$, in most applications. However, in practice our algorithm is even faster, because we use sparse vectors to represent each feature.

2.4 Model training and evaluation

We use the FITSI Algorithm for generating a set Φ of atomic features and conjunctions of length up to $m = 2$. We use these features to represent the data and train a model $\mathcal{M}$, which in our experiments can be a Perceptron or a Naive Bayes model. The performance of the algorithm clearly depends on the parameters used for the feature induction and selection: σ, k and l.

If we exclude the scoring measure σ, which can be chosen by the user based on some subjective criteria, our algorithm has two numerical hyper parameters that can be estimated from data, in a way that the resulting model has optimal performance. We use B-fold cross-validation for this purpose. We first split the data into two subsets, for parameter selection and for testing. The part that is used for parameter selection is split into

B bins (5 in our experiments). For each combination of parameters, we use $B-1$ bins for feature induction and model training and we evaluate the performance of the classifier on the remaining bin. In consequence, for each combination of parameters, a set of B performance estimates are obtained, which allows to compute mean and standard deviation. We identify the model with largest mean performance and select the simplest model (smallest k) that has the mean within one standard deviation from the best model. This is known as the one-standard-error rule, proposed by (Breiman et al., 1984). The parameters of this model are the optimal parameters k_{opt} and l_{opt}. We test this model on the excluded samples and report a test performance.

We compare the performance of our model to those of a baseline model. To this end, we repeat the cross validation described above, but we do not perform any feature induction.

For evaluating the performance of a classifier, we use either the F_1 measure, which is the harmonic mean of precision and recall.

3 Data

PA data: The "PA" dataset was developed by the Press Association[1] to enable the implementation of a system for recognition and semantic disambiguation of named entities in press releases. Given certain metadata for a number of overlapping candidate entities, an array of features derived from the textual context of their occurrence, and additional document-level metadata, the model recognizes which (if any) of the candidate entities is the one referenced in the text.

The corpus is annotated with respect to people, organization and location mentions; a special "negative" label denotes the candidates that can be considered irrelevant in the given context. In all cases, at most one of the overlapping candidates is annotated as positive. The dataset comprises a total of 2539 manually curated documents, and a total of 85602 concept mentions (this number represents the total of all candidate instances, including those annotated as non-entities).

For this dataset, the domain of the press releases is an important factor during classification, and specific features that express the belonging of a press release to a particular domain or category are also available. The dataset comprises articles from two domains: "General News" and "Olympics".

We remove non-location entity candidates, thus reducing the problem to the binary classification task of discerning locations from non-entities. We split the corpus into a training set (2369 documents) and a held-out test set (160 documents). As a result of this preprocessing, we have 2 classes ("Location" and "Negative"), 46273 instances, and a target to irrelevant instance counts ratio of 0.17.

From the training document set, we extracted 50455 atomic features.

As performance measure we report the F1 score of the positive class (i.e. 'Location').

4 Results

We performed feature induction using in turn all the measures of feature relevance mentioned in Section 2.1, followed by training Naive Bayes and Perceptron classifiers. The parameters k and l ($l > k$) of the feature induction step were iteratively selected from the set:

$$\{0.1\%, 0.25\%, 0.5\%, 0.75\%, 1\%, 2.5\%, 5\%, 7.5\%, 10\%, 25\%\}$$

of the total number of features p.

We used 5-fold cross-validation for selection of the optimal parameters k_{opt} and l_{opt}, as explained in section 2.4. Figure 2 illustrates the cross-validation search grid for the particular combination of feature induction with MI score and a Perceptron classifier. The intensity of the shade of gray is proportional to the average F1 measure over the 5 folds. The standard deviation for each combination of parameters is not shown in the image. The largest average F1 is 0.825 and is achieved for $k = 25\%$ and $l = 0.75\%$ of p.The standard deviation of this model is 0.009, estimated based on the 5 cross-validation folds. A simpler model, with $k = 25\%$ and $l = 0.1\%$ has average performance of 0.824, which is within one standard deviation from the maximum performance, hence there is no statistically significant difference between the two models. We thus chose the simpler model as optimal.

In Table 1, we show the F1 performance of the optimal models (determined by cross validation). The models that we investigated are various combinations of feature scoring measures (as rows) and classifiers (as columns). We compare

[1] http://www.pressassociation.com/

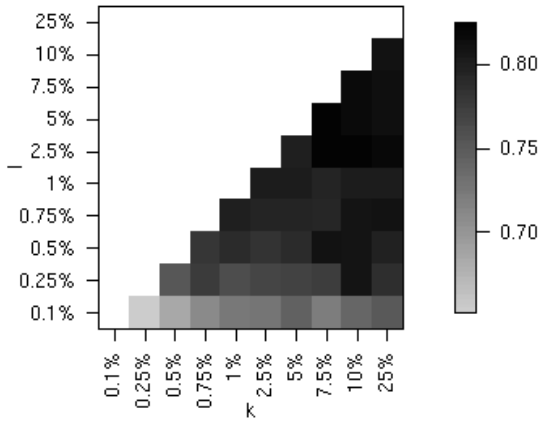

Figure 2: Parameter grid search by cross-validation.

	NB	**Perc**
MI	0.70	0.83
IG	0.70	0.83
SU	0.70	0.80
FT	0.75	0.83
Baseline	0.54	0.79

Table 1: Performance on the test dataset of classification models using various measures of feature relevance and comparison with the baseline.

	NB (k_{opt}, l_{opt})	**Perc** (k_{opt}, l_{opt})
MI	$25\%, 0.25\%$	$25\%, 0.1\%$
IG	$25\%, 0.25\%$	$25\%, 0.1\%$
SU	$25\%, 0.1\%$	$10\%, 0.1\%$
FT	$25\%, 0.1\%$	$10\%, 0.25\%$

Table 2: Parameters of the resulting models.

to a Baseline model (last row), which is either a Naive Bayes or Perceptron, without any feature induction. Clearly, all our models outperform the Baseline, by a large margin: up to 20% in the case of Naive Bayes models and up to 4% for Perceptrons. In general, Naive Bayes classifiers are worse than the Perceptron. Fisher test ranking appears to work best for Naive Bayes classifiers, whereas for Perceptron models achieve similar performance for most of the scoring measures (apart from Symmetrical Uncertainty).

In Table 2, we show the k_{opt} and l_{opt} that we estimated by cross validation.

Table 2 shows for each model the optimal parameters k_{opt} and l_{opt} that were selected via cross validation. In general many atomic features are included in the model, indicated by the large values of k_opt, which are in general 25%. Only two models select 10% features, namely the Perceptron with FT and SU. The most common values for l_{opt} are 0.25% and 0.1%, which means that the useful conjunctions are those that comprise at least one high scoring atomic feature (from top 0.01% or 0.25%). In contrast, a larger l_{opt} would mean that the model benefits from conjunctions between two low scoring atomic features, which is not the case, according to our results.

5 Analysis of conjunctions

The purpose of feature induction is to generate useful combinations of features without the help of an expert in the domain of the application. After we trained our models with the optimal parameters determined by the quantitative results, we analyzed the conjunctions that were ranked highest final models.

Below we comment on the (types) of conjunctions that have the highest rank according to the Perceptron model using the MI criterion for feature ranking.

- `http://www.geonames.org/ontology#P.PPLC & ANNIES=location_city`

  The Geonames ontology[2] indicates that the candidate is a capital and Gate's ANNIE[3] suggests that the instance is a city. Therefore, the conjunction reinforces the recommendation for a location. The conjucntion is very informative.

- `http://www.geonames.org/ontology#P.PPLC & MOST_PROBABLE=true`

  As before, the Geonames Ontology suggests a capital and the respective location is also indicated as most probable label of the entity. The decision wheather an annotation is most probable for a candidate or not is made as follows. For each candidate with several possible annotations, the experts from Press Association rank the annotations by how likely they are. If a candidate has 'location' as most likely annotation, the value of the feature 'MOST_PROBABLE=true' is 1, otherwise 0.

- `MOST_PROBABLE=true & ANNIES=location_city`

  The most probable label of the entity is Location and Gate's ANNIE suggests a city.

[2] www.geonames.org
[3] http://gate.ac.uk/

- `NoCandidates=1 & ANNIES=location_city`

  There is only one candidate for the given entity and Gate's ANNIE suggests city.

- `http://www.geonames.org/ontology#P.PPLC & NoCandidates=1`

  Similar to the conjunction above, if there are no other suggestions for the specific entity and Geonames suggests a populated place, then there is a strong indication for Location.

- `PrevWord = "in" & MOST_PROBABLE=true`

  If the word preceding the candidate is 'in' and Location is most probable label of the entity, then there is a strong indication that the entity is indeed a Location.

- `PrevWord = "in" & NoCandidates=1`

  The conjunction between previous word being 'in' and the absence of other candidates at the specific location is a strong indicator for Location. This is a linguistic pattern that most experts would add to the model. Our algorithm automatically generates this pattern.

- `ANNIES=location_country & pacategory:Olympics`

  A very interesting domain-specific conjunction is formed by the indication of ANNIE to a country and the category of the document being Olympics. Even though ANNIE points to a country, the fact that the document belongs to the Olympics category makes the Location less likely, because the candidate is most probably referring to a team. Such conjunctions are specific to domain adaptation tasks and our algorithm generates it automatically, without defining a domain adaptation problem explicitly. Only adding atomic domain features allows for generating of domain specific-conjunctions.

- `CountOverlapping=1 & ChunkLength=2`

  There are no overlapping candidates with the target candidate and the length of the candidate is two seems to be a strong indicator for Location.

- `multiple conjunctions including the domain name`

  Our algorithm ranks high various conjunctions that include the domain of the document. As commented already above, our approach seems to implicitly perform domain adaptation, by adding conjunctions between features that play different roles in different domains and the respective domain features (very similar to the approach of (Daumé, 2009)).

6 Discussion and future work

We introduced a greedy heuristic for feature selection and induction. The method is applied as a preprocessing step, prior to model fitting, therefore it is independent from the classifier chosen by the user. It is very fast in practice, having all the advantages of the filter-based methods over complex wrappers and embedded methods.

We applied the method on a custom dataset from Press Association, for named entity disambiguation. In particular, we recognized Locations from negative entities. The results, presented in the form of F1 measure corresponding to the Location class, show great improvements over the baseline.

We provided with a qualitative analysis of some of the highest ranking conjunctions and they appear to be strong predictors for Location, that a domain expert would also consider adding to the model.

A limitation of our method is the univariate filter of the features, which might eliminates features that are not univariately strongly associated with the outcome, but only in combinations. Even though our results suggest that valuable conjunctions tend to include at least one highly-ranked atomic feature, we cannot generalize to all applications. Therefore, we think that embedded methods such as logistic regression with Lasso penalty, albeit slower in practice, could improve feature ranking.

While running experiments, we observed that a strongly predictive atomic feature tends to generate many conjunctions that score very high themselves, being very correlated with the original feature. The resulting conjunctions then occupy a good rank, on the expense of other atomic features or conjunctions which are eliminated from the model (due to the selection threshold). The eliminated features, even if they yield smaller scores, provide with essentially distinct information that helps the model as a whole, unlike the correlated, high-scoring conjunctions. An embedded feature selection such those based on Lasso penalties for example would eliminate correlations and retain

the relevant information, therefore we will pursue such approach in our future work.

References

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. 1984. *Classification and Regression Trees*. Chapman and Hall/CRC.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

H. Daumé. 2009. Frustratingly easy domain adaptation. *CoRR*, abs/0907.1815.

Asif Ekbal, Sriparna Saha, Olga Uryupina, and Massimo Poesio. 2011. Multiobjective simulated annealing based approach for feature selection in anaphora resolution. In *DAARC*, pages 47–58.

R.A. Fisher. 1928. *Statistical methods for research workers*. Oliver and Boyd.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Patrick Haffner, Steven J. Phillips, and Robert E. Schapire. 2005. Efficient multiclass implementations of l1-regularized maximum entropy. *CoRR*, abs/cs/0506101.

Richard W. Hamming. 1986. *Coding and Information Theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Noppadon Kamolvilassatian. 2002. Property-based feature engineering and selection. Master's thesis, Department of Computer Sciences, University of Texas at Austin.

Ron Kohavi and George H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–324.

John Lafferty. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289. Morgan Kaufmann.

Changki Lee and Gary Geunbae Lee. 2006. Information gain and divergence-based feature selection for machine learning-based text categorization. (1):155–165.

Chang-Hwan Lee, Fernando Gutierrez, and Dejing Dou. 2011. Calculating feature weights in naive bayes with kullback-leibler measure. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 1146–1151.

David D. Lewis. 1992. Feature selection and feature extraction for text categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217. Morgan Kaufmann.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields.

A Moh'd A Mesleh. 2007. Chi square feature extraction based svms arabic language text categorization system.

P. M. Narendra and K. Fukunaga. 1977. A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.*, 26(9):917–922.

F. Rosenblatt. 1957. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Cornell Aeronautical Laboratory report.

Karl-Michael Schneider. 2004. A new feature selection score for multinomial naive bayes text classification based on kl-divergence. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, ACLdemo '04.

Hirotoshi Taira and Masahiko Haruno. 1999. Feature selection in svm text categorization. In *Proceedings of the AAAI '99*, pages 480–486.

Harun Uguz. 2011. A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. *Knowledge-Based Systems*, 24(7):1024–1032.

Celine Vens and Fabrizio Costa. 2011. Random forest based feature induction. In *ICDM*, pages 744–753.

Jihoon Yang and Vasant Honavar. 1998. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(2):44–49.

Yiming Yang and Jan O. Pedersen. 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.