Two Machine Learning Approaches to Coreference Resolution

Dinh Le Thanh

PFL054 2008/09

Table of contents

I. Introduction

Coreference resolution is a process of determining whether two expressions in natural language refer to the same entity in the world. For example: “Father claimed disliking opera-going. He was mainly angry with the style of opera singing.” The words “father” and “he” denote the same individual – a father of the major hero from the selected book. Both words take part in the process called reference, where you can use more different expressions (words, phrases) to refer to individuals, subjects or situations of the real world. A natural language expression used to perform reference is called a referring expression, and the entity that is referred to is called the referent. Thus, father and he are referring expressions, and the real-word man is their referent. Two referring expressions that are used to refer to the same entity are said to corefer, the relationship between them is coreference and these two expressions form a coreferential pair. Usually the second expression in a coreference pair (he) is called anaphora, the first one (father) is antecedent.

The knowledge of coreferential pairs is an important part of many applications in the natural language processing – e.g. information retrieval, document summarization, machine translation. Since the volume of texts is enormous so it is practically impossible for human beings to go through them manually. Therefore, the need of an automatic procedure which is capable to detect coreferential pairs in the text is necessary. There were several experiments dealing with the automatic coreference resolution using rule-based approaches [5, 6]. However, recently machine learning approaches have become more attractive given that such methods overcome many difficulties in rules-based approaches, such as: their adaptability and flexibility to different contexts and cost of maintenance.

Thus in this report, two machine learning methods are described and experimented in order to solve the coreference resolution problem. The first method is Naïve Bayes classifier which has been proven of being successfully applied to many natural language processing tasks, i.e text categorization, question answering… The second method, Decision Tree, is also a classical machine learning approach yet showed very sufficiently. The aim of this work is two-folds: first, given the data and set of extracted features, find the best model of each method (in the sense of accuracy and f-measure by choosing the combination of features and tuning the classifier’s parameters). Second, the comparisons between two approaches are implemented using significant tests: paired t test with cross-validation and bootstrap.

II. Data and features description

Data are extracted from the Prague Dependency TreeBank 2.0 (PDT 2.0) and already prepared in the features format. Originally, the PDT 2.0 contains 3168 news paper annotated at the tectogrammatical level and consists of 49431 sentences. Coreference was annotated manually in all this data. There are 45631 coreference links (counting both textual and grammatical ones) [5, 6].

Hence, it should be noted that the data used in this work is just a subset from the original data in PDT and from now on, we will only mention about this subset data which was used for this experiment work.
Data is divided into the train and evaluation (test) sets. Each row in data files corresponds to one word pair. In case of anaphora and candidate-antecedent, the pair is classified as being coreferential, i.e. a positive instance. The data are prepared to apply machine learning methods so there are noncoreferential (i.e. negative) instances as well (i.e. a pair of anaphora and candidate-non-antecedent). There is at least one noncoreferential pair for each coreferential pair (and vice versa) in the data. There are around 14% coreferential pairs out of all pairs in the data. Following is the detailed table of the data size and its proportion of coreferential pair:

	No of pairs	No of coreferential pair	No of non-coreferential pair
Training set	10001	1342	8659
Testing set	3009	396	2613
Total	13010	1738	11272

Table 1: Coreferential pair proportions in the data.

There are 55 comma-separated values (features) on each row. The first 54 values are features which are served for classification purposes. Values and names of the features are described in the below table 2. The very first feature is a technical one and serves as an indicator of the anaphora. The rest of the features are divided into the categorical and continuous type. All possible values of the categorical features are also listed in table 2.

Index	Feature	Type	Values	Description
1	anaph_id	-	-	ID of anaphora. For one value of ID there have more than one candidate.
2	cand_gen	categorical	neut, inher, empty, anim, fem, inan, nr	Gender, number for candidate and anaphora + agreement of pair in gender and number.
3	cand_num	categorical	inher, sg, empty, pl, nr	
4	anaph_gen	categorical	neut, inan, anim, fem, nr	
5	anaph_num	categorical	sg, pl, nr	
6	gen_agree	categorical	1, 0	
7	num_agree	categorical	1, 0	
8	cand_coord	categorical	1, 0	Is candidate in coordination?
9	app_in_coord	categorical	1, 0	1 ONLY IFF anaphora is APP, anaphora+candidate is within the same clause and they have same (pre)ancestors CONJ|DISJ
10	cand_epar_fun	categorical	MEANS, PRED, TSIN, TTILL, …	Some properties of candidate and anaphora such as: functor, semanic POS, agreement within functors (or semPOS or lemmas)
11	cand_epar_sempos	categorical	advspec_dotdenotspec_dotgradspec_dotnneg, advspec_dotdenotspec_dotgradspec_dotneg, adjspec_dotquantspec_dotindef, …	
12	anaph_epar_fun	categorical	MEANS, PRED, TSIN, BEN, …	
13	anaph_epar_sempos	categorical	advspec_dotdenotspec_dotgradspec_dotneg, nspec_dotquantspec_dotdef, adjspec_dotdenot, v, …	
14	epar_fun_agree	categorical	1, 0	
15	epar_sempos_agree	categorical	1, 0	
16	epar_lemma_agree	categorical	1, 0	

17	cand_fun	categorical	MEANS, TSIN, TTILL, BEN, …	Functor (both for candidate and anaphora) on T- and A-layear + agreement
18	anaph_fun	categorical	MEANS, BEN, HER, LOC, …	
19	fun_agree	categorical	1, 0	
20	cand_afun	categorical	AtvV, AuxC, Pnom, AdvAtr, …	
21	anaph_afun	categorical	Pnom, AdvAtr, AtrAtr, ExD, …	
22	afun_agree	categorical	1, 0	
23	cand_apos	categorical	A, J, T, N, P, C, D, R, I, empty	Some properties of candidate and anaphora from a-tag (from 1st to 8th tag position)
24	anaph_apos	categorical	R, P, empty	
25	cand_asubpos	categorical	S, T, N, K, 7, 2, E, …	
26	anaph_asubpos	categorical	H, S, R, P, empty, V, 5	
27	cand_agen	categorical	F, X, undef, N, Y, H, Z, M, I, empty	
28	anaph_agen	categorical	F, undef, N, X, Y, Z, M, I, empty	
29	cand_anum	categorical	S, D, undef, X, P, empty	
30	anaph_anum	categorical	S, undef, X, P, empty	
31	cand_acase	categorical	6, undef, 3, X, 7, 2, 4, 1, empty, 5	
32	anaph_acase	categorical	6, 3, X, 7, 2, 1, 4, empty	
33	cand_apossgen	categorical	Z, F, M, X, undef, empty	
34	anaph_apossgen	categorical	F, Z, undef, X, empty	
35	cand_apossnum	categorical	S, undef, P, empty	
36	anaph_apossnum	categorical	S, undef, P, empty	
37	cand_apers	categorical	3, undef, empty	
38	anaph_apers	categorical	undef, 3, empty	
39	cand_akt	categorical	1, 0	Info about being actants + agreement
40	anaph_akt	categorical	1, 0	
41	akt_agree	categorical	1, 0	
42	cand_subj	categorical	1, 0	Info about being subject + agreement
43	anaph_subj	categorical	1, 0	
44	subj_agree	categorical	1, 0	
45	sent_dist	categorical	1, 0	Order of candidate, the closest ones to anaphora have 0.
46	clause_dist	continuous		
47	file_deepord_dist	continuous		
48	cand_ord	continuous		
49	cand_tfa	categorical	c, f, t	Topic-focus + agreement
50	anaph_tfa	categorical	c, f, t	
51	tfa_agree	categorical	1, 0	
52	sibl	categorical	1, 0	Are candidate and anaphora siblings?
53	coll	categorical	1, 0	Are they within the same colocation?
54	cand_freq	categorical	1, 0	Is the frequency of candidate more than 1 (within the whole corpus)?
55	KOREF	categorical	1, 0	Aimed variable, info about coreference/uncoreference pair.

Table 2: Feature descriptions

Following is an example taken from the data which describes two instances: noncoreferential and coreferential for the same anaphora:

tundefcmpr9410undef001undefp10s2a0, fem, sg, fem, sg, 1, 1, 0, 0, PAT, v, CNCS, v, 0, 1, 0, PAT, ACT, 0, Obj, empty, 0, N, empty, N, empty, F, empty, S, empty, 4, empty, undef, empty, undef, empty, undef, empty, 1, 1, 1, 0, 1, 0, 1, 1, 11, 3, f, t, 0, 0, 0, 0, 0.

tundefcmpr9410undef001undefp10s2a0, fem, sg, fem, sg, 1, 1, 0, 0, RSTR, v, CNCS, v, 0, 1, 0, ACT, ACT, 1, Sb, empty, 0, P, empty, 4, empty, F, empty, S, empty, 1, empty, undef, empty, undef, empty, undef, empty, 1, 1, 1, 1, 1, 1, 1, 1, 15, 4, t, t, 1, 0, 0, 0, 1.

There are two issues concerning about the data and the algorithms. From the above table, one can see that data contains both categorical and continuous attributes in which most of them are categorical. This is one reason why Naïve Bayes and Decision Tree are chosen as classifiers since they are most suitable in handling such attributes. However, another issue raised by the discrete data when further analyzing is that some categories have no instances. This means that the algorithms would have to be able to handle disjunctions of expressions, where each expression consists of conjunctions of attributes. For decision tree, it can be easily handled by its nature while with naïve bayes, we should consider the smoothing technique (described later) to prevent zero probabilities problem.

III. Classifiers

III.1. Decision Tree

Decision tree is an inductive inference algorithm and which approximates the target function. It can also be considered as a set of if-then rules based on the features values of the data. The classification is therefore obtained by traversing the tree starting from the root node to its leaves. Each node is corresponded to an attribute and each edge corresponds to a value of an attribute. A leaf determines a classification of an instance and hence the target function is discrete. This algorithm is proven robust to the noisy data. Moreover, it does not need to contain all the attribute values hence can handle disjunction of conjunctions of attributes.

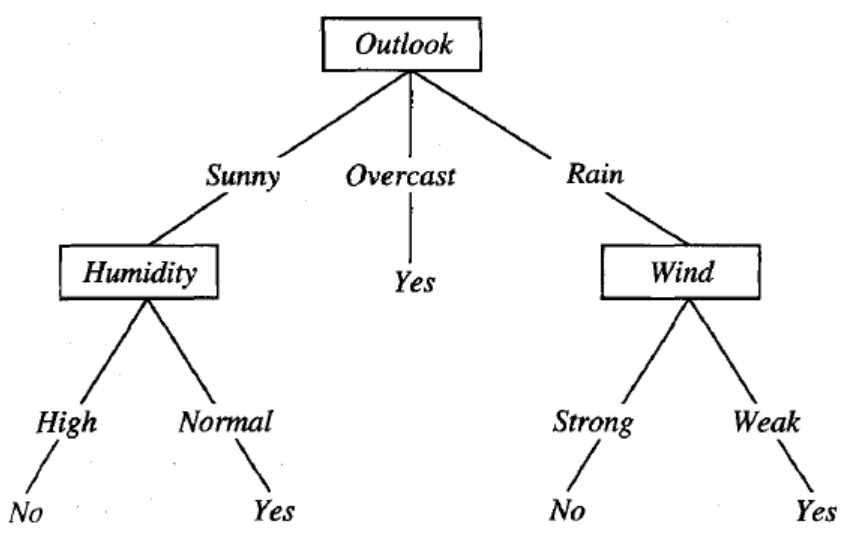

Figure 1: An example of decision tree

The basic decision tree algorithm is started by building the root node which corresponds to all the training examples. Then following the “top-down” procedure, the children are added according to the attributes of the training data. There is one key question in the algorithm is that “which attribute is the best choice for a given node”. Depending on existing algorithms, several techniques are used in which the most popular ones are “Gini impurity” (CART algorithm) and “Information Gain” (ID3, C4.5, C5.0). Gini impurity is based on squared probabilities of membership for each target category in the node. It reaches its minimum (zero) when all cases in the node fall into a single target category.

Suppose y takes on values in {1, 2, ..., m}, and let f(i, j) = probability of getting value j in node i. That is, f(i, j) is the proportion of records assigned to node i for which y = j.

$$I_G(i) = 1 - \sum_{j=1}^{m} f(i,j)^2 = \sum_{j \neq k} f(i,j) f(i,k)$$

Information gain is based on the concept of entropy used in information theory.

Gain(D,A) = Entropy(D) - ΣvЄValues(A)(|Dv|/|D|)Entropy(Dv)

where

D = training data

Dv = {d Є D; A(d) = v}

A = attribute

Values(A) = set of all possible values of attribute A

Entropy(D) = -Σ pi log2 pi

Following is an example of using the “Information gain” to decide which attribute will be chosen in the next step to build the decision tree [1]

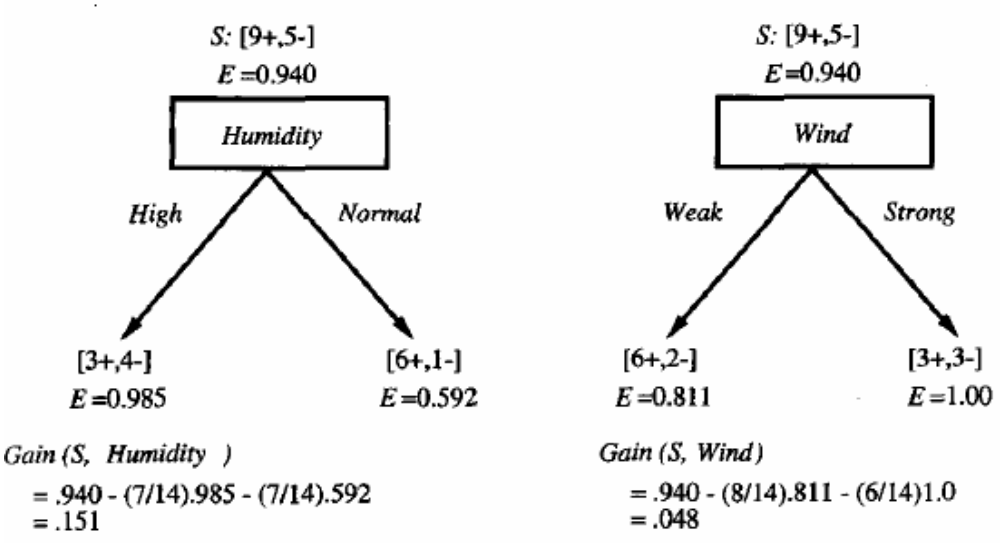

Figure 2: Humidity provides greater information gain than Wind, relative to the target function. Hence, humidity will be chosen as the next attribute in the tree.

There are some advantages of using decision tree for classification:

- Simple to understand and interpret.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
- Able to handle both numerical and categorical data.
- Use a white box model. If a given situation is observable in a model the explanation for the condition is easily explained by Boolean logic. An example of a black box model is an artificial neural network since the explanation for the results is excessively complex to be comprehended.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Robust, perform well with large data in a short time. Large amounts of data can be analyzed using personal computers in a time short enough to enable stakeholders to take decisions based on its analysis.

III.2. Naïve Bayes

Naïve Bayes is also an inductive inference algorithm and suitable when each instance x is represented as a conjunction of attribute values and the target function f(x) can take any value from a finite set V. A set of training examples along with the target function are provided and a new instance is presented by a

tuple of attribute values <a1, a2,.., an>. The learner is then asked to classify this new instance.

For the Bayesian approach, it will assign the new instance to the most probable target value, v_{MAP} as following:

$$v_{MAP} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, .., a_n)$$

By using Bayes theorem, we can rewrite this as:

$$v_{MAP} = \arg\max_{v_j \in V} \frac{P(a_1, a_2, .., a_n \mid v_j) P(v_j)}{P(a_1, a_2, .., a_n)}$$

$$v_{MAP} = \arg\max_{v_j \in V} P(a_1, a_2, .., a_n \mid v_j) P(v_j)$$

It is easy to estimate P(vj) simply by counting the frequency of vj occurs in the training data. However, estimating P(a1, a2,.., an) in this fashion is not feasible unless having the very large set of training data.

Therefore, the Naïve Bayes classifier deals with this problem by simply assuming that the attribute values are conditionally independent given the target value. In other words, the assumption is that, the probability of observing the conjunction <a1, a2,.., an> is just the product of the probabilities of the individual attributes: P(a1, a2,.., an) = ∏ P(ai|vj). Substituting this into the above equation, we have the Naïve Bayes classifer:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

The probabilities in the Naïve Bayes model could be also easily estimated by counting the frequency (log likelihood) as mentioned above. However, this may raise two difficulties. First, it produces a biased underestimate of the probability. Second, when this probability estimate is zero, this will dominate the whole model even for some non-zero probabilities in the future. To overcome this problem, we can adopt a Bayesian approach to estimating the probability, using m-estimate (Laplace) as follows:

$$\frac{n_c + mp}{n + m}$$

where n correspond to the total number of time that vj appears in the data and nc is the number of times of vj in the presence of the attribute value ai, p

corresponds to a prior estimation of the probability 1/k (where k is the number of possible values) [1].

There are some advantages of using Naïve Bayes classifer:

- The Naive Bayes algorithm affords fast, highly scalable model building and scoring.
- It scales linearly with the number of predictors and rows.
- It requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.
- Naive Bayes can be used for both binary and multiclass classification problems.

IV. Implementation

The entire implementation was done in R since it has a powerful support for data manipulation, fast development and predefined libraries. There are two libraries were used for the experiments: rpart (decision tree) and e1071 (naïve bayes).

For clarity, the source scripts was implemented and organized based on the main tasks and process of the problem. The following table presents the organization of the implementation script files and their brief descriptions.

Script R file	Description
data_preparation.R	Load the data from “etest.csv” and “train.csv” files. Also tranform the attributes into continuous/categorical according to “all.names” file.
decision_tree.R	Provide a function to train the loaded training data and classify the loaded testing data using rpart library.
naïve_bayes.R	Provide a function to train the loaded training data and classify the loaded testing data using e1071 library.
feature_selection.R	Test the combinations of features to find the “best” one. Two strategies are used: accumulated and individually.
parameter_tuning.R	With the found best features, test each classifier model with different parameters (i.e laplace for naïve bayes and complexity for decision tree) in order to find the optimal parameter.

hypothesis_testing.R	With the above best models, compare their performances by using hypothesis testing. Two strategies are used: confidence interval using cross-validation and confidence interval using bootstrap.
demo.R	A script file contains some demos how to run the experiments

Table 3: Organization and descriptions of script files

Since the source scripts are rather long, the detailed information of each script will not be mentioned here. For further details of the algorithms and ideas, one should refer to the actual script files by following the source code and comments.

V. Experiments and results

A number of experiments were conducted for both algorithms in order to find the best set of features, best parameter for each model and also to test the significance of their performances. The next sections will describe and analyze the experiments and results in more details.

V.1. Feature selection

The objective of feature selection is to reduce the number of features used to characterize a dataset so as to improve an algorithm's performance on a given task [7]. Besides, feature selection is necessary because of [3]:

- Alleviating the effect of the curse of dimensionality.
- Enhancing generalization capability.
- Speeding up learning process.
- Improving model interpretability. It also helps people to acquire better understanding about their data by telling them which are the important features and how they are related with each other.

In general, feature selection algorithms have three components [7]:

- Search algorithm: It searches the space of feature subsets, which has the size of 2^d where d is the number of features.
- Evaluation function: Its input is a subset of features and output a numeric evaluation. The search's algorithm goal is to maximize this function.
- Classifier: It is the target algorithm that uses the final subset of features (i.e., those found by the search algorithm to have the highest evaluation).

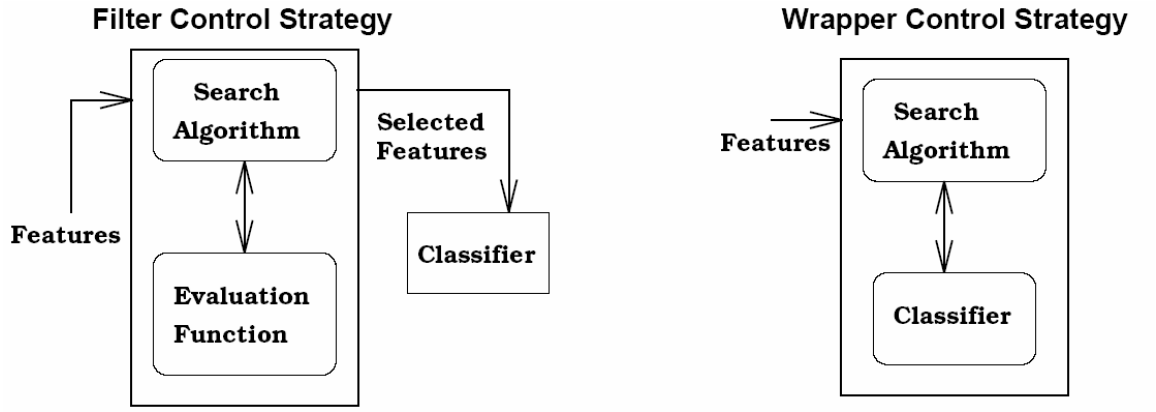

Figure 3: Common control strategies for feature selection approaches

Typically, the components interact in one of the two ways described in figure 3. Either the search and evaluation algorithms locate the set of features before the classifier is consulted (filter control strategy), or the classifier itself the evaluation function (wrapper control strategy). John, Kohavi and Pfleger (1994) argued that the wrapper strategy is superior because it avoids the problem of using an evaluation function whose bias differ from the classifier [7].

Following Doak (1992), there are three types of search algorithms: exponential, sequential and randomized. Exponential algorithms have exponential complexity in number of features. Sequential searches have polynomial complexity; they add or remove features by using hill climbing search strategy. Randomized algorithms have genetic search methods. By the sake of complexity and well performance proven, this work will concentrate only to sequential search.

The most common sequential search algorithms are Forward Sequential Search (FSS) and Backward Sequential Search (BSS). FSS begins with zero attribute, evaluates all feature subsets with only one feature, and selects the one with the best output of evaluation function. It then adds to this subset the feature that yields the best output of evaluation function for subsets of the next larger size. And this procedure is repeated until no improvement is obtained or there are no features left (the output of FSS is the original set of features). Conversely, BSS starts with all the features and repeatedly remove the feature that maximal increase the performance when removed.

For the experiments of this work, the combination of FSS algorithm with wrapper control strategy was chosen as the implementation of feature selection. In particular, the feature selection algorithm begins with zero attribute; evaluate all one-feature subsets, then selects the one with the best f-measure tested by the classifier. In the next step, it again adds to this subset the feature that the classifier when testing the combination of the previous step subset and this feature will output the best f-measure. This cycle is repeated until no improvement is found or no features left. The detail of the algorithm is described in the below figure:

```
# Feature selection algorithm


S = {}; # Set of the best output features
max_fmeasure = 0; # best f-measure for each iteration step
step = 0


# iteratively adding features, stop only no features left or no improvement found
while (step < number_of_features)
      F = {} # set of output f-measures by classifier
      for (each feature f which has not been chosen)
            f_measure = classifier(S U f)
            F = F U {f_measure}


      # if there is no improvement, stop and output the result
      if (max(F) <= max_fmeasure) break


      # otherwise, adjoin this new feature
      S = S U {feature with max(F)}
      max_fmeasure = max(F)


      step = step + 1
return S
```

Figure 4: Feature selection algorithm

The output of the algorithm is the set of 15 features which are: gen_agree (6), cand_fun (17), cand_asubpos (25), num_agree (7), clause_dist (46), file_deepord_dist (47), sibl (52), coll (53), anaph_gen (4), anaph_akt (40), anaph_tfa (50), subj_agree (44), afun_agree (22), cand_epar_sempos (11), cand_freq (54). The best performance is achieved by Naïve Bayes classifier with f-measure of 71%.

Accuracy	Precision	Recall	F-Measure	Confusion matrix			
						predict	
						1	0
0.918	0.67	0.75	0.707	real	1	298	98
					0	148	2465

Table 4: The best result achieved by 15 features and Naïve Bayes classifier

In order to justify the performance of the algorithm, two baseline metrics were also implemented. The first metric is the performance score of using all 54 features. For the second metric, 15 features are chosen randomly from the 54 features and repeated with replacement 100 times, and then the set with the highest f-measure was chosen to compute as the second baseline score. The comparisons are given in the below table:

	Accuracy	Precision	Recall	F-Measure
Naïve Bayes - all features	0.846	0.45	0.72	0.552
Naïve Bayes - random features	0.896	0.59	0.69	0.635
Naïve Bayes - best 15 features	0.918	0.67	0.75	0.707
Decision Tree - all features	0.834	0.41	0.60	0.49
Decision Tree – random features	0.898	0.61	0.63	0.618
Decision Tree – best 15 features	0.916	0.72	0.60	0.652

Table 5: Performance comparisons of using different feature sets

From the table, we can see that the result of 15 features achieved by the feature selection algorithm outperforms the others and hence proves the efficiency of the algorithm. Interestingly, the results from random features showed that random features work much better than using all the features. With larger number of sample times (currently 100), its performance would not only overcome the best current 15 features but also converge to the optimal set of features.

V.2. Parameter tuning

Apparently, features play an important role in affecting the performance of classifiers. However, parameters of each method can also influence these measures. Hence, the aim of the experiments in this section is to find the best parameter for each method and verify how much impact of the parameter on the models. In particular, for decision tree, one useful parameter is pruning which is defined by prune function in R. For Naïve Bayes, one is laplace threshold. To run the experiments, a set of plausible parameters will be tested for each model and finally, the one which gives the best performance will be chosen. Followings are the details of the experiments.

Complexity (cp)	F-Measure
0.05	0.644
0.02	0.644
0.01	0.651
0.005	0.654
0.002	0.633
0.001	0.634
0.0001	0.646

Table 6: Performances of decision tree with different complexity parameters

Laplace	F-Measure
0	0.708
10	0.674
20	0.664
30	0.614
40	0.536
50	0.464
60	0.394
70	0.360
80	0.331

Table 7: Performance of naïve bayes with different laplace parameters

From the above tables, the complexity for decision tree is chosen with the value of 0.005 and laplace for naïve bayes is 0. Followings are the confusion matrices of the best found models so far using test set.

Accuracy	Precision	Recall	F-Measure	Confusion matrix			
0.918	0.67	0.75	0.707			predict	
						1	0
				real	1	298	98
					0	148	2465

Table 8: Naïve Bayes performance using best 15 features and laplace parameter 0

Accuracy	Precision	Recall	F-Measure	Confusion matrix			
0.904	0.62	0.69	0.654			predict	
						1	0
				real	1	273	123
					0	165	2448

Table 9: decision tree performance using best 15 features and complexity parameter 0.005

It should be noted that decision tree is not only useful in classifying, but it also offers a graphical of the classification. This functionality of decision tree is extremely helpful when analyzing the data, and also offers a tool to see which features are proven to be most important for classification. For example, the below figure shows the representation of the decision tree model using the best 15 features, complexity cp = 0.005 and it could be clearly seen that the most useful attributes are: gen_agree, cand_fun and num_agree.

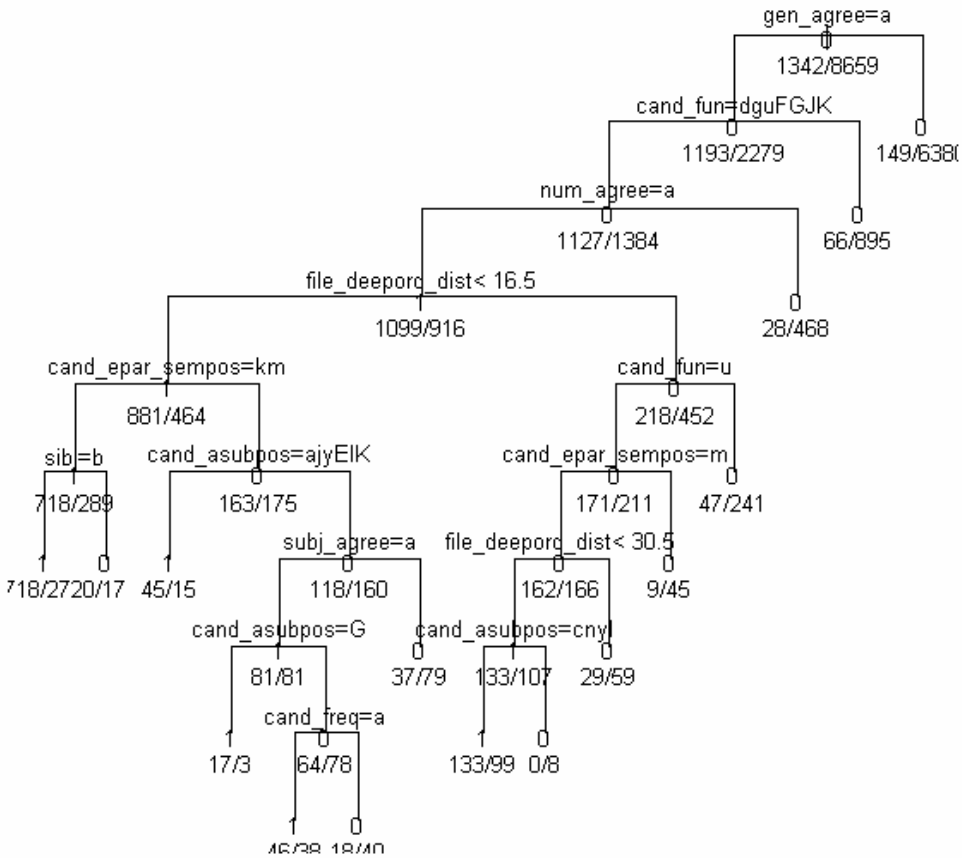

Figure 5: graphical representation of decision tree using 15 features and complexity parameter cp = 0.005

V.3. Performance comparisons

So far, the best model and parameter for each method were assumed. However, in order to compare the performances of the two methods, those information are not enough and reliable. There are two experiment methods implemented so that the confidence intervals of f-measure for each model will be defined. Hence it will provide a better measure to determine which algorithm outperforms the other.

V.3.1. Paired test using cross-validation

In this experiment, the training data is repeatedly partitioned into disjoint training and test sets and to take the mean of the test set f-measures for comparison and finding the confidence interval. The procedure first partitions the training data into

k disjoint subsets of equal size, where the size is at least 30. It then train trains and tests the learning algorithm k times, using each of the k subset in turn as the test set, and using all the remaining data as the training set. In this way, the learning algorithms are tested on k independent test sets, and the mean difference in f-measure is returned as an estimate of the difference between the two learning algorithms.

$$\bar{\delta} = \frac{1}{k}\sum_{1}^{k}\delta_i$$

where б indicates the difference in f-measure between two algorithms

The approximate N% confidence interval for estimating the difference is then given by:

$$\bar{\delta} \pm t_{N,k-1} s_{\bar{\delta}}$$

where

$$s_{\bar{\delta}} = \sqrt{\frac{1}{k(k-1)}\sum_{1}^{k}(\delta_i - \bar{\delta})^2}$$

For the experiments, k is set to 10 and significant level is chosen 95%. The experiment results are described in the below table:

CI of Naïve Bayes f-measure	(0.657, 0.692)
CI of decision tree f-measure	(0.635, 0.676)
CI of the difference f-measure of two methods	(-0.02, -0.01)

Table 10: Confidence intervals of two algorithms using paired test.

From the table, one can see that at 95% of confidence, there is a significant difference of f-measure between two algorithms and hence it showed that Naïve Bayes classifier outperforms decision tree in classification task.

V.3.2. Bootstrap test

In this experiment, the two models are firstly trained using the training set. Then, each model was tested using 1000 subsets of size 500 which are taken randomly with replacement from the test set. The output results of the differences in accuracies/f-measures are stored in a vector. Finally, the vector is sorted and the elements 25th and 975th are obtained as the confidence interval with confidence level of 95%. The output results are in the below table:

CI of Naïve Bayes f-measure	(0.476, 0.880)
CI of decision tree f-measure	(0.416, 0.838)
CI of the difference f-measure of two methods	(-0.2, 0.07)

Table 11 Confidence intervals of two algorithms using bootstrap test.

In contrast with the paired-test result, there was no statistical significance in the difference of f-measure between two methods since the interval includes the value 0. It means their performances are considered equal. The results also shows that the width of confidence interval of each method is much wider compared to one using paired test. This could be explained by the small proportion of coreferential pairs in the test set (396 coreferential pairs out of 3009) so that it is more likely to sample a set with very few coreferential pairs and for each wrong/correct pair classified, the performance of classifier will have very big impact.

V.4. Conclusion

In summary, the work has dealt with the following main tasks:

- Solve the problem of coreference resolution using two machine learning classifers: naïve bayes and decision tree.
- Find the “best” combination of features out of 54 original features.
- Obtain the best parameter for each model. For decision tree, represent the graphical tree in order to find the most useful attributes.
- Compare the performance of the two algorithms.

The experiment results have shown that, the two algorithms were successful applied and well performed. The 15 output combination features helped boost up remarkably the performances of algorithms. For Naïve Bayes, the highest f-measure is 71%, paired-test confidence interval of (0.657, 0.692) and bootstrap confidence interval of (0.476, 0.880). For decision tree, the highest f-measure is 65%, paired-test confidence interval of (0.635, 0.676) and bootstrap confidence interval of (0.416, 0.838). However, we cannot conclude that this is the best set of features and though there may need more experiments or different algorithms to improve the result. It can also be seen that, the role of parameters for each model did not help much in improving the performance.

For comparison between two algorithms, the results showed that Naïve Bayes performs better than decision tree in the coreference resolution classification task. Though in bootstrap test, it is concluded that there is no difference between two methods, but by the results of paired test in addition with looking at the confidence intervals of both method, this claim could be easily seen. However, because the difference is rather small and also the experiments are only tested

by one set of features, it is hard to conclude naïve bayes outperforms decision tree for this task in general. For more reliable conclusion, I believe there should have more experiments with different set of features as well as with bigger data.

In future, other interesting tasks could be experiment with other algorithms, such as k nearest neighbors or support vector machines, in order to find the best algorithm suitable for the problem. The more experiments with feature combination should also need to be conducted to improve the result.

References

[1] *Mitchell, Tom*. **M., Machine Learning**, 1997.

[2] *Emmanuel Paradis*, **R for beginners**, Institutes des Sciences de l'Evolution, Universite Montpellier II.

[3] *http://en.wikipedia.org/wiki/Feature_selection*

[4] *http://ufal.mff.cuni.cz/~hladka/project.html*

[5] *Nguy Giang Linh*: **Návrh souboru pravidel pro analýzu anafor v českém jazyce**, MFF UK 2006.

[6] *Nguy Giang Linh; Žabokrtský, Z.*: **Rule-based approach to pronominal anaphora resolution applied on the Prague Dependency Treebank 2.0 data**, In Proceedings of DAARC 2007 (6th Discourse Anaphora and Anaphor Resolution Colloquium).

[7] *David W. Aha, Richard L. Bankert*: **Feature Selection for Case-based Classification of Cloud Types: An Empirical Comparision**, In Proceedings of the AAAI-94 Workshop on Case-Based Reasoning