NAMED ENTITY TYPE CLASSIFICATION: TWO MACHINE-LEARNING APPROACHES

Zorana Ratkovic
PFL054
January 13, 2008

PROJECT DESCRIPTION

Named entities (NEs) can be classified according to the type of entity that they represent (i.e. person, geographical place, company, etc.). Different NE classification systems exist. In this project, two different supervised machine-learning algorithms were implemented for the classification task: decision trees and a Naive Bayes Classifier. The performance of these two algorithms on the same data was compared. Similarly, the performance was measured against the type of information that was used for the learning.

TABLE OF CONTENTS

1 DATA DESCRIPTION

1.0 Introduction

For any type of Machine Learning task, one needs to have appropriate information that the algorithm can be trained, or learned from. For this task, morphologically processed data from the Czech National Corpus (CNC) was used[1]. CNC contains sentences in Czech and is rich in morphological information. For each word in CNC the form, the lemma and the morphological tag are provided. The form is what the word looks like in the sentence. The lemma represents the underlying lexical units. The morphological tag contains fifteen different values that can be selected, such as the part of speech (POS), the gender of the word, etc. Each word also contains a unique id. This id contains the number of the sentence that the word is in, as well as the index of the word.

The classification here was done using a supervised learning method. That is, a list of the correct named entities (NE) was provided. Characteristics extracted from these NEs would be used for the learning to occur. This way new NEs could be classified based on this learned information.

1.1 Data Size

For this project the data used was divided into three parts: a training set, a development set and an evaluation set. The first set is used for the algorithm to learn the parameters of the NEs (i.e. to train). The second set is used to test out different aspects of the data and the algorithm. Finally, the third set is used strictly for evaluation of the algorithms. Originally, the training set consisted of 6112 NEs, the development set of 826 NEs and the evaluation set of 806 NEs. Since this project deals with supervised learning, the NEs and their correct classification types were known. Along a file with a list of all the NEs and their correct type classification, a file was used that contains a list of the NEs and the sentence that they are

[1] http://ucnk.ff.cuni.cz/english/index.html

extracted from. These files also contain the morphological information on each word, as provided by the Prague Dependency Treebank.

Several different NE type classification hierarchies exist. In this project the system provided by Magda Ševčíková and her colleagues was used[2]. This hierarchy has both one level (i.e. p – personal names) and two-level classification (i.e. pf – first names and ps – surnames). In this project I decided to use one-level classification due to the fact that for the two-level classification there were too many categories with too few instances. Sufficient amount of data is needed for each class in order to insure that proper learning is taking place. I also decided to discard the NE types that were not part of the provided methodology file. This included the following types: *f*, *segm*, *text*, *cap, lower*, *upper* and *?*. After removing these NEs, the training set had 5732 instances, the development set had 783 instances and the evaluation set had 759 instances.

1.3 Extracted Features

There are many different types of features that can be extracted for the NEs. I decided to extract nineteen features. I divided these features into four categories based on the type of information they contain: morphological features, context features, typographic features and structural features. Morphological features are based on information that can be found inside the morphological tag of a word or the lemma. Context features look at the context that the NE is in. Typographical features look at the typography of the word. Structural features are those that are based only on information that can be extracted from the sentence and the location of the NE in the sentence.

Morphological features are important because they show deeper information about the word. It has also been shown that context (i.e. words that surround the word being examined) can be very helpful for learning. It is therefore useful to consider extracting morphological features not only for the NE itself, but for the words that surround it as well.

There are four morphological features that I extracted. While there are fifteen morphological values inside the tag, I extracted the part of speech (POS) at index one, the

[2] Magda Ševčíková, TR-2007-36 *Zpracování pojmenovaných entit v českých textech*

gender at index three and the case at index five. All these features are categorical. For the POS there are twelve possible values, for the gender ten and for the case eight[3]. The next morphological feature looks at the lemma of the word. In examining the lemmas it became clear that a lot of them contain the form *;X* (where *X* represents some letter). I extracted this categorical feature with five possible values: *G* if *;G* was found inside the lemma, *S* if *;S* was found, *Y* if *;Y* was found, *K* if *;K* was found and *N* otherwise. There were more *;X*s than the four specified, but their number of occurrences were low, so I decided not to use them.

The context a word is in is also very important. I extracted ten context features. I extracted the morphological tag for the previous word (w_{i-1}), the word before the previous (w_{i-2}) and the word following the NE (w_{i+1}). For each of these, I extracted the part of speech (POS) found at index one, the gender (index three) and the case (index five). Second I extracted whether there are any NEs before the current NE in the same sentence. For the look-back window I choose to look back four indexes. This is a categorical variable with two possible values (Y/N).

The typographical features look at the actual form of the word. In looking at the NEs, one of the first things that come across is that some NEs are strictly numerical, while other are made up of characters. Also, some have capitals while other are small letters. There are also similar patterns that can be found inside the lemma.

I extracted one form feature, which looked at the typography of the word. This is also a categorical feature. There are five possible values: *C* if the word begins with capitals, *A* if the word is made up of all capital letters, *N* if the word has numbers, *S* if the word is made up of all small letters and *O* otherwise.

The structural features are features that do not look at morphological information, but look at the word and the sentence that it occurs in. These types of features do not look at the morphological information of the context, but rather looks at the word and how the sentence structure relates to it.

I extracted four structural features. First, there is a feature that tells whether the NE is the first word in the sentence or not. This is a categorical variable with two possible values

[3] For more information about the morphological tag and the information it contains see: http://ufal.mff.cuni.cz/rest/CAC/doc/cac-guide/eng/html/chapter12.html

(Y/N). Next, I extracted the word length. This is a continuous variable with values ranging from one to twenty. Third, I extracted the length of the sentence. This is also a continuous variable with values ranging from one to one hundred and twenty. Finally, I extracted the index of the word in the sentence. Once again, this is a continuous variable with values ranging from one to one hundred.

1.4 Data Issues

One of the issues that was encountered is that in all the three files there were instances for which one NE had multiple (i.e. two) classifications. In the training file I found 344 such cases, 58 in the development file and 48 in the evaluation file. This poses two problems. First, this hinders the learning because you have the same extracted features with different results. Second, our evaluation assigns only one NE type for each instance. In these multi-type cases at least one of the two classifications is bound to be wrong, thereby increasing the error rate. In the training file I found 344 such cases, 58 in the development file and 48 in the evaluation file. This represents about 6-7.5% of the entire data.

There are two ways to deal with this problem: one is to add an extra feature which states whether this NE has a single or multiple classifications. The second is to remove the second classification for these multi-type instances. I decided to use the second method. The reason is that since the algorithm can only produce one type for an NE (even one with multi-type classification), it makes sense to only look at single classification NEs. Removing the multi-NEs resulted in the training set having 5388 instances, the development set 725 and the evaluation set 701.

1.5 Algorithms

In deciding which machine-learning algorithm to use, there were several issues that needed to be considered. First, it is possible that the training data might contain some errors. Therefore, any algorithm used would have to be robust to noisy data. Second, it is likely that not all possible values for each attribute would be encountered in the training data. Also, the algorithm would have to output a discrete value from a given set (i.e. the given set of possible

NE types). Further, the features include both discrete and continuous values. Therefore, the algorithms used would have to be able to handle both such attributes. Finally, the algorithms would have to be able to handle a disjunction of expressions, where each expression consisting of a conjunction of attribute values. Therefore, I chose to use the decision tree and Naive Bayes algorithms. Both these algorithms fit the above criteria, and both are good machine-learning algorithms to use for classification of items based on a conjunction of features.

2 THEORETICAL ASPECTS OF METHODS (MODELS)

2.1 Decision Trees

For the first machine-learning algorithm, I chose to use decision trees. This is an inductive inference algorithm, where the learned function is represented as a decision tree. That is, in the tree each node represents an attribute with each edge that leads from the node representing the value of the attribute. A classification is therefore obtained by traversing the tree from the root to a leaf. Each leaf is a classification. Therefore, its output is a discrete value. This algorithm is robust to noisy data. Further, the decision tree does not have to contain all possible attribute values. This algorithm however requires all the attribute values be categorical. It is easy to turn continuous values into categorical ones by creating intervals. Therefore, this algorithm is a good candidate for the task at hand.

In the decision tree algorithm first the root is created. From there each node represents one attribute with the edges leading from this node representing the possible values for this attribute. The most important part of this algorithm is choosing which attribute is placed where in the tree. For this, the information gain measure is used.
Information Gain is defined as follows:

$Gain(D,A) = Entropy(D) - \Sigma_{v \varepsilon Values(A)}(|D_v|/|D|)Entropy(D_v)$
where
D = training data
D_v = {d ε D; A(d) = v}
A = attribute
$A(d)$ = attribute value A in instance d
$Values(A)$ = set of all possible values of attribute A

and

$$\text{Entropy}(D) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

where
p_i is the proportion of the training data that belongs to class i
c is the number of discrete values that the classification can produce

This information gain is used to decide which attribute is assigned at which node. This gain represents how effective this attribute is in classifying the training data[4].

2.2 Naive Bayes Classifier

For the second algorithm I choose to use the Naive Bayes Classifier. This classifier can be used when each instance can be described as a conjunction of attribute values and where the output is discrete valued. For our task, each training example is defined as a conjunction of features. Also, the output is discrete valued. Further, the algorithm is robust to noisy data, and can handle if not all the values of an attribute are present in the training data (i.e. by using smoothing).

This classifier is based on Bayesian learning where probabilities of different hypothesis are estimated based on counts of their occurrences in the data. The classification is therefore made by choosing the most probable hypothesis, given the training data. The Naive Bayes classifier is different from the Bayes classifier in that it assumes that the attribute values are independently conditioned for any given target value.
That is, the classification is produced by the following equation:

$v_{NBC} = \text{argmax}\ v_i \in V\ P(vi) \prod_i P(a_i | v_i)$

where
v_i = target value
V = set of all possible target values
a_i = attribute value
The above probabilities are estimated by frequency counts obtained from the training data.[5]

Therefore, the Naive Bayes classifier does not search through a hypothesis space, but rather calculates the most probable classification, based on the counted frequencies from the training data.

[4]Mitchell, Tom M. (1997) *Machine Learning.* WCB McGraw-Hill: New York, New York; pp. 52-78.
[5] Mitchell, Tom M. (1997) *Machine Learning.* WCB McGraw-Hill: New York, New York; pp. 154-199.

3 PRACTICAL ASPECTS OF METHODS (MODELS)

For the implementation of the NE type classification I used the R environment. I used the R environment because it provides a remarkable number of already implemented machine learning algorithms. It also has a class that can be used to handle XML data.

The first thing I did is extract the nineteen features from the training file. I used the unique id specified for each NE in the *train.ne.oneword.xml* file to locate the word inside the *train.m.xml* file and extract the specified features. I created a data frame with all this extracted information. In the data frame each row contains the values of the nineteen extracted features for this particular NE and its correct type.

For the decision tree algorithm I used the *rpart* library in R. I used the *rpart* method to grow the decision tree. For the Naive Bayes classification I used the *e1071* class, and the *naiveBayes* method to estimate the probabilities.

I used the development files (*dtest.ne.oneword.xml* and *dtest.m.xml*) to create a data frame with all the extracted features (like for the training files), except that the correct classification was withheld. I used the predict method for both the algorithms to see what classification was produced for each of the two algorithms. I used the development files to see how changes in the features used would affect the result.

Finally, I extracted the features from the evaluation testing files (*etest.ne.oneword.xml* and *etest.m.xml*) and once again used the *predict* method. I computed the accuracy of the two algorithms by comparing the produced classification to the true classification provided in the *etest.ne.oneword.xml* file.

4 PROJECT ANALYSIS

In order to properly evaluate the performance of the NE classification, a baseline has to be created. I created a baseline by classifying all the NE instances by the most common type (*p*). This resulted in a 45.44% baseline for the development data and 51.14% for the evaluation data.

4.1 Development Testing

I used the development data to test and further experiment with several features. First, I experimented with the three continuous features. That is, I tested the results with the features being left as being continuous and with the features being divided into discrete intervals. However, this produced no change for either algorithm. Also, playing around with different intervals produced no change as well. Next, I played around with the look-back interval for any previous NEs. Originally I used a look-back interval of four words. Increasing this interval to six improved the accuracy by 2%. Further increasing it to eight produced no change, while increasing it to ten brought the accuracy down by 0.5%. I also tried including the number for the NE, the next word, the previous word or the word before the previous. However, this produced no improvement in the accuracy.

4.2 Parameter Testing

All algorithms have different parameters that whose changes can influence the results. For decision trees one useful feature is pruning. That is, pruning a tree often leads to improvements in the accuracy. I used the *prune* function and changed around the complexity parameter. However, this did not improve the results. That is, the results stayed the same, or even started to decrease when the complexity parameter became too big. (See Table 4.1 for the pruning results.)

Complexity parameter	Accuracy
Cp=0.2	0.704
Cp=0.1	0.776
Cp=0.05	0.787
Cp=0.01	0.834
Cp=0.005	0.834

Table 4.1 changes in cp and the resulting accuracy for the *etest* files.

For the Naive Bayes classifier, one of the parameters that can be changed is the threshold that will be used when a probability being estimated is 0. Playing around with this feature, once again, did not increase the accuracy. That is, for certain values it remained the same as when the threshold was not changed, and for some it decreased. (See Table 4.2 for details.)

Threshold	Accuracy
0.0001	0.796
0.001	0.836
0.05	0.836
0.1	0.82

Table 4.2 shows how the changes in threshold affect the accuracy on the *etest* files.

4.3 Evaluation Testing

Next, I tested the accuracy for both the development and evaluation data under several conditions. I first tested both the algorithms with all the nineteen extracted features. Next, I tested the two algorithms with all the features except those extracted from the morphological tags (for the current word, the next, the previous and the one before the previous) since this is the most numerous feature category. For this project, the training data contained morphological information about the NEs. This requires the data to be pre-processed. This type of data is expensive. I therefore decided to test the two algorithms only with features which can be extracted from the sentence and the word itself (this includes the structural features, the presence of NEs before the current one and the typography of the word). Finally, in this instance we are dealing with text and therefore have access to information such as the typography. However, if we were dealing with speech instead of text this information would

not be available. Therefore, I decided to test the two algorithms with just the structural features.

4.4 Results

Type of test	Decision Trees – etest	Naive Bayes – etest
Baseline	0.834	0.836
Structural Features only	0.820	0.813
Structural + Topographical	0.643	0.640
No Tag Information	0.526	0.511
All features	0.501	0.501

Table 4.3 The results.

Table 4.3 shows the results of the testing while the figure below show a comparison of the results for the two algorithms for the different instances that were being tested, for the evaluation data (see alsoFigure 4.4). (For the results based on the development data, see Appendix A).

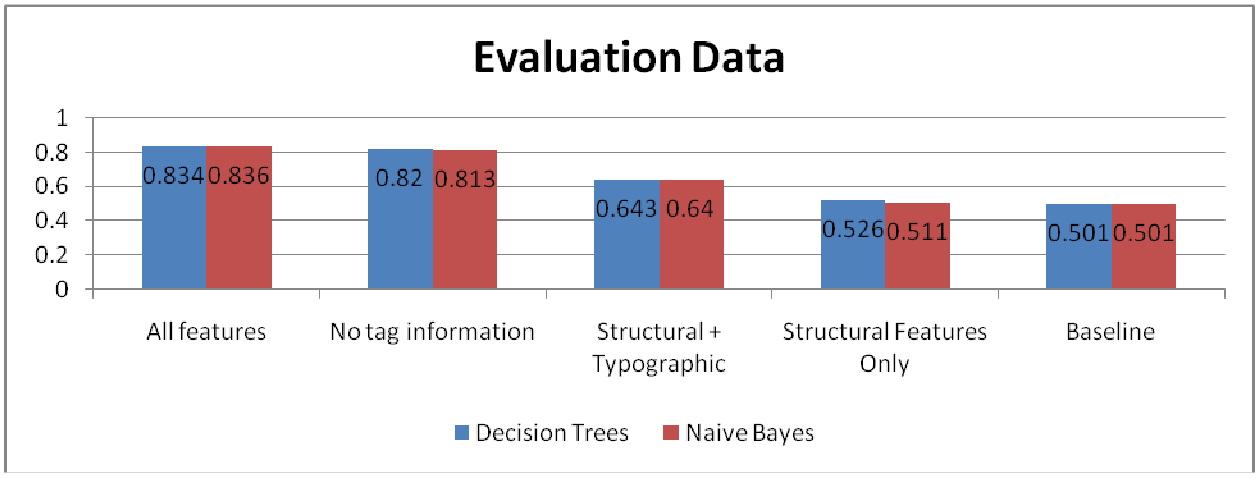

Figure 4.4 The results based on the *etest* files for both the algorithms.

There are several things to be noticed. First, the best overall result is obtained by the Naive Bayes classification for the evaluation files with all nineteen features extracted. However, the performance of the two algorithms is quite similar, and varies by 2-3% at the most.

Regarding the features, removing the tag information (morphological features) decreases the result, but only so slightly. Removing the lemma as well the morphological information decreases the accuracy quite a bit (20%). However, the results are still approximately 15% higher than the baseline. Finally, removing the topographical information (as well as the lemma and morphological features) produces results that are no better than the baseline. What these results suggest is that the more information we have, the better the results. It also shows how different types of features all seem to play an important role in the classification. The data here is not very large, and one must not generalize from one set of results to all other instances. However, what these results do show is that playing around with different types of features can produce different results and is an aspect that should further be examined.

4.5 Error Analysis

Sometimes results can be hampered by one (or a couple) class which is being misclassified and bringing down the accuracy rate. That is, it could be the case that all (or almost all) the errors are being caused by one class being misclassified. In analyzing the NEs that were misclassified, it becomes clear that this is not the case for this project. That is, the misclassification occurs for several different classes, for both the methods. A confusion matrix for the Decision Trees Algorithm (Table 4.5) and the Naive Bayes Classifier (Table 4.6) can help us analyze the errors.

TRUE

PREDICTED	a	g	i	m	o	p	s	T
a	0	0	0	0	0	0	0	0
g	0	113	20	0	1	12	0	0
i	0	2	25	11	9	1	13	0
m	0	0	0	0	0	0	0	0
o	0	0	0	0	17	0	0	0
p	0	14	8	2	14	338	0	4
s	0	0	0	0	0	0	0	0
t	0	0	0	0	5	0	0	91

Table 4.5 shows the confusion matrix for the Decision Tree Algorithm.

From the above table, we can see that 'i' is very often wrongly predicted. Also, the classification does not classify 's' (even though there are thirteen instances of 's', which are misclassified as 'i').

TRUE

PREDICTED	a	g	i	m	o	p	s	t
a	0	0	0	0	0	0	0	0
g	0	115	22	1	3	12	0	3
i	0	9	22	3	7	7	2	0
m	0	0	3	3	0	2	0	0
o	0	1	1	1	27	7	1	4
p	0	4	4	0	5	323	0	3
s	0	0	1	5	3	0	10	0
t	0	0	0	0	1	0	0	85

Table 4.6 shows the confusion matrix for the Naive Bayes Classifier.

From the above table we can see that for the Naive Bayes Classifier 'i' is very often misclassified, as is 'm' and 'o'.

For both the methods there are misclassifications for 'g' and 'p', but proportionally to the number of correct classifications, the errors are small.

It is interesting, as well, to see the differences in errors. The Naive Bayes Classifier makes more different types of errors. The Decision Tree Algorithm makes fewer types of errors, but for each type, the errors are more numerous.

4.6 Significance of Results

There is a difference in the best performance of the two algorithms. However, are these results statistically significant?

To test the significance, we test against the null hypothesis (Ho).

Null Hypothesis:

Ho: $d = p1 - p2 = 0$ where p1 = success rate of Decision Trees = 0.836

p2 = success rate of Naive Bayes = 0.834

d = difference in performance rates

Alternative hypothesis:

H1: $d' = 0.836 - 0.834$

$= 0.002$

To test the hypothesis, we create a 95% two-sided confidence interval, using the following formula:

$d' +- z_n \sigma$

where $\sigma = \sqrt{(error(p1)(1-error(p1)))/n1 + (error(p2)(1-error(p2)))/n2}$

where n1 = n2 = 701 (size of evaluation data)

error(p1) = 0.164

error(p2) = 0.166

$\Rightarrow \quad \sigma = \sqrt{[(0.164*0.836)/701] + [(0.166*0.834)/701]}$

$= 0.01983$

For a two-sided confidence interval $z_n = 1.96$

$d' +- (1.96)(0.01983) \Rightarrow 0.002 +- 0.0389$

CI: [-0.0369, 0.0409]

Since the confidence interval contains zero (0), we cannot reject the null hypothesis. Therefore, at a 95% level, the difference between the two algorithms is not statistically significant.

4.7 Graphical Representation of Classification

A decision tree is not only useful in classifying future NEs, but it also offers a graphical representation of our classification. We can examine the tree to see how the classification occurs, and also to see which features are proven to be most useful for our classification.

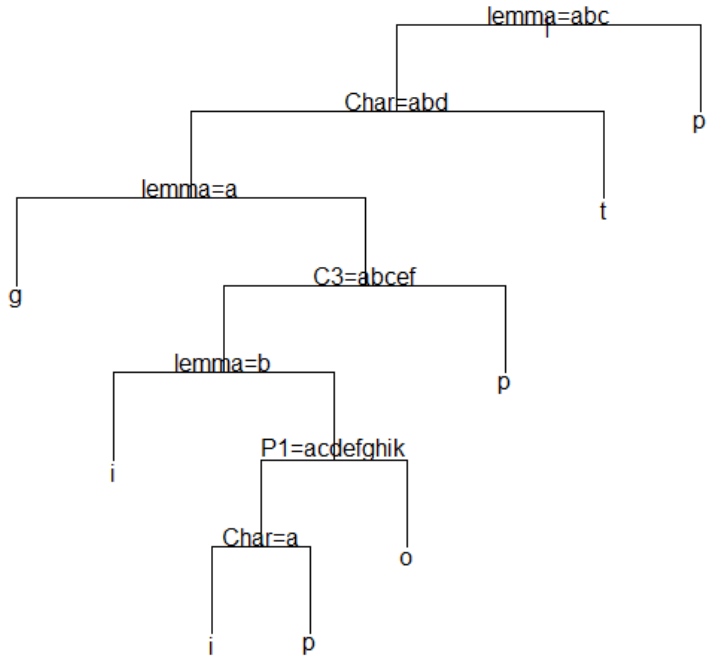

Figure 4.7 shows the decision tree that is generated for the nineteen extracted features.

Pruning a tree, demonstrates a way of getting a much simpler tree (which requires less decisions and classification features), by keeping the accuracy close to that of the non-pruned tree.

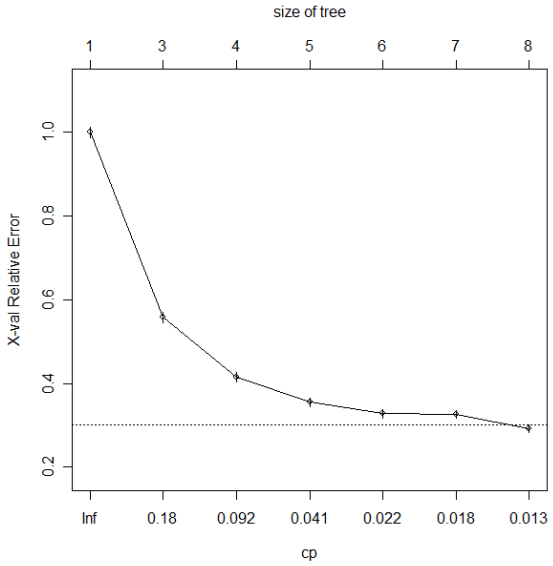

Figure 4.8 shows the *plotcp* plot of the Decision Tree.

We use the above graph to select a cp=0.20 for the pruning.

Figure 4.9 shows the resulting pruned tree:

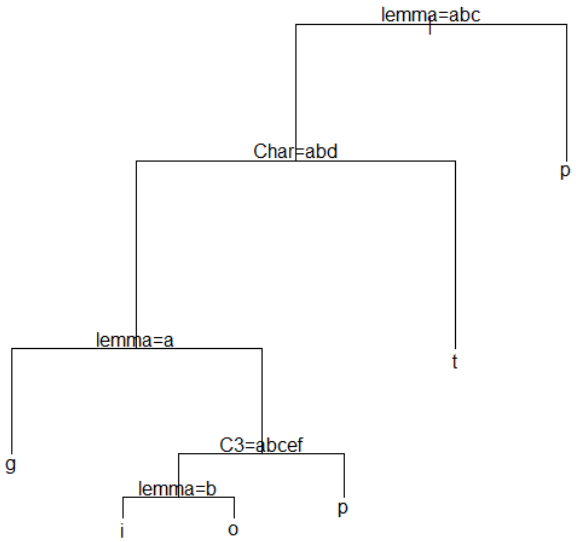

Figure 4.9 The pruned tree.

The difference between the pruned and the non-pruned tree produce the same leafs, yet the non-pruned tree has more branches (eight, compared with six for the pruned). One of the features that is in the non-pruned tree but is not in the pruned tree is the POS of the previous word (labelled 'P1'). What this suggests is that for this particular data, the context feature did not bring forth as much information for learning as the other two morphological and the one topographic features did.

4.8 Conclusions

Overall, the accuracy rates show that the methods chosen were successful for the classification task. Further, the similar results for the two different methods show that the two algorithms are mutually competitive. Also, the results are much higher than the baseline, further showing the success of the classification. The results show the benefits of using supervised methods in machine learning, and the benefit of having pre-processed training data.

What we can tell from the trees (Figure 4.8 and Figure 4.9) is the features that are most useful for the classification. We can tell that the typography of the word (labelled 'Char'), the lemma and the gender of the current word (labelled 'C3') are the most useful in classifying the data. These are two morphological and one typographic features. This shows that for this data with the Decision Tree algorithm, the morphological and typographic features were most useful for the classification task.

4.9 Future Directions

One thing that could be tested in the future is two-level classification. The two-level classification is more fine-grained, and I believe that with sufficient training data it would outperform the one-level. Other future directions include looking at getting similar results that are not so heavily based on pre-processed data. This would not only be less expensive, but would also allow the classification task to be performed on more languages (ones that do not have corpuses or Treebanks available).

5 CONCLUSION

This project has shown that both the Naive Bayes Classifier and Decision Trees can be successfully used for the Named Entity Type Classification. Moreover, it has shown that the two methods are competitive machine-learning algorithms. Further, while there are many different types of features that can be extracted, the best results are those based on heavily pre-processed data. This shows how pre-processing data in such a way can later be beneficial for many other applications. Finally, the results of this project show how machine-learning algorithms can be quite successful for natural language processing tasks.

6 APPENDIX A

Type of test	Decision Trees-dtest	Naive Bayes – dtest
Baseline	0.837	0.800
Structural Features only	0.820	0.797
Structural + Topographical	0.579	0.583
No Tag Information	0.480	0.483
All features	0.454	0.454

Table A.1 The results of the tests run on the development data.

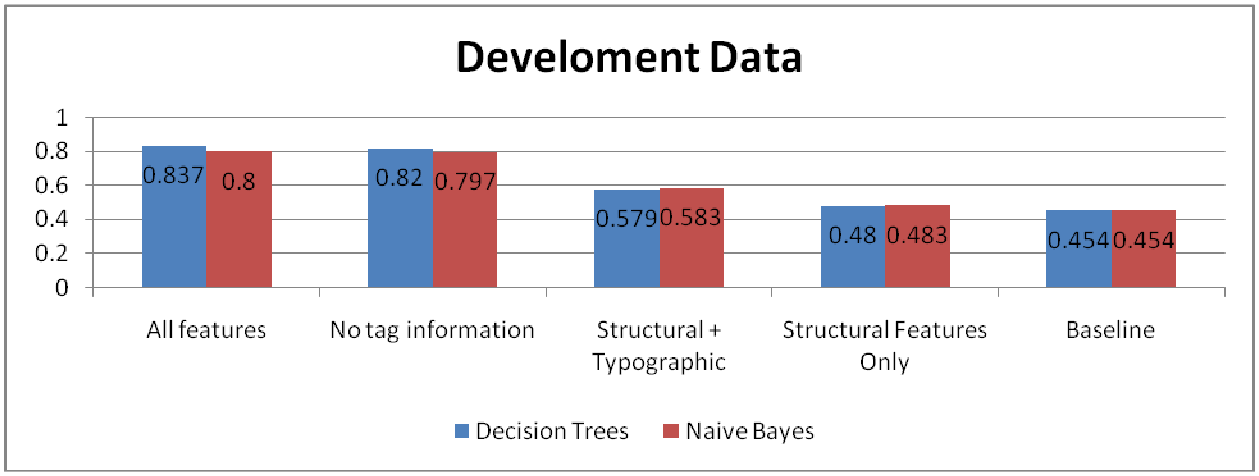

Figure A.2 The comparison of results for the two methods on the development data.

7 REFERENCES

Mitchell, Tom M. (1997). *Machine Learning.* WCB McGraw-Hill: New York, New York.

Ševčíková, Magda. TR-2007-36 *Zpracování pojmenovaných entit v českých textech*

http://ufal.mff.cuni.cz/rest/CAC/doc/cac-guide/eng/html/chapter12.html

http://ucnk.ff.cuni.cz/english/index.html