Automatic word lemmatization

Dunja Mladenić

J.Stefan Institute, Ljubljana, Slovenia and
Carnegie Mellon University, Pittsburgh, USA
J.Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia
Dunja.Mladenic@{ijs.si,cs.cmu.edu}

Abstract

This paper is addressing a problem of automatic word lemmatization using machine learning techniques. We illustrate a way that sequential modeling can be used to improve the classification results, in particular to enable modeling sub-problems mostly having less than 10 class values, instead of addressing all 156 class values in one problem. We independently induced two models for automatic lemmatization of words using the complementary data representation modeled by using (1) a set of if-then classification rules and (2) the naive Bayes classifier. The model induction was based on a set of hand labeled words of the form (word, lemma) and some unlabeled data. We used data for Slovenian language as an example problem, but the approach can be applied to any natural language provided the data is available. Actually, the labeled data we have used is a part of a larger dataset containing the same kind of information for different European languages. The data representation was based on two independent feature sets describing the same examples. The first feature set is using the letters in the words to give a structured representation of words on which a "classical" learning algorithm is applied. In our case we used classification rules algorithm, that was shown to work well on different machine learning problems. The second feature set is using the unlabeled data to get context of the words representing each example with a set of short documents - context. Here we applied the Naive Bayesian classifier directly on text data, as an approach shown to work well in document classification. The experimental results show that both approaches perform better than a simple, majority classifier.

1. Introduction

In text classification the goal of assigning class labels to documents is often achieved by constructing features based on some transformation of words, such as stemming, phrase construction or expanding known abbreviations. For stemming of English words, there are known algorithms giving approximation of the mapping from word to its stem, eg. Porter's stemmer (Porter, 1980). These approximations have been reported to work well for the purpose of most text classification problems involving stemming. However, stemming words in other natural languages requires separate consideration. In particular, there is no algorithm available for stemming Slovenian words and the same is true for many other languages. The related research on learning English past tense addresses a subproblem of a general word normalization problem addressed here. The additional difficulty of our problems is that we are dealing with highly inflected natural language, having up to 30 different word forms for the same normalized word. Some researchers use decision trees or neural networks to predict separate letters of the transformed word based on the letters in the past tens form of the word (Ling, 1994). Some work using relational learning wit decision lists was also proposed for learning English past tense (Mooney and Califf, 1995) and learning stemming (Dzeroski and Erjavec, 2000). Unsupervised learning capturing different morphological transformations (Yarowsky, 1995) among others uses some hand constructed rules for defining the context of a word and additionally avoids labeling of the data.

2. Data description

The problem addressed in this paper is learning a mapping of words based on some labeled data defining the mapping and a set of unlabeled documents in the same natural language. The mapping is from many to one, where we're mapping from up to 30 different forms of the same word to the normalized form of the word. We illustrate the approach on the problem of stemming words and define a mapping in an indirect way as a transformation on a word suffix. This transformation is from different forms of the same word to the normalized form of the word (lemma). In this problem presentation, the words with the same class value are the words having the same suffix transformation and not the words that have the same normalized form. For instance, *working, laughing, eating* would be all assigned the class value mapping from *ing* to an empty string. Notice that in the original problem *working* is in the same class with *works, worked, work*, as the words with the normalized form *work*. However, the final result of our transfromation is the same. Using the both problem formulations as two complementa view to the same problem, and combining their results is potential part of future work.

The goal here is to be able to apply the learned mapping on new text, and output the transformed text. One of the direct applications of such system is in using it as pre-processor of text for text classification or in search engines. For example, stemming of words is especially valuable for text classification of document written in highly inflected natural language such as Slovenian, Check or Croatian, where an average word has about 20 different forms.

The same labeled dataset was already used is some experiments of learning stemming (Dzeroski and Erjavec, 2000). The whole dataset contains about 500 000 different words. For each word its normalized form is provided and additionally, the word is annotated by some information about the word form such as singular/plural, noun/verb/adjective, etc. Since our goal is to develop approach that can be used on any texts in the selected natu-

ral language and this additional information is not readily available in the usual texts, we are not using this additional information with learning the mapping.

We generated two complementary classifiation models, one using classification rules on a "classical" feature representation of examples and the other using Naive Bayesian classifier on text representation (see Section 3.). The text representaiton of a word is based on the context in which the word usually occurs. The context was obtained from a set of 28 novels (4Mb of text) known from Slovenian literature. Since our set of words contains all different inflections of the included lemmas, only about 15% of these words were found in the used set of documents.

3. Approach Description

Learning a specific transformation of a word that is needed to obtain its normalized form first requires representing the word in a form suitable for data mining. We identified two sets of independent features: (1) letters in a word itself and (2) context of a word from some set of documents. In this paper we describe experiments using each of the feature sets independently. Combining them using Co-training (Blum and Mitchell, 1998) is part of the future work.

(1) We define the letter-based representation of word as a set of features giving word suffix of a fixed length. In our experiments, we used up to five last letters as a suffix. In that way, each word is represented with five discrete features.

(2) We defined a context of a word to contain a word and up to six words in a window around the word (position ∓ 3). Each word is thus represented with a set of words collected from all the contexts that the word appears in. As already mentioed, only about 15% of the labeled words were found in the set of unlabeled documents. For the remaining words, we could not generate representation based on the word context. Thus the experiments using the context features were performed only on these 15% of the words. Testing the approach using much larger set of unlabeled documents is part of the future work. Namely, we plan to use a subset of Slovenian corpus containing abut 40 000 Web documents obtained in experiments for "automatic Web search query generation to create minority language corpora" (Ghani et al., 2001).

The class value is defined from word and the associated lemma by giving the transformation to be applied on the word in order to get the lemma. The transformation is given in the form of map X to Y (XToY), where X is a suffix of the word and Y is a suffix of the lemma Since out class value is strongly connected to the last letter of a word, we applied a simple way of *Sequential modeling* (Even-Zohar and Roth, 2001). Namely, instead of taking all the training examples with 156 different class values, we generated a one level decision tree that divided examples into subsets depending on the last letter in the word represented by the example. In this way we decomposed the problem into 26 independent problems, each having 1-40 class values (most of the subproblems have less than 10 class values). Because of the nature of the problem, the words that have different last letter can share the class value only if that value is "do not change the word" (in our notation value _To_) or "add some suffix" (_ToY). This means that with the applied sequential modeling the problem was divided into subproblems with almost non-overlapping class values. The decomposition of a problem according to the last word letter reduced the number of class values inside each subproblem and improved the classification results, as shown in Section 6.. As shown in Table 6., the classification performance on the letter-based word representation is improved as well as on the context-based word representation for 5% and 28% respectively.

4. Classification rules on the word suffix

We defined the problem of mapping different word forms to its normalized form as a data mining problem and applied classification rules for learning the mapping. The algorithm we used in our experiments was developed in ATRIS rule induction shell (Mladenic, 1993). The algorithm is based on local optimization and adopts the covering paradigm as follows:

```
Hypothesis := ∅;
for each class do
   PExs := positive training examples;
   NExs := negative training examples;
   repeat
      Rule := new rule induced from PExs and NExs;
    {where induction is performed by a combinatorial
    optimization algorithm}
      Hypothesis := Hypothesis ⋃ Rule;
      From PExs delete examples covered by Rule;
   until PExs = ∅ (or some other stopping criterion is
    satisfied);
end for
output Hypothesis;
```

The covering algorithm loop is controlled by a *stopping criterion*, used to decide when to stop the induction of rules for the current class. This criterion can be referred to as the *sufficiency criterion*. Combinatorial optimization algorithms contain a *stopping criterion*, usually referred to as the *necessity criterion*, if it is a part of the heuristic function that is used to guide the search. In domains with exact and non-noisy data, the sufficiency criterion requires completeness (coverage of all positive examples) and the necessity criterion requires consistency (no covered negative example). In domains with imperfect data, the two criteria are implemented as heuristics for evaluating the expected classification accuracy, aimed at avoiding overly specific hypotheses. In our experiments the sufficiency criterion is a combination of three parameters: coverage of all but `MxUncoveredExs` number of positive examples, the number of unsuccessful runs of optimization algorithm (each starting with the different initial rule set randomly) and the maximum number of rules per class. The necessity criterion used to stop the rule refinement is based on finding any changes of a rule that would improve its quality. We maximize *rule quality*, that is here defined as the rule accuracy estimated using m-estimate (Cestnik and Bratko, 1991) ($m = 10$) of probability that the example covered by the rule has the same class value as predicted by the rule. In case of the same accuracy rules, the rule simplicity is considered, where the simpler rules are preferred. We

define *rule simplicity* as the number of literals in the rule either including a new feature or adding a new value to the already included feature.

A single rule is a conjunction of selectors, where a *selector* relates a feature to a value or a disjunction (so-called *internal disjunction*) of values. A conjunction of selectors forms a *complex*:

$$R_i = \bigwedge_j \left[F_j = \bigvee_k V_{jk} \right]$$

Where R_i stands for the i-th *complex*, F_j for the j-th feature and V_{jk} for the k-th value of the j-th feature. A hypothesis is a set of *if-then* rules of the form **if** *Complex* **then** *Class*. An example rule is the following: **if** [*ThreeLastCh= HOM* $\vee$ *NOM* $\vee$ *DOM* $\vee$ *SOM* $\vee$ *POM* $\vee$ *BOM* $\vee$ *FOM* $\vee$ *XOM*] **then** *Trnasformation = OMTo_* (Quality = 0.8280).

Deterministic search algorithms named *k-Opt* originally proposed in combinatorial optimization for traveling salesman problems (Syslo et al., 1983), can be modified for the purpose of a single rule optimization as listed below. The *transformation* used in these algorithms for 'moving' in the candidate solution space is the modification (negation) of *1, 2, . . . , k*, randomly chosen bits in a binary vector representing a rule. This internal rule representation enables efficient evaluation of the rule, where each attribute value is represented with a bit (set to 1 if the value occurs in the *complex*).

$current\_rule$:= random rule; {random binary vector}
repeat
  Find one (or two) values in the rule binary vector, modification of which maximizes the quality of the rule (i.e., no other modification of one (or two) values provides a rule with better quality);
  if $modified\_rule$ is of better quality than $current\_rule$ **then**
    $current\_rule$:= $modified\_rule$;
  end if
until no modification to better rule quality is possible;
output $current\_rule$;

The above algorithm is called *1-Opt* if (in the first step of **repeat-until** loop) it checks only rules with one modified value while *2-Opt* checks rules with one or two values modified in a single search step. In general *k-Opt* can be performed by checking rules with up to *k* values modified. We used the greedy variant of *2-Opt* algorithm that at each step searches for the first modification that provides a rule with better quality. Figure 1 shows the optimization progress for three out of the fourteen rules generated for the first class value.

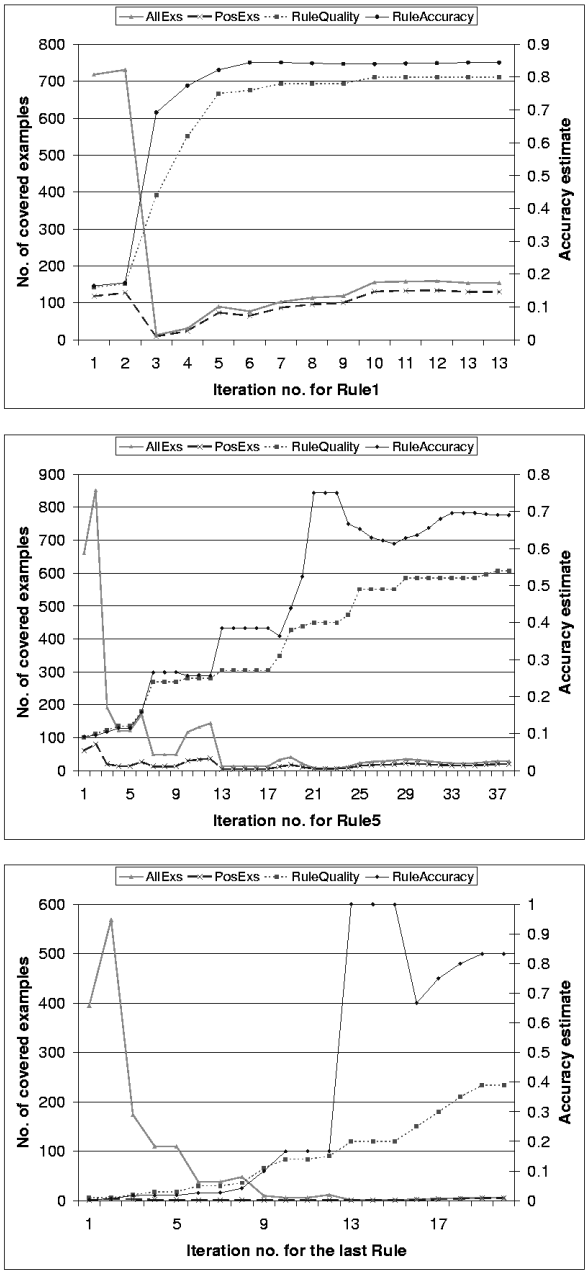

Figure 1: Rule refinement through the iterations of the optimization algorithm in one of the 5-fold cross-validation splits for the problem with 156 class values. As illustration, we show three out of the 14 rules generated in order to cover examples with the first class value. AllExs and PosExs show (on the left y-aix) the numebr of examples covered by the rule, all examples and positive examples respectivey. RuleQuality is m-estimate of the rule accuracy and RuleAccuracy is the rule accuracy estimated simply by dividing number of positive examples covered by the number of all covered examples. It can be seen that the first rules cover larger number of positive examples, while the later rules are left with the problem of covering the remaining, usually small number of uncovered positive examples.

5. Naive Bayes on the word context

Nave Bayesian classifier have been shown to work well on a number of text classification problems despite the used word independence assumption. Our experiments are performed using the Naive Bayesian classifier base on the multinomial event model as outlined in (McCallum and Nigam, 1998). Notice that the product goes over all words that occur in the representation of the testing document Doc.

$$P(C|Doc) = \frac{P(C)\Pi_{W_j \in Doc} P(W_j|C)^{TF(W_j,Doc)}}{\sum_i P(C_i)\Pi_{W_l \in Doc} P(W_l|C_i)^{TF(W_l,Doc)}}$$

Where $P(W_j|C)$ is the conditional probability here estimated using the Laplace probability estimate, $TF(W_j, Doc)$ is the frequency of word W_j in document Doc. The calculation is illustrated on classification of word *ADAMOM* in Figure 5..

Context-based word representation can be very short, containing just a couple of words, but this depends on the goodness of the match between the labeled and unlabeled data. Namely, this will not be the case if the unlabeled data contains several occurrences of the labeled words. Nevertheless, using the Naive Bayesian classifier on relatively short documents is problematic, since there is not enough evidence for classifying testing examples. In order to preserve information about inflection, we use all the words without changing them (no stop-list or stemming in applied). In many cases, very few or none of the words from a testing example is found in the training examples, and when the word is found in the training examples it may be distributed over examples having different class values. The relatively low number of different words for each class value contributes to unreliable probability estimates used in the classification.

Figure 5. shows how the probability estimate for the correct example class (cut the ending "om") is calculated using the Naive Bayesian classifier. This class value has the prior probability 0.3065, the calculated posterior probability estimate is 0.94598 and it is based on the 3 words from the testing document with $TF(W_i, C_5) > 0$. The three words are very common words, namely `"je, z, in"` that are Slovenian for `"is, with, and"`. Classes that do not share any words ($\forall_{W_i \in Doc} TF(W_i, C) = 0$) with the testing document are ignored in the final classification.

6. Experimental Results

Experiments are performed on two sets of independent features: (1) letters in a word and (2) context of a word (see Section 3.). We report results of classification accuracy from 5-fold cross-validation, where exactly the same examples, but represented in a different way, were used for generating and testing the model. We used a subset of the whole labeled data set containing 3970 examples that were split randomly in 5 folders, each having 294 examples. The whole training-testing loop was repeated five times, with each of the five subsets being excluded from training and used as testing exactly once (cross-validation). We repeated the experiments on a subset of a double size having about 8000 examples, and the variation in the performance was small.

In addition to comparing the performance of the two independent representations, we also tested the hypothesis that using sequential modeling in a very simple way improves the performance. We generated a one-level decision tree that divided examples into subsets based on the last letter in the word. In this way, we decomposed the problem into 26 independent problems. Since many testing examples are short, it may happen that the Naive Bayes classifier has no information except the prior probability of the class values. Thus, we are reporting average accuracy only on the examples for which at least one word from our vocabulary of 11086 words had $TF(W, C) > 0$ for at least one of the class values. In this way about 15% of all testing examples were classified and the reported accuracy is on that 15% of examples. This evaluation is consistent with the classifiers needed for co-training approach, where we actually consider for labelling only the unlabeled examples for which the prediction is the most certain. The results of experiments are given in Table 6..

6.1. Discussion

A simple sequential modeling we used is actually based on one of the features from the letter-based example representation, namely feature *LastCh*. This is also why we can observe a small improvement of the performance in the letter-based representation experiments, since the same information was also made available to the classification rules generation algorithm as one of the five features. However, the information about the last letter of a word was a new and very useful for the complementary, context-based representation. We can see that from the experimental results, where sequential modeling almost doubled the classification accuracy of the context-based representation performance. It should be pointed out that the default classifier assigning the majority class value had the largest improvement in the performance due to sequential modeling, from 12% to 40.8% average accuracy.

Our definition of class value as a transformation of the word suffix resulted in some class values having very small number of example, that is additionally reduced by employing cross-validation. In our experiments using the letter-based representation, that small classes were ignored as noise meaning that no rules were generated for them. We tested omitting this pruning of rules and observed a very small variation in the average classification accuracy (it decreased from 74.2% $\pm$ 4.9 to 73.2% $\pm$ 1.4). Notice the change in the standard deviation from 4.9 when the small classes where ignored, to 1.4 when for all class values the set of rules was generated, even for only one positive example.

Our experiment using context-based representation were performed on about 15% of the labeled words that were present in the set of unlabeled documents and we were able to generate a context-based representation for them. We also tested ignoring that and testing on all the words, meaning that 85% of the words were classified based only on the word itself. The classification accuracy achieved using the sequential modeling approach decreased from 50.2 $\pm$ 2.1% to 21.8 $\pm$ 2.1%

Context-based representation of words had several rea-

```
(a)
CatWgt[5]=(6.000+1)/(467.000+11086), log CatWgt[5]=-7.40879
CatWgt[5]=(0.000+1)/(467.000+11086), log CatWgt.Key[5]=-16.76349
CatWgt[5]=(47.000+1)/(467.000+11086), log CatWgt.Key[5]=-22.24699
CatWgt[5]=(0.000+1)/(467.000+11086), log CatWgt.Key[5]=-31.60169
CatWgt[5]=(18.000+1)/(467.000+11086), log CatWgt.Key[5]=-38.01195
CatWgt[5]=(0.000+1)/(467.000+11086), log CatWgt.Key[5]=-47.36665

(b)
Document vector: JE BILO Z ADAMOM IN EVO
correct     --> predicted
[5:OMTo_]   --> [5:OMTo_]:0.94598
    Model evidence:TF(JE)=6.0 TF(Z)=47.0 TF(IN)=18.0
```

Figure 2: (a)Illustration of the Laplace probability estimate calculation in the loop of Naive Bayes classification, looping over all words from the testing document $P(W_i|C) = (TF(W_i, C) + 1)/(sum_j P(W_j, C) + \#Words)$. Here we are showing the value of each part separately and give the natural logarithm of the calculated value. We used logarithm in the calculation, because the product of probabilities in the loop is usually some very small number that may cause numerical errors in the computation. (b) Class value `5:[OMTo_]` shares three words with the testing document. All three words are functional words that can also occur in some other ordering or other context of a word having different class value. This shows that even tough this particular classification is correct it is not very robust and reliable.

Classification accuracy [%]	**Original problem**	**Sequential modeling**
(1) Letter-based representation using classification rules	69.4 ± 2.1	**74.2** ± 4.9
(2) Context-based representation using Naive Bayes	22.4 ± 2.8	**50.2** ± 2.1
Majority classifier	12.0 ± 2.1	40.8 ± 2.1

Table 1: Experimental comparison of letter-based and context-based word representation used for the problem of word classification with the goal to transform the word into its normalized form. For each representation we also show the influence of using sequential modeling, where we can observe improvement of average classification accuracy of about 5% and 28% for letter-based and context-based representation respectively. We report average classification accuracy and standard deviation on 5-fold cross validation experiments and compare the results with the default classifier assigning the majority class value.

sons for making errors in prediction. One is the small number of words in the representation of some examples, which resulted in unreliable model as well as predictions based on only a couple of words. Figure 6.1. illustrate the problem of a testing document "sharing with the class values model" none or only small number of words. This means that, since for all other words W_i from the testing document $TF(W_i, C) = 0$, the classification is based only on a limited evidence. We show some examples where the classification was not correct. The first example illustrates how two words from different class value having similar context can have negative influence on the classification results. Namely, the testing example contains two personal names "Cankarjem" and "Presernom". They are both "instrumental case" but have different suffix. Thus, the personal name "Cankarjem" should be replaced by "Cankar" where the correct mapping is cutting off the ending "jem" and not cutting off the ending "om" as predicated. However, the predicted transformation is correct if the same context document is formed for mapping the other personal name "Presernom", which should be mapped to "Presern". This example also illustrates potential problem of using the context words as feature values, since the same or very similar context can belong to different class values. A typical example is when two words having different class value tend to occur close to each other in the texts and thus have similar context. In case when the two words have different last letter, it helps to use the proposed sequential modeling resulting in decomposition of the problem into a set of subproblems based on the last letter of the word.

We expect that using a larger set of unlabeled data with better overlap with the labelled words will produce more reliable probability estimates. Having a larger number of context for the same word should also help with the second problem, since in general, it is not likely that two words will repeat appearing together in different documents (unless they represent some kind of a phrase).

7. Conclusions and Future Work

We have described an approach to word classification which adopts a simple sequential modeling to obtain a set of simpler problems. These new problems mostly have up to 10 class values, instead of 156 class values that we have in the original problem. Our experiments show that this resulted in an improvement of classification accuracy of 5% and 28% on the two independent example representations we used. The baseline classifier we used assigns the majority class values to all examples. Its performance improved

```
(a)
Document vector: BILO SE S SEM TRDINO PRESERNOM CANKARJEM
[51:JEMTo_]   --> [5:OMTo_]:0.79520
        Model evidence:TF(SE)=7.0 TF(S)=4.0 TF(SEM)=2.0

(b)
Document vector: NI STORIL GORENJCEM PREZIR
[43:CEMToEC] --> [22:MTo_]:0.54037
        Model evidence: TF(NI)=2.0
(c)
Document vector: V V V V V V V V JE IN NAJ NA NA TUDI SO NI SLA DA DA STA ROVAN
        NEKAJ GRE DRUGEGA HVALO POJE PRECEJ SAMEGA ODSEL KONCU PAPEZA RIM RIM RIM RIM RI
        M RIM RIM RIM RIM RIM SIBKIH ROMANJE RIHARJU KOGA CYRIACUS
[20:_ToA]   --> [22:MTo_]:0.96590
        Model evidence:TF(V)=3.0 TF(JE)=11.0 TF(IN)=7.0 TF(NA)=49.0
          TF(TUDI)=2.0 TF(SO)=3.0 TF(NI)=2.0 TF(DA)=2.0
```

Figure 3: Illustration of classification errors. (a) Two words having different class values appear together in the testing example: personal names: PRESERNOM, CANKARJEM. The predicted classification is correct for one of them, in this case not the one that was used to collect its context. (b) Prediction based on very little model evidence is not very reliable, as in case when only one word from the testing example is used. The word for which we are predicting transformation here is GORENJCEM. The proposed transformation actually gives a valid (but not normalized) word which is a noun in accusative, plural (instead of a normalized variant, which is nominative, singular) (c) More words in the representation of the testing document do not necessary lead to correct prediction, especially if we are dealing mainly with functional words. Moreover the proposed transformation is not applicable to the word, since the words ends with "A" while the transformation calls for cutting off "M". The word for which we are predicting transformation here is PAPEZA (Pope). .

from 12% to 40.8% due to the usage of sequential modeling.

We propose two independent feature sets, one based on the word letters and the other based on the context of the word in unlabeled documents. The letter-based word representation achieved better results than the context-based representation. We expect that some of the problems we observed with the context will be solved by using a larger set of unlabeled documents containing the words from our labeled data with higher frequency.

Nevertheless, the two independent features sets motivate future work on combining them using co-training. It is also interesting to investigate the potential of using a similar approach combining "classical" feature-based and context-based representation on other related problems, such as abbreviations resolution or phrases mapping.

Using unlabeled data for post-processing the classification results is also an interesting direction for future work. We can apply additional scoring on the predicted class values based on our believe that the transformed word is a valid word in the addressed natural language. Namely, we know that our mapping should produce valid words in the same language (a normalized word is a valid word itself) and we can check if the transformed words appear in the unlabeled texts. One possibility is that we have a dictionary with all normalized words from the language. In that case we can use it not only for checking if the word is valid but also to get some alternative classifiers by combining the supervised learning using our labeled data with the unsupervised learning similar to what is proposed in (Yarowsky, 1995) for getting morphological transformations. The other possibility is to just get some texts in the same language and use them to check the result of each testing example classification. In that case, we do not have a complete set of all possible normalized words but we have a large set of valid words that hopefully contains many of the normalized words as well.

8. References

Blum in Mitchell. 1998. Combining labeled and unlabeled data with co-training. V: *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

B. Cestnik in I. Bratko. 1991. On estimating probabilities in tree pruning. V: Y. Kodratoff, ur., *Proc. Fifth European Working Session on Learning*, str. 151–163, Berlin. Springer.

Saso Dzeroski in Tomaz Erjavec. 2000. Learning to lemmatise slovene words. V: *Learning language in logic, (Lecture notes in computer science, J.Cussens and S.Dzeroski (eds))*, str. 69–88.

Yair Even-Zohar in Dan Roth. 2001. A sequential model for multi-class classification. V: *Proc. of Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*.

Rayid Ghani, Rosie Jones, in Dunja Mladenic. 2001. Automatic web search query generation to create minority language corpora. V: *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

C. X. Ling. 1994. Learning the past tense of English verbs: The symbolic pattern associator vs. connectionist models. *Journal of Artificial Intelligence Research*, 1:209–229.

A. McCallum in K. Nigam. 1998. A comparison of event

models for naive bayes text classifiers. V: AAAI-98 Workshop on Learning for Text Categorization.

Dunja Mladenic. 1993. Combinatorial optimization in inductive concept learning. V: *Proc. 10th Int. Conf. on Machine Learning, Morgan Kaufmann*, str. 205–211.

R.J. Mooney in M.E. Califf. 1995. Induction of first-order decision lists: Results on learning the past tense of english verbs. V: L. De Raedt, ur., *Proceedings of the 5th International Workshop on Inductive Logic Programming*, str. 145–146. Department of Computer Science, Katholieke Universiteit Leuven.

M.F. Porter. 1980. An algorithm for suffix stripping. V: *In ACM SIGIR Conference on Research and Development in Information Retrieval*, str. 318–327.

M.M. Syslo, N. Deo, in J.S. Kowalik. 1983. *Local search heuristics*. Prentice-Hall. Inc. Englewood Cliffs.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. V: *Meeting of the Association for Computational Linguistics*, str. 189–196.