Morphology Learning Using Tree of Aligned Suffix Rules

Ksenia Shalonova KSENIA@CS.BRIS.AC.UK
Peter Flach PETER.FLACH@BRIS.AC.UK

Department of Computer Science, University of Bristol, Bristol, UK

Abstract

Linguistic morphology concerns the structure of words, for instance how the plural form of nouns is obtained from their singular form, or how the past tense of verbs is generated from their infinitive. We describe an approach for function learning in morphology, where given a basic form of a word the goal is to generate a grammatical wordform. This task concerns learning the regular grammar or transformation in the format stem+suffix1 -> stem+suffix2 where suffix1 and suffix 2 are suffixes of two words in one word-pair and stem is the coinciding part in two words. Our approach is based on the tree of aligned suffix rules (TASR) where suffix rules represent left-hand and right-hand word suffixes of the input word pairs. The tree is built top-down, from general rules to specific rules, using suffix rule frequency and rule subsumption to decide which rules go where in the tree. The tree is executed bottom-up, i.e., the most specific rule that fires is chosen. In comparison to rule-induction approaches with similar functionality from the literature, the proposed method is faster, generates less rules and combines the following set of properties: it is relatively simple, achieves high performance and has a more clear linguistic interpretation. We also describe preliminary thoughts on inducing morphological rules that are close to the complexity of a context-free grammar.

1. Introduction

1.1 Framework of the Current Research

Morphology is the study of the internal structure of words that can be represented in terms of regular and context-free rules. Morphological structure of most European languages can be represented as finite state automaton, whereas in some African and Asian languages morphological phenomena go beyond the capability of finite state automaton and require using memory.

Morphological Analysis is used in many different domains of Speech and Language Technology, including text-to-speech (TTS) systems, automatic speech recognition, machine translation, information retrieval and spell-checkers. Although a lot of work has been done for morphology learning, by far the most common way of creating morphological analysers is the rule-based approach. In this approach, finite-state techniques are used for decomposing the word structure into its constituents. The rule database consists of the available stem/affix lists with grammatical features, morphotactic rules governing their concatenation, and orthographic rules that change the shape of word constituents. Building the rule database for morphological analysers by hand (stem/affix lists, morphotactic and orthographic rules) is extremely time-consuming and can take several years for a single language.

Morphology learning or automated techniques for generating morphological rules for morphological analysis can lead to great savings in time and resources for the languages of interest. The current research was carried out in the framework of the project that aims at creating Speech/Language applications for indigenous languages (www.llsti.org) where key linguistic resources (in particular, rule databases for morphological analysers) are unavailable, as is funding to produce such linguistic resources. In this project we are concentrating on synthetic languages with complex morphological rules. We started to learn regular morphological grammars for Russian language, but this task will be extended in future by learning morphological phenomena that go beyond the capability of finite state automaton: reduplication, vowel harmony and infixes.

1.2 Related Work

There are three main types of learning morphological grammars.

First of all, supervised learning of morphological rules is based on the annotated data as the alignment of orthographic words with their morpheme representations (for example, the morpheme representation *box^s* is aligned with orthographic word *boxes* where ^s and *es* represent the plural noun suffix). In order to apply supervised learning methods, the data should further be

CAGI Workshop, 2007

extended with information about inflectional classes and features (part-of-speech tags), thus making the output of the training mechanism compatible with the input for morphological analysers. Instance-based learning is one of the most popular mechanism of supervised learning of orthographic rules (van den Bosch & Daelemans, 1997; Oflazer et al., 2001). Oflazer provided an interactive supervision method in which, during the training process, language experts provide the required feedback and corrections. Theron and Cloete (1997) learned two-level rules for English, Xhosa and Afrikaans, but only single character insertions, replacements and additions were allowed. The supervised algorithms are normally learning the rules for two-level morphology presented in (Koskenniemmi ,1983).

Secondly, there is unsupervised learning of morphological rules. The advantage of unsupervised morphological learning is that it requires only the set of orthographic words (completely "raw data") without any stem/affix lists or grammatical annotation. However, most of the unsupervised methods put restrictions on the number of morphemes per word, on rule complexity etc. The goal of these methods is merely splitting a word into stem and suffix without the capability of assigning any grammatical information. A number of unsupervised learning techniques have been applied: genetic algorithms (Kazakov, 1997), a minimal description length approach based on spelling of words and the set of suffixes that appear with each stem (Goldsmith, 2001), and the quasi-roots algorithm (Sheremetyeva and Nirenburg, 1999)

And finally, there is another approach that can be classified in between supervised and unsupervised learning where the input for training are the following pairs of words: word in its basic form and word in its grammatical form. Several Inductive Logic Programming (ILP) techniques (Mooney & Califf, 1996; Muggleton & Bain, 1999; Erjavec & Dzeroski, 2004) as well as statistical approaches (Clark, 2002) were applied in the framework of this learning approach. The algorithm described in this paper also takes wordpairs as an input for morphological learning. This approach can be considered as a compromise between the quality of the obtained rules and the amount of effort required to produce the training set.

As was mentioned earlier, Machine Learning (ML) techniques for morphology learning are hardly used for developing morphological analysers for new languages of interest. Combining ML techniques for morphology learning with linguistic theories may increase the chances that these ML techniques will be widely used in speech/language technologies (Kazakov, 2000). Another important factor is the relative simplicity and good performance of the ML algorithms. The ultimate goal is to get the most accurate and minimal number of morphological rules that can be induced in a reasonable amount of time. As the aim of the on-going project is automatic induction of morphological rules for languages with a complicated morphology, we started our experiments with the Russian language as in Russian every basic form generates a large number of grammatical forms – up to 200 forms. The time issue is quite important for the current project as the induction of rules will be used in real-life applications.

The algorithm for learning morphological rules presented in this paper has got almost the same functionality as FOIDL (Mooney & Califf, 1996) and CLOG (Erjavec & Dzeroski, 2004):

- The algorithm provides the function learning (given basic word form – generate grammatical form of this basic form).
- The algorithm does not generate overlapping rules.
- The algorithm does not require explicit negative examples.

The performance of our algorithm does not depend on the order of the training examples, whereas both FOIDL and CLOG are using a hill-climbing strategy and are sensitive to example order.

Another difference is that FOIDL and CLOG algorithms provide the rules in the form of a list (a set of ordered rules), whereas our algorithm provides the rules in the form of a tree. Both FOIDL and CLOG are first-order decision list learning systems with the difference in treating generalisations: CLOG considers generalisations that are only relevant to an example and also treats a set of generalisations as a generalisation set. This makes CLOG more efficient than FOIDL. Our approach uses a tree structure that makes the rule search more efficient

In this paper we compare the results of our algorithm that we call Tree of Aligned Suffix Rules (or TASR) with CLOG and FOIDL on English Past Tense data and with CLOG on Russian nouns. The comparison with CLOG and FOIDL is carried out not only because their similar functionality, but also due to the fact that these systems are freely available on-line with all settings required for the morphological task. This makes possible to carry out accurate comparison using the same training and test data. Other ILP systems were tried, but proved to be insufficiently robust to give comparative results.

It is worth mentioning another related work for automatic word lemmatization (given a grammatical wordform – generate its basic form) that in some respect has got some similarities to our approach (Mladenic, 2002).[1] The similar transformation task is being solved in this work: learning rules of the format X TO Y where X and Y are suffixes of two words in one word-pair. They applied ATRIS algorithm with local optimization using up to five last letters of a word (five discrete features) and the rules were represented as binary features for the 2-opt search algorithm. The assessment of rule quality at each optimization loop used a combination of heuristics and one

[1]It is important to notice that our algorithm can also be used for lemmatisation task with the rules presented in the opposite direction.

of them seems to be rule precision (corrected with m-estimate).

Our algorithm is not restricted to the number of suffix length , it is less complicated and is represented in the form of an explicit tree that is readable and clear for human interpretation.

2. The Algorithm for Suffix Rule Induction

2.1 The concept of TASR

Before defining the training algorithm itself, we want to present the concept of the Tree of Aligned Suffix Rules (TASR) that is used for learning morphological rules. By morphological rule we mean the rule in the following format: left-hand suffixes of the basic word form (LHsuff) and the corresponding aligned right-hand suffixes of the grammatical form (RHsuff). As in our case morphological rules are derived only from the word terminal position, we call them suffix rules.

Definition 1 (Aligned suffix rules) *Rule LHSuff->RHSuff is aligned with rule LHSuff'->RHSuff' if there is a wordpair with left-hand word and right-hand word LW-RW for which LW=s+LHSuff=s'+LHSuff' and RW=s+RHSuff=s'+RHSuff' where s and s' are the prefix substrings in LW and RW, and "+" denotes string concatenation.*

For example, the wordpair *stay-stayed* generates the following aligned suffix rules: stay->stayed, tay->tayed, ay->ayed, y->yed and 0->ed where 0 denotes the zero-length suffix or an empty string.

Definition 2 (TASR) *Tree of Aligned Suffix Rules (TASR) is a tree with the following properties:*

- *The root of the tree is the LHSuff->RHSuff rule with the minimal length of LHSuff.*
- *Nodes represent aligned suffix rules in the form LHSuff->RHSuff . The winning rule has got the highest frequency (Definition 3) and is not subsumed (Definition 4) by the rule in a parent node.*
- *LHsuff' in child node is represented by lefhand one letter expansion of LHSuff in parent node.*

In general words, TASR represents the transducer rules generalised in the following format: s+LHsuff->s+RHsuff where s is the coinciding part of the string, LHSuff and RHSuff are terminal substrings and "+" denotes string concatenation.

Definition 3 (Suffix rule with higher frequency) *Rule LHSuff->RHSuff has got higher frequency than rule LHSuff->RHSuff ' if the number of wordpairs Lw-Rw where Lw=s+LHsuff and Rw=s+RHSuff is larger that the number of worpairs where Lw=s+LHsuff and Rw=s+RHSuff '.*

Definition 4 (Rule subsumption) *Child node LHSuff->RHSuff is subsumed by parent node LHSuff'->RHSuff' if there exists a character x such that Lhsuff=x+LHsuff' and RHSuff=x+RHSuff'.*

The example of the tree outlined below (Figure 1) is suitable for generating English Past tense given a basic word terminating on *y*. This model for generation of morphological rules is suitable for the languages where the grammatical information is preserved in the word terminal position or in word suffix (like in English and Russian languages). The tree is built on the basis of the LHsuff where the root of the tree contains LHsuff equal to "0" or an empty string. Children of each node are generated by lefthand one letter expansion of LHSuff.

Suffix rules in Figure 1 are presented in several font types that define several rule properties used for building the TASR :

- bold font: winning rules for English Past Tense generation (the target ruleset). The winning rule can be only the rule that has got higher frequency than other rules with the same LHsuff at a particular node (see Definition 3). For example, the rule y->ied has got higher frequency than rules y->yed and y->id.
- italic font: the rules that are not marked as winning ones at a particular node due to the fact that their frequencies are lower that those of the winning rule (Definition 3). It is interesting to notice that for each rule with the same LHsuff in TASR the calculation of the precision measure p/(p+n) is equal to the calculation of the frequency of each rule (where p - number of positive and n – number of negative examples). For example, while selecting the rule with the highest frequency among the rules y->ied, y->yed and y->id the positive examples for the rule y->yed will be all wordpairs of the form *s*+y->*s*+yed, whereas the number of negative plus positive examples for this rule will be the same as for the other two rules[2] . The rules in italic are not chosen at the current node, but they are represented at lower levels of the tree by more specific rules, for example, the rule y->id is represented by the rule say->said at the lower level. Thus such a tree representation selects as general rules as possible for the frequent suffix rules where the less frequent suffix rules are represented at lower tree levels by more specific rules.
- normal font: rules that are subsumed by the rules in the parent node and thus are not marked as the wining ones. The term subsumption is explained in Definition 4.

[2] The negative examples for the rule y->yed are the wordpairs in the training set of the following form: stem+y->stem+ied and stem+y->stem+id.

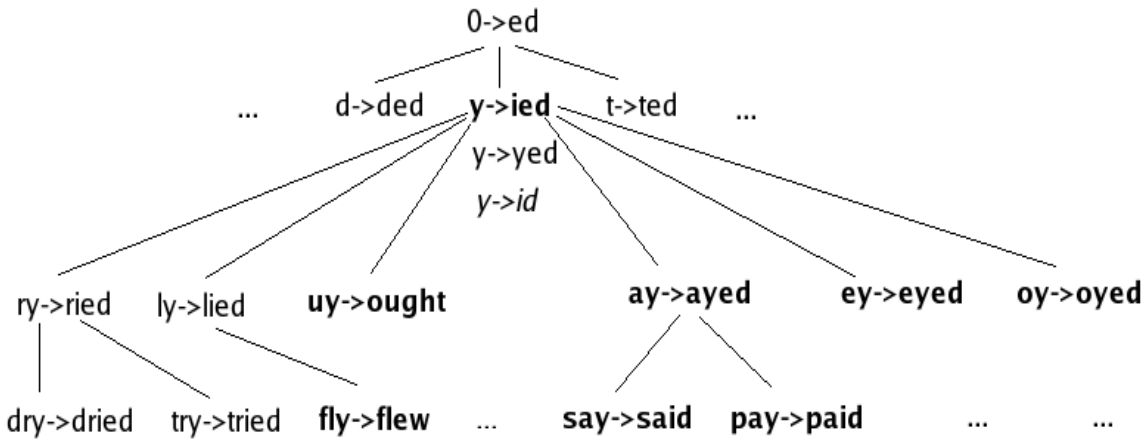

*Figure 1.*TASR for English verbs terminating on letter "y".

The rules for the target ruleset are selected on the basis of the following principle: if a suffix rule at a particular node is NOT subsumed by the rule in the parent node than this rule is selected as the target rule. The term "rule subsumption" is quite natural from the linguistic perspective (for example, rule y->ied is NOT subsumed by rule ay->ayed , whereas rule y->ied is subsumed by rule ry->ried. Thus the winning rule has got the following conjunction of criteria:

- it has higher frequency than other rules with the same LHSuff;
- it is not subsumed by the rule in the parent node.

The tree is searched bottom-up for all rules that are marked as winning ones. It is important to notice that bottom-up search of TASR is more natural from the linguistic perspective as such a search order represents a certain hierarchy of suffix rules with the leaves of the tree that normally represent word exceptions. For example, the rules say->said and pay->paid can be considered as word exceptions whereas the rule 0->ed is the most general and default rule. In order to generate a past tense form from the basic form, the tree is searched bottom-up by carrying out the following matching procedure: if the suffix in basic form matches the LHsuff at a particular node, than the rule containing this LHsuff is applied and past tense is generated. For example, in order to generate the wordform *stayed* given the basic form *stay*, the match search starts from the leaves and goes one level up until the LHsuff *ay* from the rule ay->ayed matches the basic form suffix *ay* in *stay* and the wordform *stayed* is generated.

The tree presented above can be expanded for the whole set of words terminating on different letters. The complexity of this tree is

$$O(n*m\log n*m) \quad (1)$$

given n as a number of word pairs and m as the average length of a basic form.

2.2 The Training Algorithm

In order to get the suffix ruleset from the training data, the following algorithm is applied.

Algorithm 1 Extract List of Aligned Suffix Rules

Input: set of wordpairs: Lw-Rw
for each word pair$_i$: <*Lw*> -<*Rw*>
s= first letter in word Lw;

proc extract aligned suffix rules (s)

1. Extract the rule LHsuff->RHsuff if it fits the following match: *Lw = s + LHSuff* and *Rw = s + RHSuff* where s is the coinciding string;
2. s' = s where substring s' is expanded by next letter after letter s from Lw.
3. Extract aligned suffix rules (s').

end each

In order to building the tree on the basis of the alinged suffix rules and their frequencies the following algorithm is applied.

Algorithm 2 Build a Tree on the basis of Aligned Suffix rules

Input: set of unique aligned suffix rules with their frequencies

Current node of the tree=Root;

LHSuff="0" (or empty string);

proc Build_Tree (LHsuff)

1. For LHsuff find the Rhsuff with largest frequency (see Definition 3);
2. Current node of the tree = LHsuff -> Rhsuff with largest frequency;
3. Check subsumption criterion

if the rule LHsuff->Rhsuff does not SUBSUME LHsuff-> Rhsuff (see Definition 4)

then mark the rule LHsuff-> Rhsuff as the winning one .

end if

4. For all lefthand 1-letter extensions LHsuff' of LHsuff call Build Tree (LHsuff') and add the resulting sub-tree as child of Current node.

3. Experimental Results

The first data type consist of 1392 wordpairs of English basic forms and their past tense forms (the wordpairs are represented in alphabetic form). We are using alphabetic representation (not the phonetic one) as it is much more likely that the induction of morphological rules for indigenous languages will be based on orthography that is cheaper and less time consuming to generate. English Past Tense data is a good choice for performance comparison as they include a lot of exceptions and thus are difficult to train.

The second data type comprises 1000 most frequent basic word forms for Russian nouns (Sharoff, 2005). For each of these basic forms a number of grammatical forms in singular form was constructed for five different cases on the basis of the available morphological dictionary for Russian "Zaliznyak Morphological dictionary" . For each of five cases 1000 randomized pairs of basic and grammatical forms were tested using 10-fold cross-validation method. Thus for each folder 900 training examples and 100 test examples were used.

Table 1 shows the comparison of a 10-fold cross-validation tests using CLOG and TASR. As can be seen from the results, the performance of the TASR algorithm is better than that of CLOG, but the time required for running these two algorithms differs drastically (Table 2). The number of rules generated by the TASR algorithm is also smaller that that generated by CLOG (Table 3). CLOG in comparison to the current algorithm generates a small amount of rules (one or two for each fold) that takes into account the prefix information. For example, if the basic word starts with letter "s" and terminates on "end" than the past tense form is generated by substituting terminal "end" into "ent" like in *send-sent*. Although the amount of such rules generated by CLOG is small and hardly influences the results, it is possible to assume that this kind of information is not required for the languages where the grammatical information is present only in the word terminal position.

Table 1. Comparison of CLOG and TASR performance for each data set.

Data Set	CLOG (mean over 10 folds)	TASR (mean over 10 folds)	T-test Better?
English Past Tense	86.4±2.8	88.3±3.6	✓
Genitive Case (Russian)	90.1±3.7	92.1±3.2	✓
Dative Case (Russian)	90.4±3.3	92.7±3.4	✓
Instrumental Case (Russian)	89.8±3.7	90.6±4.1	-
Prepositional Case (Russian)	90.8±3.3	91.7±3.3	-
Accusative Case (Russian)	87.2±6.0	84.1±5.9	x

The performance of FOIDL on English Past Tense data gave 82.2% (in comparison to 88.3% of TASR) that made a significant difference with the performance of TASR due to t-test. The average running time of FOIDL on 1242 examples is approximately 90 min and the average number of generated rules is 175.

Table 2. Comparison of time required for inducing rules by CLOG and TASR.

Data Set	CLOG (average time)	TASR (Average time)
English Past Tense (1242 examples)	3 min	0.3 min
Russian Case (900 examples)	1.5 min	0.12 min

Table 3. Average number of rules generated by CLOG and TASR for each data set.

Data Set	Average Number of Generated Rules	
	CLOG	TASR
English Past Tense	183	159
Genitive Case (Russian)	99	79
Dative Case (Russian)	95	65
Instrumental Case (Russian)	103	86
Prepositional Case (Russian)	95	75
Accusative Case (Russian)	139	114

Table 4. TASR performance with differentiating the gender for four cases and with differentiating animation feature for Accusative case.

Data Set	Feature_Value	Performance (%)
Genitive Case	F	99.3
	M	96.3
	N	99.6
	Total	**98.4**
Dative Case	F	99.3
	M	96.7
	N	99.6
	Total	**98.5**
Instrumental Case	F	99.1
	M	94.8
	N	99.3
	Total	**97.7**
Prepositional Case	F	99.3
	M	96.3
	N	99.2
	Total	**98.3**
Accusative Case	Animated	97.5
	Not_Animated	97.8
	Total	**97.7**

The analysis of the errors on the test data for the TASR algorithm revealed that most of errors can not be fixed in principle (for example, when one of the wordpairs *light-lit or fight-fought* is present in the training set and another in the test set the error is inevitable regardless the algorithm you are using).

Features such as gender and animation are two main features of a word basic form in Russian which may affect the choice of a suffix rule. These features were assigned to each of the word pairs and the algorithm was run both taking into account and ignoring these features. When the features were taken into account, the TASR was built for each feature value (for example, the TASR was built for animated and not-animated wordpairs). The results of the tests on Russian singular nouns are represented in Table 4.

The obtained results show that the performance of the algorithm increases when for four cases (Relative, Dative, Instrumental and Prepositional) the feature gender is taken into account. For accusative case the performance increases significantly when the feature animation is taken into account. The mentioned features were taken into account during training on the basis of the available linguistic knowledge. The future task is to determine automatically the relevance of the features for inducing rules of a particular grammatical category. Experiments with chi-square applied for the root of the tree gave promising results.

4. Preliminary Thoughts on Inducing Regular Languages with Memory

The principle of rules subsumption (see Definition 4) and of tree structure can be used for inducing rules that go beyond the capability of regular expressions. We are currently working on this task and can only describe some preliminary thoughts.

A classical example of such complicated rule is vowel harmony when vowels in a particular word are assimilated to a vowel in a terminal word position in order to generate another grammatical form.

For example, word "manosholu" is transformed into the word "manushulu". The general rule for this transformation is that all vowels in a word except the first vowel are assimilated to the final vowel "u". If we regard only vowel sequence the transformation can be written in the following regular expression notation: ^V.(V)*.U#-->^V.(U)*.U# where V-any vowel, ^-initial position in a sequence, U – vowel U, #- terminal position in a sequence and (V)* - the sequence of any vowels.

Part of a tree for vowel rule representation is outlined below where the rules in italic font represent the rules that can be subsumed (see Definition 4) by the rules in the parent node and thus are not selected as the winning rules, whereas the rules in bold are the winning rules as they are not subsumed by the rules in the parent node.

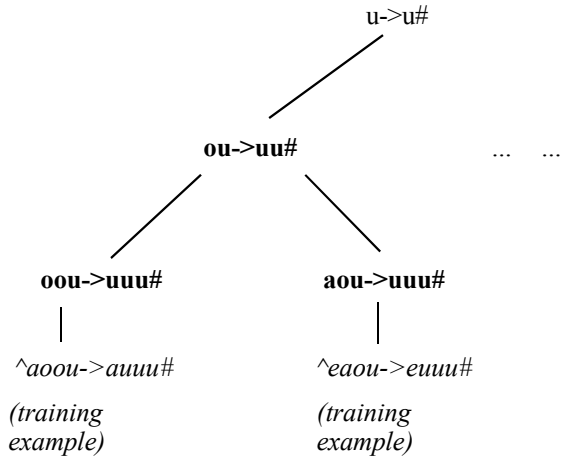

Figure 2. Inducing Specific Rules for Vowel Harmony using subsumption criterion.

Introducing regular expressions into the nodes of the tree allows more generalisations for the vowel harmony rule (Figure 3).

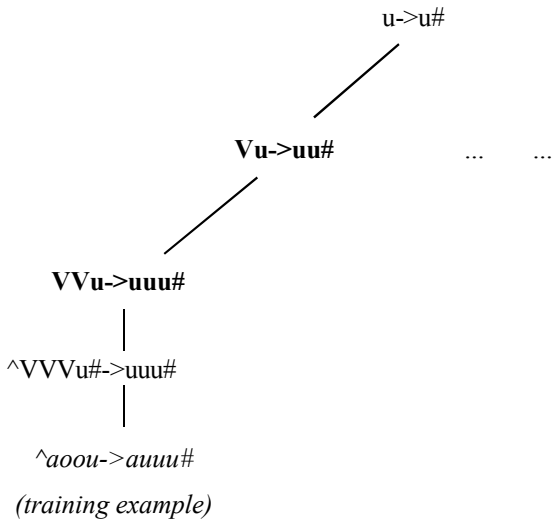

Figure 3. Inducing General Rules for Vowel Harmony using subsumption criterion.

For each general rule in a tree node (expressed by a regular expression) we calculate the number of positive and negative examples from the training set and on the basis of such statistics we make a decision either to mark this rule as a possible winning rule or not. For example, for the general rule VVu->uuu# the negative example is aau#->uau# and the positive example will be oou#->uuu#. The ultimate decision about the winning rule is take on the basis of the subsunption criterion.

As this algorithm is still under development, we can't give any precise results yet.

5. Conclusions and Future Work

We have developed the TASR algorithm that is based on rule frequencies and rule subsumption. It seems that the concept of rule subsumption is quite powerful and will be explored in future for learning regular languages with memory that is close to learning context-free grammars.

There is also a set of immediate tasks to be fulfilled for learning regular grammars:

1. Implement stochastic approaches to solve the current task like, for example, pair HMM (Clark, 2002) and compare the results with the TASR algorithm. It is important to notice that the fact that the TASR algorithm is not stochastic has got the advantage of representing exceptions (isolated example in the leaves of a tree) and ambiguities that is very important for language learning (Daelemans et al., 1999).
2. As the current algorithm can define only one boundary between the stem and a suffix (or a prefix), the future task is to induce both prefixes and suffixes

simultaneously. The possible approach will be creating two TASR trees of potential suffixes and prefixes and based on the statistics of their rules make the decision about the stem. This approach can be tested on the Turkish language. When the stem is extracted, an expansion of this task is decomposition of a suffix (or a prefix) sequence into their sub-sequencies.

3. As the choice of word pairs is crucial for the TASR algorithm, another important task is an efficient selection of word pairs from a set of words. We believe that for learning morphology "from scratch" (when there are no available resources like for some African languages), we can implement an incremental (or active) learning when questions can be asked at each iteration and on the basis of the answers the choice of wordpairs can be made.

Acknowledgments

The research is sponsored by the EPSRC grant "Learning the morphology of complex synthetic languages".

References

Clark, A. (2002). Memory-Based Learning of Morphology with Stochastic Transducers. *Proceedings in the Annual Meeting of Association of Computational Linguistics*. pp. 513-520.

Daelemans, W., van den Bosch, A. & Jakub Z. (1999). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, volume 34, pp. 11-41.

Erjavec, T., & Džeroski, S. (2004). Machine Learning of Morphosyntactic Structure: Lemmatising Unknown Slovene Words. *Applied Artificial Intelligence* 18(1), pp. 17-40.

Goldsmith, J. (2001). Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics 27*, pp. 153-198.

Kazakov, D. (2000). Achievements and prospects of learning word morphology with inductive logic programming. In Cussens & Dzeroski (Ed.), *Learning Language in Logic*, pp. 89-109.

Koskenniemmi, K. (1983). Two-level Morphology: A General Computational Model for Word Form Recognition and Production. *PhD Dissertation*. University of Helsinki, Department of General Linguistics.

Mladenic, D.(2002). Learning word normalization using word suffix and context from unlabeled data. *ICML 2002*, pp. 427-434.

Kazakov, D. (2000). Achievements and prospects of learning word morphology with inductive logic programming. In Cussens & Dzeroski (Ed.), *Learning Language in Logic*, pp. 89-109.

Mooney, R. and Califf, M. (1996). Learning the Past Tense of English Verbs Using Inductive Logic Programming. *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pp. 370--384.

Muggleton, S. & Bain, M. (1999).Analogical prediction. In *Proc. of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, Berlin.

Oflazer, K., McShane M. & Nirenburg S. (2001). Bootstrapping Morphological Analyzers by Combining Human Elicitation and Machine Learning. *Computational Linguistics. Vol. 27, Issue 1*, pp. 59-85.

Sharoff, S. (2005) Methods and tools for development of the Russian Reference Corpus. In Archer, Wilson & Rayson (Ed.), *Corpus Linguistics Around the World*, pp. 167-180.

Sheremetyeva S. & Nirenburg, S. (1999). Porting a Rapid Morphological Analyser Across Slavic Languages. *Proceedings of the 8th Annual Workshop on Formal Approaches to Slavic Linguistics.* Philadelphia.