LEARNING RIPPLE DOWN RULES FOR EFFICIENT LEMMATIZATION

Matjaž Juršič, Igor Mozetič, Nada Lavrač
Department of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
e-mail: matjaz@gmail.com, {igor.mozetic, nada.lavrac}@ijs.si

ABSTRACT

The paper presents a system, LemmaGen, for learning Ripple Down Rules specialized for automatic generation of lemmatizers. The system was applied to 14 different lexicons and produced efficient lemmatizers for the corresponding languages. Its evaluation on the 14 lexicons shows that LemmaGen considerably outperforms the lemmatizers generated by the original RDR learning algorithm, both in terms of accuracy and efficiency.

1 INTRODUCTION

Lemmatization is the process of determining the canonical form of a word, called *lemma*, from its inflectional variants. Lemmas correspond to headwords in a dictionary. An alternative approach to abstract the variability of word-forms is *stemming* which reduces the word to its root or stem. For example, in Slovene, the word-forms *pisati, pišem, pišeš, pišemo* have a common lemma *pisati*, and a common stem *pi*. For text analysis and knowledge discovery applications, lemmatization yields more informative results then stemming. However, both problems are closely related and the approach described here can be applied to stemming as well.

The difficulty of lemmatization depends on the language. In languages with heavy inflection, such as the Slavic languages, stems can combine with many different suffixes, and the selection of appropriate ending and its combination with the stem depends on morphological, phonological and semantic factors. As a consequence, lemmatization of highly inflectional languages is considerably more difficult then the lemmatization of 'simple' languages, such as English.

In computer science, the problem of stemming and lemmatization was addressed already in 1968 [1]. For English, the problem is considered solved by the Porter stemmer [12]. However, the Porter stemmer was hand crafted specifically for English and is not applicable to other languages, specially those with heavy inflection. Manual development of a lemmatizer requires involvement of a linguistic expert, and is an impractical and expensive undertaking. An alternative is to use machine learning tools for automatic generation of lemmatization rules. There have already been several approaches to learning lemmatization rules:

- 1993-2002: A rule induction system ATRIS [8,9]
- 2002: If-then classification rules [7]
- 2002: Naïve Bayes [7]
- 2004: A first-order rule learning system CLog [3]
- 2004: Ripple Down Rule (RDR) learning [10, 11].

This paper is focused on Ripple Down Rule (RDR) learning. The RDR learning approach was originally proposed as a methodology for the GARVAN-ES1 expert system maintenance [2]. The idea is that the rules are incrementally added to the system. When new examples of decisions are available, new rules are constructed and added to the system. However, already existing rules might contradict some new examples, therefore exceptions to the original rules have to be added as well.

In this paper we describe an improved Ripple Down Rule (RDR) learning system called LemmaGen [6], especially tailored to the problem of word lemmatization. In Section 2 we describe the RDR format, how the rules can be applied to lemmatization and how is the RDR structure automatically constructed from the lemmatization examples by LemmaGen. In Section 3 we describe the application of LemmaGen to 14 different language lexicons, compare the results with an alternative RDR implementation, and evaluate the performance in terms of lemmatization accuracy, efficiency, and applicability of the approach to different languages.

2 LEARNING RIPPLE DOWN RULES

RDR rules form a tree-like decision structure with an obvious interpretation:

```
if A then C
   except if B then
E
   except if D then
F
else if G then H
```

Rules and their exceptions are ordered, and the first condition that is satisfied fires the corresponding rule. In addition, explanation is also provided. Every `if-then' rule is augmented by its explanation in terms of the `because of' appendix, which lists one or more training examples covered by the rule (examples which `fire' for the given rule), which – in the process of learning - caused the individual rule to appear in the rule list.

In the case of lemmatization, general concepts that appear in RDR rules are instantiated to domain specific terms:

- Training examples are pairs *(word-form, lemma).*
- A rule condition is a *suffix* of the word-form which ends the word that fires the rule.

```
`---> RULE:( suffix("") transform(""-->"") except(3) );
      |---> RULE:( suffix("i") transform("i"-->"o") except(4) );
      |      |---> RULE:( suffix("li") transform("li"-->"ti") );
      |      |---> RULE:( suffix("ni") transform("ni"-->"ti") );
      |      |---> RULE:( suffix("ti") transform(""-->"") );
      |      `---> RULE:( suffix("ši") transform("ši"-->"sati") );
      |---> RULE:( suffix("l") transform("l"-->"ti") );
      `---> RULE:( suffix("mo") transform(""-->"") except(2) );
             |---> RULE:( suffix("šemo") transform("šemo"-->"sati") );
             `---> RULE:( suffix("šimo") transform("šimo"-->"sati") );
```

Figure 1: *A part of the RDR tree structure, constructed by LemmaGen for the lemmatization of Slovenian words.*

- A rule consequent is a transformation which replaces the word-form suffix by a new suffix, thus forming the lemma. The transformation is written as *{word-form suffix} -> {lemma suffix}*.

Some example RDR rules for the lemmatization of Slovenian are given in Figure 1.

The original RDR learning algorithm, adapted to learn the lemmatization rules, and applied and evaluated on the Slovenian lexicon is described in [10, 11]. We have applied this RDR algorithm to several additional language lexicons and investigated the means of possible improvements. The new algorithm, LemmaGen, implements the following improvements:

- The original RDR algorithm processes training examples sequentially and does not take into account the number of examples covered by individual rules and their exceptions. As a consequence, a rule high in the RDR hierarchy ('default rule') might cover just a small fraction of examples with a non-typical transformation, and have a large number of exceptions itself. LemmaGen performs lexicographical ordering of training examples (starting from the end of words) and orders rules and exceptions by the frequency of examples.
- As there are identical word-forms with different lemmas, the nodes in the RDR tree cannot distinguish between different transformations. The original RDR algorithm simply selected the first transformation it encountered, while LemmaGen selects the most frequent transformation.
- The LemmaGen learning algorithm is considerably faster then the original RDR. It achieves speedups between factors 2 and 10, depending on the lexicon used for learning. Due to more compact RDR trees produced, the lemmatization is also considerably faster, between 10 and 40 fold. Improvements in the efficiency of learning and lemmatization are in Figure 4.

If *N* is the number of training examples, and *M* is the length of the longest word in the lexicon, then the time-complexity of our learning algorithm is *O(2*N*M)*. The worst-case time complexity is therefore linear in the number of examples.

3 APPLICATIONS ON THE MULTEXT-EAST AND MULTEXT LEXICONS

We have applied LemmaGen on two sets of lexicons, namely Multext-East [4] and Multext [5] (Multilingual Text Tools and Corpora) to automatically learn lemmatizers for different languages. There are altogether 14 lexicons for 12

	Language	No. of records	No. of different				
			Morph.forms	Lemmas	Morph.specs	Morph.forms per lemma	Lemmas per morph.form
MULTEXT- EAST	Slovenian	557.970	198.507	16.389	2.083	12,63	1,0430
	Serbian	20.294	16.907	8.392	906	2,07	1,0285
	Bulgarian	55.200	40.910	22.982	338	1,95	1,1002
	Czech	184.628	57.391	23.435	1.428	2,55	1,0441
	English	71.784	48.460	27.467	135	1,80	1,0206
	Estonski	135.094	89.591	46.933	643	2,19	1,1507
	French	306.795	232.079	29.446	380	8,01	1,0164
	Hungarian	64.042	51.095	28.090	619	2,03	1,1209
	Romanian	428.194	352.279	39.359	616	9,35	1,0447
MULTEXT	English	66.216	43.371	22.874	133	1,93	1,0182
	French	306.795	232.079	29.446	380	8,01	1,0164
	German	233.858	51.010	10.655	227	4,87	1,0174
	Italian	145.530	115.614	8.877	247	13,85	1,0636

	Spanish	510.709	474.158	13.236	264	36,07	1,0069

Figure 2: *Sizes and basic properties of the MULTEXT-EAST and MULTEXT training sets.*

East and West European languages (see Figure 2). Each lexicon contains records of the form *(word-form, lemma, morphological form)*. The last column (morph. form) was not used in our experiments, but nevertheless it indicates the complexity of different languages. One can speculate that the higher number of morphological forms per lemma indicates a more complex language. On the other hand, a higher fraction of lemmas per morphological form (e.g., Bulgarian, Estonian, Hungarian) will probably prove to be more difficult for learning and will result in lower accuracies. 'Simpler' languages with lower number of lemmas per morphological form (e.g., Spanish, German, French, English) will likely have better lemmatizers with higher accuracy. The available number of training examples and how representative the training examples are will also affect the accuracy (e.g., there are relatively few training examples for Serbian).

For learning and testing experiments we used 5-fold cross validation. For each language, cross validation was performed 10 times. Both, the original RDR algorithm and our improved LemmaGen were applied. Results are given in Figure 3:

- Accuracy – Lemmatization assigns a transformation (class) to a word-form. If there are *P* correctly lemmatized word-forms, and *N* is the total number of word-forms, then *Acc = P/N*.
- Accuracy was tested on the training set (yielding an optimistic accuracy prediction), testing set ('realistic' prediction) and on unknown words from the testing set (pessimistic prediction). In the last case we made sure that no two words with the same lemma appear in both, training and testing set in the same validation step.
- Standard deviation is averaged over all three sets above. Lower values indicate higher stability of the learning algorithm.
- Error is a relative decrease of the number of incorrectly classified examples of LemmaGen relative to the original RDR. *Error = (Acc(RDR) - Acc(LemmaGen)) / (1 – Acc(RDR))*. An Error of -25 means that LemmaGen commits 25% less incorrect classifications then RDR.

The results indicate that LemmaGen outperformed the original RDR in most of the cases, primarily due to the improvements described in Section 2.

The (reverse) lexicographical ordering of examples and subsequent use of example frequencies results in the highest improvement of accuracy on the training set. This might seem irrelevant since generally we are mostly concerned with the accuracy on new, unknown examples. However, in the case of lemmatization and lexicons provided, it turns out that they mostly cover a typical text corpora. Therefore, training examples cover most of the domain, and accuracy on the training set is very relevant for practical applications of lemmatizers.

We did test this hypothesis on a Slovene corpus of news agencies texts which comprises almost 900.000 words [6]. It turned out that 84% of the words were covered by the lexicon used for learning the lemmatizer. Therefore, the expected accuracy is best computed by using the accuracy on the training set in 84%, and accuracy on the unknown words in 16% of the cases. If *p* is the fraction of words covered by the learning lexicon then a realistic estimate of the expected accuracy is: *Acc = p*Acc(optimistic) + (1-p)*Acc(pessimistic)*. In the case of Slovenian, we get: *Acc = 84%*97.61% + 16%*82.12% = 95.13%*. This is slightly above the actual accuracy on the testing set.

	Language	Accuracy (%)									Standard deviation (%)		
		Learning set (optimistic)			Test set (realistic)			Unkown words (pessimistic)					
		RDR	LemmaGen	Errors	RDR	LemmaGen	Errors	RDR	LemmaGen	Errors	RDR	LemmaGen	Errors
MULTEXT- EAST	Slovenian	95,35	97,61	-48,6	92,59	94,38	-24,1	80,68	82,12	-7,5	0,029	0,015	-47,88
	Serbian	94,36	97,86	-62,1	70,34	73,49	-10,6	64,26	65,85	-4,5	0,150	0,059	-60,44
	Bulgarian	91,22	93,68	-28,0	74,52	76,10	-6,2	69,29	71,52	-7,2	0,107	0,074	-30,29
	Czech	96,61	97,89	-37,8	92,77	93,66	-12,3	78,09	81,13	-13,9	0,040	0,023	-41,02
	English	97,75	98,84	-48,3	92,05	93,07	-12,8	89,27	91,03	-16,4	0,038	0,021	-45,27
	Estonian	86,81	89,51	-20,5	73,52	73,93	-1,6	66,69	66,54	0,5	0,066	0,049	-25,83
	French	96,72	98,80	-63,5	91,78	92,94	-14,1	86,80	88,22	-10,8	0,032	0,015	-54,19
	Hungarian	90,23	91,88	-16,9	74,82	74,33	2,0	72,73	72,86	-0,5	0,091	0,072	-21,03
	Romanian	94,96	96,75	-35,6	78,16	79,17	-4,6	73,48	74,14	-2,5	0,036	0,033	-7,27
MULTEXT	English	98,20	99,00	-44,5	93,29	94,14	-12,7	90,82	92,48	-18,1	0,052	0,029	-45,17
	French	96,72	98,80	-63,5	91,79	92,95	-14,2	86,85	88,25	-10,7	0,034	0,012	-63,71
	German	95,88	98,70	-68,5	95,06	97,13	-41,9	79,56	84,15	-22,4	0,062	0,026	-58,54
	Italian	93,75	95,58	-29,2	85,87	86,08	-1,5	82,05	82,11	-0,3	0,041	0,040	-3,26
	Spanish	99,10	99,48	-42,1	94,65	95,73	-20,1	94,32	95,45	-19,9	0,007	0,008	7,42

Figure 3: *Comparison of accuracy between the original RDR lemmatizer and the improved LemmaGen.*

Results in Figure 3 also enable the analysis of different languages. The actual accuracies are mostly as expected, except for Hungarian and Estonian. It turns out that the two languages are not Indo-European, but belong to the Finno-Ugric language group (along with Finnish). In these languages words can be composed from morphemes in a large number of ways. Consequently, lemmatization by suffix transformation only appears to be of limited value and a more expressive transformation language is needed. Figure 4 gives the efficiency comparison.

	Language	Learning					Lemmatization				
		RDR		LemmaGen		Speedup factor	RDR		LemmaGen		Speedup factor
		sec	ms/rec	sec	ms/rec		sec	ns/rec	sec	ns/rec	
MULTEXT- EAST	Slovenian	26,80	60,0	3,02	6,8	8,9	2,53	22.633	0,10	867	26,1
	Serbian	0,23	14,4	0,09	5,4	2,7	0,03	8.089	0,00	643	12,6
	Bulgarian	2,03	46,0	0,32	7,2	6,4	0,30	26.958	0,01	670	40,2
	Czech	4,42	29,9	0,56	3,8	7,9	0,42	11.279	0,03	722	15,6
	English	0,43	7,5	0,23	4,0	1,9	0,10	6.946	0,01	752	9,2
	Estonian	4,15	38,4	0,68	6,3	6,1	0,41	15.226	0,02	800	19,0
	French	6,46	26,3	1,72	7,0	3,7	1,35	21.995	0,06	898	24,5
	Hungarian	0,99	19,4	0,23	4,5	4,3	0,12	9.575	0,01	718	13,3
	Romanian	183,12	534,6	7,23	21,1	25,3	43,51	508.043	0,08	911	557,7
MULTEXT	English	0,37	6,9	0,21	3,9	1,8	0,08	6.017	0,01	724	8,3
	French	7,02	28,6	1,56	6,3	4,5	1,34	21.819	0,05	877	24,9
	German	10,22	54,6	0,80	4,3	12,9	0,60	12.857	0,04	788	16,3
	Italian	1,18	10,1	0,80	6,9	1,5	0,26	8.821	0,03	860	10,3
	Spanish	22,97	56,2	3,88	9,5	5,9	3,57	34.923	0,09	894	39,1

Figure 4: *Comparison of the learning and lemmatization efficiency between the original RDR and LemmaGen.*

6 CONCLUSION

We have developed an improved learning algorithm for automatic generation of lemmatization rules in the form of a RDR tree, named LemmaGen. The algorithm has linear time complexity, is very efficient, and can produce very accurate lemmatizers from sufficiently large lexicons. The whole LemmaGen system is freely available under the GNU open source license from http://kt.ijs.si/software/LemmaGen .

References

[1] Beth, L.J. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistic* 11, pp. 22–31, 1968.

[2] Compton, P., Jansen, R. Knowledge in Context: a strategy for expert system maintenance. *Proc. 2nd Australian Joint Artificial Intelligence Conference,* pp. 292–306, 1988.

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

[4] Erjavec, T. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. *Proc. 4th International Conference on Language Resources and Evaluation LREC-2004,* pp. 1535-1538, 2004.

[5] Ide, N., Véronis, J. MULTEXT: Multilingual Text Tools and Corpora. *Proc. 15th Conference on Computational Linguistics* 1, pp. 588-592, 1994.

[6] Juršič, M. Efficient Implementation of a system for Construction, Application and Evaluation of RDR Type Lemmatizers. Diploma Thesis, Faculty of Computer and Information Science , University of Ljubljana, 2007.

[7] Mladenić, D. Automatic Word Lemmatization. *Proc. 5th International Multi-Conference Information Society IS-2002* B, pp.153-159, 2002.

[8] Mladenić, D. Combinatorial Optimization in Inductive Concept Learning. *Proc. 10th International Conference on Machine Learning ICML-1993, pp.* 205-211, 1993.

[9] Mladenić, D. Learning Word Noramlization Using Word Suffix and Context from Unlabeled Data. *Proc. 19th International Conference on Machine Learning ICML-2002, pp.* 427-434, 2002.

[10] Plisson, J., Lavrač, N., Mladenić, D. A rule based approach to word lemmatization. *Proc. 7th International Multi-Conference Information Society IS-2004* C, pp. 83-86, 2004.

[11] Plisson, J., Lavrač, N., Mladenić, D., Erjavec, T. Ripple Down Rule Learning for Automated Word Lemmatisation. *AI Comm.,* in press, 2007.

[12] Porter, M.F. An Algorithm for Suffix Stripping. *Program* 14(3), str. 130−137, 1980.