Ripple Down Rule learning for automated word lemmatisation

Ripple Down Rule Learning for Automated Word Lemmatisation

Joël Plisson [a], Nada Lavrač [a,b],
Dunja Mladenić [a], Tomaž Erjavec [a]
[a] *Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
E-mail: joel.plisson@ijs.si
[b] *University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia*

Lemmatisation is the process of finding the normalised forms of wordforms as they appear in text. It is a useful pre-processing step for a large number of language engineering tasks, and especially important for languages with rich inflection morphology. This paper presents a machine learning approach to automated word lemmatisation using a Ripple Down Rule learning algorithm, specially adapted to this task. By focusing on word suffixes, the induced Ripple Down Rules determine which wordform suffix should be removed and/or added to generate the lemma. The rules, induced from a lexicon of lemmatised Slovene words, were evaluated by cross-validation in the lexicon and on a hand-validated annotated corpus, and compared to previous work using two other inductive lemmatisers, ATRIS and CLOG. We show that RDR outperforms ATRIS and is more flexible than CLOG, as it can, unlike CLOG, also work without prior part-of-speech tagging. The RDR lemmatiser is easy to train and use for new languages and is, together with CLOG, available via a Web service.

Keywords: Ripple Down Rules, lemmatisation, machine learning, Slovene language

1. Introduction

Lemmatisation is the process of finding normalised forms of words, called lemmas, which correspond to headwords in a dictionary. For example, the lemma of the wordform *dogs* is *dog*, of *wolves* is *wolf*, of *sheep* is *sheep*, and of *living* is *live*. Lemmatisation thus abstracts away from the inflectional variations of words and is, as such, an important pre-processing step on texts. On the one hand, it reduces the representation space for applications of text mining and human language technologies; on the other, it helps people search texts and corpora with words in their standard form.

The paper presents the problem of lemmatisation of words from Slovene free text and explains why the RDR learning approach is appropriate for the task. We show that by learning from a lexicon of lemmatised Slovene words, the approach results in high precision easy to understand RDR rules.

Section 2 puts the problem of lemmatisation in the wider context of natural language processing and machine learning. Section 3 reviews related work on machine learning of lemmatisation, and Section 4 explains why Ripple Down Rules are appropriate for solving this problem, followed by the presentation of the proposed approach in Section 5. Sections 6 and 7 report on tests on our approach in two experiments and compare the results to those achieved by two other lemmatisation learners, ATRIS and CLOG. Section 8 introduces the web interface which allows on-line lemmatisation of Slovene texts. The paper concludes by presenting plans for further work.

2. Lemmatisation in context

We first introduce the notion of the *wordform*, which is the (inflected) form of the word as it appears in running text, e.g., *wolves* in the sentence *The sheep was eaten by wolves.* This wordform can be morphologically analyzed into its stem *wolf-* and ending *-s*. As evident from the example, phonological and morphological factors can influence how the abstract stem and ending are combined to arrive at the wordform. These factors are especially complex in languages with heavy inflection, such as Slovene and other Slavic languages, where stems can combine with many different endings, in a many-to-many relation, and the selection of the appropriate ending for a given stem and how they combine into the wordform can depend on a whole range of factors, from phonological to semantic.

In order to abstract away from the variability of wordforms, two methods are used. The first, so called *stemming* is popular in Information Extraction and Retrieval, and, essentially, tries to reduce the wordform to an invariant stem that semantically identifies it; this method often collapses different word-classes (parts-of-speech) and does not, in general, produce a surface form. So, for example, the wordforms *computes, computing, computed* should be most likely stemmed to *comput*. The second method is *lemmatisation*, which aims at transforming a wordform to get its canonical form, the lemma, where the canonical form is one particular wordform from an inflectional paradigm that, by convention, serves to identify an abstract word. This distinction is not so important in English, where the stem is typically identical to the lemma, but is much more obvious in Slovene; for example, the wordform *ovce* (genitive of *sheep*) has as its stem *ovc-*, while the lemma form, by convention the nominative singular, is *ovca*. As opposed to stemming, lemmatisation is more selective (a single stem can have more than one lemma, e.g., verbal and nominal) and results in an intuitively understandable form of the word. It is also more difficult: not only does the word ending need to be removed from the stem, but the correct ending corresponding to the lemma form needs to be added.

Lemmatisation is also faced with the problem of ambiguity: a wordform, and especially that of an unknown word can have multiple possible lemmas. So, for example, the wordform *hotela* can be lemmatised as *hotel* (the noun *hotel*), or *hoteti* (the verb *to want*). Which is the correct lemma depends on the context that the wordform appears in. The task of morphosyntactic disambiguation (i.e., determining if a certain wordform in the text is a noun or a verb, and, typically, also its other inflectional properties) is the domain of part-of-speech taggers, or, more accurately, morphosyntactic taggers. By using the information provided by such a tagger a lemmatiser is in a much better position to correctly predict the lemma form. However, as will be seen, even without such information, a lemmatiser can still function, although with a lower precision.

Rules in grammars of natural languages typically obey the Elsewhere condition [12], which states that in cases where more than one rule is applicable, the most specific should apply. In other words, rules for, say, lemmatisation need to be ordered, with exceptions coming first, followed by more general rules.

Traditionally, such rules, together with extensive morphological lexica incorporating inflectional information (e.g., that a particular stem should be inflected according to a certain paradigm) were hand-coded, and various methods (typically finite state automata or transducers) were used to compress the resulting language model. High-coverage, precise and fast morphological analysers (which, as a side-effect, could also produce lemmas of wordforms) have been developed for a number of languages, to the extent that lemmatisation was often taken as a solved problem. However, such systems, in common with other hand-crafted approaches, have several shortcomings:

– They do not do well on out-of-vocabulary words: information associated with a lemma is crucial for predicting which wordforms should be lemmatised to it.
– They are expensive to construct: there are still languages without such infrastructure, and developing a large morphological lexicon and rule set is a long-term and labour intensive undertaking, and only experts can do it.
– They are difficult to adapt: while such systems do well on standard modern day languages for which they were developed, they degrade considerably when faced with language varieties, such as historical language, informal language of e-mails, etc.
– They are difficult to acquire and install: the lexicon and rules work with dedicated software, which is often not available to others, or only for a price, and typically imposes requirements on the client platform.

These are mostly the reasons why machine learning approaches to morphological analysis and lemmatisation became, and still are, the subject of research: they are robust and can handle out-of-vocabulary words; they can be (re)trained on small amounts of data, and these datasets are easier to construct than complicated rule sets; and it is easier and cheaper to acquire training datasets than software. For example, a researcher in multilingual language processing needing the functionality of lemmatisation can acquire lexica for many languages via the European Language Resources Association ELRA or the U.S. based Linguistic Data Consortium LDC, train the learner on them, and use the resulting lemmatisers. It would be much more difficult to acquire, install and integrate the equivalent number of proprietary lemmatisers. Of course, if no training resources exist for a language, then the development of such resources is still a large undertaking, which could timewise be as demanding as the manual development of rules. However, the task is typically much simpler and can be performed even by non-linguists.

3. Related work

A lot of work has been carried out in word lemmatisation and the related task of stemming, particularly for English. Traditionally this involved hand-crafted rules, as is e.g., the case with the well-known Porter stemmer [20]. On the other hand, automatic lemmatization was, among other tasks, addressed in [25], where the main idea is to use existing text analysis tools for English, apply them to bilingual text corpora and project their output onto the second language. This approach does not require any hand-annotated training data in the given language; it was tested on French, Chinese, Czech and Spanish. A data-driven context-sensitive approach to lemmatization in running text was proposed in [2]. The approach is based on computing the so called shortest edit script between the wordform and its lemma strings. It was tested on a range of typologically different natural languages, for example Polish (as Slovene, a Slavic, hence highly inflecting language), where it achieves an 81.7% precision (at 77.5% recall) in lemmatising unknown words. Another approach [19], interesting as it deals with Bulgarian (also a Slavic language, although significantly less inflecting), simply orders suffix removal rules by frequency in the training set, to arrive at a stemmer; they report an accuracy of 78.8% over a lexicon.

3.1. Machine learning on related tasks

One of the early relational learning approaches is FOIDL [17], which induces first-order decision lists using Inductive Logic Programming. Relational learning of decision lists was applied to the problem of learning English past tense [18]. Other approaches [13] have used decision trees and neural networks in order to predict separate letters of the transformed word based on the letters in the past tense form of the word. There are several machine learning approaches addressing related tasks of learning the morphological structure. For instance, Van den Bosch and Daelemans [24] propose using memory based classification to perform morphological analysis of Dutch words. Another related approach to a more general task of using machine learning to perform morphological analysis is proposed in [23], where words are mapped to trees representing their morphological structure. A memory based approach was proposed in [3] to handle English past tense, Arabic broken plurals as well as German and one particular form of Slovene nouns.

3.2. Machine learning for lemmatisation

In this paper we use the RDR learning approach to the lemmatization of arbitrary text, and evaluate it on the Slovene language. Closely related work on lemmatisation includes two approaches, the first using co-training with if-then rules and Naive Bayes, and the second using first-order decision lists applicable to some types of words. The first approach, proposed in [16], uses co-training combining if-then rules constructed by the ATRIS rule learner [15] and Naive Bayes for text classification. In our comparisons we consider only the part of this approach using if-then rules. The second approach uses first-order decision list learner CLOG [14], and is described and evaluated in [9,6,10] and relies on having information from a part-of-speech tagger; this allows it to attain a higher accuracy, but note that such a tagger is not, in general, available for all languages.

4. The proposed RDR learning approach

The approach to word lemmatisation presented in this paper relies on a Ripple Down Rule (RDR) learning algorithm [5,22] which has been adapted to the task of lemmatisation. Similarly to FOIDL, CLOG and ATRIS, the training set for the learner is a lexicon of pairs `(wordform,lemma)`. By focusing on invariant portions of the word, the induced RDR rules determine which word suffix should be removed and/or added to get the lemma of a wordform. It should be noted that the invariant portion of the word can be empty, which caters for cases of total suppletion, such as *is/be*; the suffix to be removed simply spans the complete word.

Once the lemmatisation rules have been learned, the rules can be used as a lemmatiser for known words (the words used by the learner for training), as well as unknown words (out-of-vocabulary words, not used in training the lemmatiser). The rules will typically be applied to word tokens in free text, after the text has been transformed into a list of word tokens by a tokeniser, and these, possibly, tagged by a part-of-speech tagger. The algorithm is not biased to Slovene; given an appropriate training set it can be applied without modifications to other inflectional languages.

4.1. Format of the training set

In our approach to learning Ripple Down Rules (RDR), a training set for word lemmatisation consists of wordforms, described by their suffixes (the attributes) and the transformation (the class) to be applied to the wordform suffix in order to get the lemmma. Examples of Slovene words with their attribute representation, and transformations are presented in Table 1.

Attributes. In standard machine learning settings, the training examples are described by a fixed set of attributes. E.g., in the learning setting of ATRIS used in previous work, the suffix length was limited to five attributes regardless of the word length. On the other hand, in the RDR learning setting, as the wordforms have different lengths, they are described by a variable number of attributes $A_i \in [A_1, \dots, A_n]$ which correspond to word suffixes of i letters, where the number of attributes depends on the word length n. In Table 1, attribute A_1 corresponds to the last character, attribute A_2 to the last two characters, etc.

Classes. As proposed in the ATRIS lemmatization learner [16], each wordform in the training set is labelled by a class label that represents the transformation that should be applied to get the lemma of the wordform. To determine the class, the 'stem' of the wordform should be found first. Notice that this is not necessary the same stem as the morphological stem or that one obtained by applying a stemmer to the wordform. Here, by a stem we simply mean the invariant part of the word, i.e., the part that the wordform and

Table 1

Training examples with class labels corresponding to suffix transformations.

	A_1	A_2	A_3	...	Classes
Examples	LastCh	2LastCh	3LastCh	...	s1 $\rightarrow$ s2
`breskev`	`v`	`ev`	`kev`	...	`_->_`
`breskvah`	`h`	`ah`	`vah`	...	`vah->ev`
`breskvam`	`m`	`am`	`vam`	...	`vam->ev`
`breskvama`	`a`	`ma`	`ama`	...	`vama->ev`
`breskvami`	`i`	`mi`	`ami`	...	`vami->ev`
`verzija`	`a`	`ja`	`ija`	...	`_->_`
`verzij`	`j`	`ij`	`zij`	...	`_->a`
`verzijah`	`h`	`ah`	`jah`	...	`h->_`
...	...	...	...	...	...

lemma have in common. The wordforms *property* and *properties* have both the stem *propert* in common. Or in Slovene, the wordforms *breskev* and *breskvah* have *bresk* in common; in lemmatisation we should remove the suffix *vah* from *breskvah* and add the suffix *ev* to get the lemma *breskev.* Thus, the transformation (the class) in the form `s1->s2` corresponds to a mapping $s1 \rightarrow s2$, where $s1$ is the suffix of the wordform and $s2$ is the suffix of the lemma; e.g., in Table 1 class label `vah->ev` is assigned to the wordform *breskvah*, while class label `_->_` means that the wordform does not change.

4.2. The Ripple Down Rules representation

The Ripple Down Rules (RDR) [5,22] were initially used for knowledge acquisition and maintenance of rule-based systems. In comparison with the standard if-then classification rules Ripple Down Rules (RDR) resemble decision lists [21] of the form if-then-else: new RDR rules are added by creating **except** or **else** branches to the existing rules. Take a simple Ripple Down Rule for verb lemmatisation:

```
if 2LastCh = ed then class = ed->_
  except if 3LastCh = ied then class = ied->y
```

The rule states that one should remove the suffix `ed` and add an empty suffix to get the lemma unless there is an `i` before `ed`, in which case we should remove the suffix `ied` and add the suffix `y`. This rule would, for instance, correctly generate the lemma of wordforms `walked` (`walk`) and `classified` (`classify`).

```
if m then m->ti because of zadolzim
  except
    if am then m->_ because of verzijam
      except
        if sam then m->ti because of raznasam
        else if jam then m->_ because of stojam
        else if cam then m->_ because of dvojicam
        else if kam then m->_ because of strankam
      end except
    else if om then om->_ because of teroristom
      except
        if kom then kom->ek because of izdelkom
      end except
    else if im then im->_ because of zagotovljenim
      except
        if tnim then nim->en because of prisotnim
      end except
    else if em then em->_ because of premesanem
      except
        if tnem then nem->en because of prisotnem
        else if jem then jem->eti because of zamrjem
      end except
  end except
```

Fig. 1. Induced RDRs for words ending with letter `m`. Notice that in a RDR the ordering of examples (shown one by one to the RDR learner) is important, thus the same class can repeat several times, e.g., `m->ti` in this example rules.

An additional feature of RDR rules, compared to decision lists, is that exceptions that have created an **except** or **else** branch are added to the branch with a **because of** keyword, in order to explain the reason for the creation of the new rule. This feature of Ripple Down Rules turns out to be extremely helpful to achieve a better understanding of the complex rules. E.g., if the above listed rule were induced from training examples `walked` and `classified`, the rule is as follows:

```
if ed then ed->_ because of walked
  except if ied then ied->y because of classified
```

As can be seen in Figure 1 (in which RDRs are simplified so that attribute names and class are omitted), the **because of** construct of Ripple Down Rules has a good explanatory potential, especially in the case of complex rules. Explicit listing of exceptions (enclosed in an *except* block) enables the interpretation of induced RDR rules as grammatical rules, potentially interesting for linguists, who can check the induced rules for grammatical consistency and completeness with respect to the grammar of the language for which the rules have been induced.

4.3. Using the RDR set for lemmatisation

The result of rule induction is an RDR set that can be used to lemmatise words. We apply the rules on each wordform to be lemmatised and if a rule fires, its conclusion (class) is the transformation to apply to get the lemma of the wordform. This means that for using the rules on free text, one must first convert it with a tokeniser into a list of words and present them one by one to the induced RDR rule-set.

5. Learning Ripple Down Rules for word lemmatisation

As opposed to standard if-then classification rules, induced using the covering algorithm for rule set construction, such as the well-known CN2 algorithm [4] and the ATRIS algorithm [15] (used previously for Slovene word lemmatisation), Ripple Down Rules are learned by creating exceptions to existing rules. If a rule fires but produces an incorrect conclusion then an **except** branch of the rule is created. If no rule fires then an **else** branch of the rule is created. Consequently, rule changes are confined to the context of the rule and will not affect other rules.

Our approach adapts a standard RDR algorithm by incorporating numerous features, specific to the problem of word lemmatisation. The result is a set of rules that can be interpreted as a decision tree. The depth of the tree depends on the complexity of differentiating between the different wordforms in the training set.

5.1. Rule set construction algorithm

In creating Ripple Down Rules all the examples are presented, one by one, to the following RDR induction algorithm:

```
if rule fires then
    if correct conclusion then
        Do not add an exception.
    else if incorrect conclusion then
        Find differences with the example that
        induced the rule.
        Create an exception to the rule that fired.
else
    create a new rule with an else if branch
```

To create an exception to a rule, the algorithm first recovers the wordform that induced the rule that fired. Then the differences between the two wordforms are calculated. The conditions of the exception rule correspond to these differences. Suppose that the algorithm has by now constructed the following rule:

if `a` **then** `_->_` **because of** `mina`
else if `vah` **then** `vah->ev` **because of** `breskvah`

which would result in lemmatised forms `mina` and `breskev`, and that new wordform `breskvama` is presented to the algorithm that needs to be lemmatised into `breskev` as well, then a new rule (exception) will be added:

if `a` **then** `_->_` **because of** `mina`
  except if `vama` **then** `vama->ev` **because of** `breskvama`
else if `vah` **then** `vah->ev` **because of** `breskvah`

In the above rule, `breskvama` needs to be distinguished from `mina`. In general, the system tries to form the rule condition using the shortest suffix distinguishing the wordform from the general case (e.g., suffix `ma` for the wordform `breskvama` to distinguish it from `mina`). However, the suffix needs to be at least as long as the suffix of the class determining the transformation (e.g., `ma` is insufficient, the only alternatives considered are `vama`, `kvama`, ... as the class is `vama->ev`).

5.2. Improved RDR construction algorithm

In initial studies of RDR [5], only a single case was stored with each rule. The rule had to distinguish between the new case and the case associated with the rule that gave the wrong conclusion. But with this original algorithm, some cases which were first correctly classified became wrong after adding an exception which erroneously covered the previous cases. E.g., if the training data contained words `breskev`, `postavitev` and `zahtev` (with lemmas `breskev`, `postavitev` and `zahteva`) the system would fail to predict the correct class for `postavitev` given the following rule:

if `v` **then** `_->_` **because of** `breskev`
  except if `tev` **then** `_->a` **because of** `zahtev`

During the training, `breskev` has created the first rule, `postavitev` has not created a new rule because the first rule fired with the correct conclusion and `zahtev` has created an exception to the first rule. But `postavitev` is now also covered by this exception as it has the suffix `tev`. Therefore if `postavitev` occurs as a test case, the classifier returns the wrong class (`_->a` instead of `_->_`).

A solution to this problem is to keep all the wordforms covered by the rule as in [11]. Therefore, for each new rule, a list of all covered cases is kept. So that when an exception is added, the conditions are chosen so that they exclude the cases covered by previous rules and exceptions. In our example we now have to use `htev` as a condition instead of `tev` in order to differentiate the examples covered by the new rule (`zahtev`) from the examples covered by the previous rule (`breskev`, `postavitev`).

The improved RDR learner will generate the following example rule, assigning the correct class for the word `postavitev`.

if `v` **then** `_->_` **because of** [`breskev`, `postavitev`]
  except if `htev` **then** `_->a` **because of** [`zahtev`]

We have also tried another way of improving rule construction through feeding the constructed set of rules (together with the original examples) back to the system for revision. This improved the performance of the system but not as much as keeping all the words covered by the rule; therefore, these results are not reported in the paper.

6. Experiments on samples of the MULTEXT-East lexicon

In this section we first introduce the lexicon which was used for training the RDRs and, for RDRs learned from samples with different set sizes, evaluate their performance with cross-validation. We also compare the RDR results to the results achieved in previous work [16].

6.1. The Slovene MULTEXT-East lexicon

MULTEXT-East [7] are freely available multilingual language resources for language engineering research, focusing on the morphosyntactic level of language analysis: they contain morphosyntactic specifications, defining the features that describe word-level syntactic annotations; medium scale morphosyntactic lexica; a small parallel corpus (Orwell's novel "1984", cca 100,000 words),

`naženimo`	`nagnati`	`Vmmp1p`
`naženita`	`nagnati`	`Vmmp2d`
`naženite`	`nagnati`	`Vmmp2p`
`naženiva`	`nagnati`	`Vmmp1d`
`ne`	`ne`	`Ccs`
`ne`	`ne`	`Q`
`neambiciozen`	`neambiciozen`	`Afpmsnn`
`neambiciozen`	`neambiciozen`	`Afpmsan-n`
`neambiciozna`	`neambiciozen`	`Afpfsn`
`neambiciozna`	`neambiciozen`	`Afpmda`

Fig. 2. The MULTEXT-East lexicon format.

hand annotated with morphosyntactic descriptions and lemmas, and a comparable corpus ($2 \times 100,000$ words; newspapers and a novel).

The MULTEXT-East morphosyntactic lexicons have a simple structure, where each lexical entry is composed of three fields:

1. the *wordform*, which is the inflected form of the word, as it appears in the text, except for sentence-initial capitalisation;
2. the *lemma*, i.e., the base-form of the word; and
3. the *MSD*, i.e., the morphosyntactic description, a feature-structure giving the part-of-speech and other morphosyntactic attributes of the wordform.

Figure 2 illustrates the format of the lexicon, by giving the last part of the paradigm for Slovene verb *nagnati* (*to dismiss somebody*), the function word *ne* (*no*), and the start of the paradigm for the adjective *neambiciozen* (*unambitious*). The MSDs are represented as compact strings, with positionally coded attribute values. So, for example, the MSD `Afpmsan-n` expands to Category = Adjective, Type = qualificative, Degree = positive, Gender = masculine, Number = singular, Case = accusative, Definiteness = not appropriate for combination, Animacy = no.

The Slovene lexicon, the basis for our experiments, contains the complete inflectional paradigms of the lemmas present in the 300,000 word corpus, i.e., 557,970 entries with 198,507 different wordforms, 16,510 different lemmas, and 2,083 different MSDs.

6.2. Experimental setting

We have performed two different experiments aimed at providing a comparison with if-then

Table 2

Classification accuracy computed as the average in 5-fold cross-validation on 5720 examples.

Learner	Original data	Sequential modeling
ATRIS	62.6% ±0.07	72.8% ±0.7
RDR	77.0% ±0.6	77.0% ±0.6

rules induced by the ATRIS rule induction algorithm [15,16]. We took the same data as used in [16], namely five datasets obtained by random sampling of different sizes (160, 920, 1890, 3820, 5720) taken from the Slovene MULTEXT-East lexicon. These datasets include only the wordform and lemma from the lexicon, i.e., they make no use of the morphosyntactic information associated with each entry.

For each of the five datasets we measured the classification accuracy achieved in 5-fold cross-validation.

6.3. Experiments on raw data samples

For this experiment, the training data was not modified: the samples were presented to the algorithm in a random order. We can see (Figure 3) that the variation in the performance is small, after having included more than 1890 examples into the training set. For the same maximal number of 5,720 samples presented, on average RDR achieved 77.0% accuracy, while ATRIS achieved 62.6% (see Table 2). We can attribute this improvement to the fact that RDRs are executed by first testing of exceptions (more specific rules) followed by more general rules. The results indicate that RDR can cover more examples with less training and get better accuracy with fewer examples presented. The improvement may be also attributed to the construction of RDR keeping all the cases covered by the rules in order to build reliable exceptions.

6.4. Experiments on data pre-processed with sequential modelling

Results of previous work indicate that the best accuracy of 72.8% was achieved with ATRIS when using sequential modeling, described in [16]. In the ATRIS sequential modeling approach, the training data was pre-processed by ordering all the examples (words) alphabetically according to their ending letter (words ending with A first, then the words ending with B, etc.). Thus, all the words

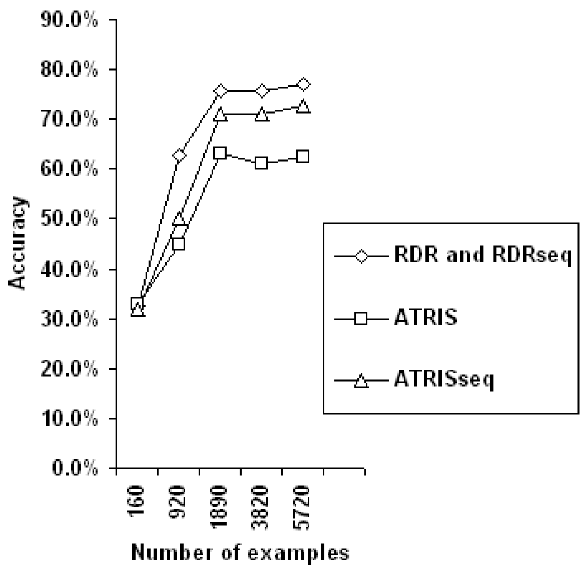

Fig. 3. Results of 5-fold cross-validation, for each of the five training sets.

ending with the same letter are grouped together. As mentioned in [16], the effect is that instead of dealing with all the training examples with 156 different class values in the MULTEXT-East dataset, the problem is decomposed into independent subproblems, one for each ending letter, each having between 1 and 40 class values (most of the subproblems have less than 10 class values). Using this data pre-processing, the accuracy of ATRIS had significantly improved (72.8%) while RDR learning resulted in exactly the same accuracy (77.0%) as without data pre-processing (see Table 2 and Figure 3). The reason is that RDR rules are induced in parallel for different conditions (suffixes) which obviates the need for this kind of pre-processing.

7. Experiments on the SVEZ-IJS corpus sample

This section presents the results of the experiment in which RDRs were learned from the entire Slovene MULTEXT-East lexicon, and tested on a sample from the SVEZ-IJS corpus [8,10], with the intention of validating the lemmatiser as close as possible to a real-life scenario, i.e. training on all available data and measuring its accuracy on word tokens in free text.

```
<ab n="3452">
  <seg lang="en">
    <w ana="Dd" lemma="all">All</w>
    <w ana="Ncnp" lemma="harbour">harbours</w>
    <c>,</c>
    <w ana="Ncnp" lemma="airport">airports</w>
    <w ana="Cc-n" lemma="and">and</w>
    <w ana="Ncns" lemma="border">border</w>
    <w ana="Ncnp" lemma="station">stations</w>
  </seg>
  <seg lang="sl">
    <w ana="Pg-npn----a" lemma="ves">Vsa</w>
    <w ana="Ncnpn" lemma="pristanišče">pristanišča</w>
    <c>,</c>
    <w ana="Ncnpn" lemma="letališče">letališča</w>
    <w ana="Ccs" lemma="in">in</w>
    <w ana="Afpmpn" lemma="mejen">mejni</w>
    <w ana="Ncmpn" lemma="prehod">prehodi</w>
  </seg>
</ab>
```

Fig. 4. One translation unit from the SVEZ-IJS corpus.

We first introduce the corpus testing set, and then evaluate the performance of RDR, giving special attention to unknown words. We evaluate RDR in two settings, one using only wordforms as input, and the other utilising also morphosyntactic tags as assigned to wordforms by a tagger. Furthermore, we compare our results to those achieved on the same training and testing dataset by CLOG [10].

7.1. The corpus dataset

The SVEZ-IJS parallel English-Slovene corpus [8] contains EU legal texts, the so called Acquis Communautaire. This corpus, freely available for research, contains 10 million words and was made on the basis of the translation memory produced by the Translation Department at SVEZ (the Office of the Slovene Government for European Affairs). The corpus was compiled from the parallel English-Slovene translation units, where each such unit consists of two segments, one in English and one in Slovene, and typically contains one sentence or a part of a sentence, e.g. an item in a list. Figure 4 gives a (slightly simplified) translation unit from the corpus.

7.2. Experimental setting: The SVEZ-IJS gold-standard used for testing

In previous work, the SVEZ-IJS corpus was automatically MSD tagged by TnT [1] and lemma-

Table 3

Statistics on the SVEZ-IJS sample testing set.

	n	%
Segments	821	
Tokens	15,765	
Punctuation	2,346	
All words	13,419	
"Pseudo" words	5,077	
Dataset words	**8,342**	**100.0%**
Known words	6,126	73.4%
Unknown words	**2,216**	26.6%

tised with CLOG [14,9]; to train the tagging and lemmatisation models, the Slovene lexicon and corpus of the MULTEXT-East project [7] were used. More precisely, the lexicon was split into separate training sets, one per MSD, and the lemmatiser was trained on each one separately. When performing the lemmatisation, the lemmatiser uses the tagger assigned MSD to first select the appropriate model, and only then lemmatises the word.

To evaluate the tagging accuracy on the Slovene portion of SVEZ-IJS, a sample of the corpus was extracted containing 3 consecutive segments out of every 1000 segments, giving 0.3% of the Slovene part of the corpus. The tags and lemmas in the sample were then manually checked and corrected if needed, while preserving the automatically assigned annotations [10].

Table 3 gives the statistics over this dataset and the subset that we used for testing the RDR performance. As shown in the table, the tokens are divided into punctuation marks and words, and the words into "pseudo words", which are not appropriate for lemmatisation, and those that are, and so constitute our test dataset. We further divide the dataset into wordforms that occur in the MULTEXT-East Slovene lexicon, i.e., known wordforms which were in the training set of RDR (and CLOG), and those that are unknown to the learner.

The "pseudo words" were determined on the basis of their morphosyntactic descriptions, and contain the kinds of words that need not, or, in some cases, should not be passed to a (Slovene) lemmatiser. In particular:

- Numeric expression: these constitute Arabic and Roman numerals, formulas etc. and are already recognised and flagged by the tokeniser.
- Abbreviations: they do not inflect, and can be identified on the basis of their capitalisation and punctuation, e.g., "NASA", "etc.".
- Foreign words: they do not inflect or have inflections which obey different rules from Slovene. They need to be identified separately and excluded from Slovene lemmatisation.
- Conjunctions, Prepositions, Particles: they form a closed class of words and do not inflect.
- Pronouns: they form a closed class of words and have very irregular inflections, so they are standardly put in a stop-word list or full-form lexicon.

We should stress that the exclusion of pseudo-words from the sample was made in order to arrive at a controlled experimental setting, in which the lemmatiser is evaluated only on open class inflecting Slovene words. End-to-end applications incorporating lemmatisation will, of course, need to deal with identifying and excluding the above listed classes of tokens from (Slovene) lemmatisation.

To conclude, our gold-standard corpus testing set for lemmatisation contains 8,342 Slovene wordform tokens, in particular Nouns, Adjectives, Verbs, Adverbs, and (spelled out) Numerals. Of these 6,126 were already seen by the lemmatiser in the training set, while 2,216, or just over a quarter, are unknown. The unknown words are mostly nouns and adjectives, with a much smaller number of verbs and adverbs. They are typically technical terms and proper names; as these are less frequent words, they are by and large also more inflectionally regular than the high-frequency known words.

The experiments were performed using two training regimes, one using only wordforms as input, and the other utilising also morphosyntactic tags as assigned to wordforms by a tagger.

7.3. Experiment using shallow knowledge

The first experiment assumes that the RDR model has at its disposal only the wordform, and that it was trained on the training set consisting simply of pairs `(wordform,lemma)` which is the minimal requirement for a supervised training regime. This setting is similar to the experiments described in Section 5, but in this experiment the complete MULTEXT-East lexicon of pairs (wordform, lemma) was used as the training set.

Table 4
Accuracies of baseline and shallow RDR.

	Baseline		RDR	
	Errs	Acc	Errs	Acc
All	6,679	19.9%	754	91.0%
Known	5,105	16.7%	485	92.1%
Unknown	1,574	29.0%	269	87.9%

Table 5
Accuracies of TnT, CLOG+TnT, RDR+TnT.

	TnT		CLOG		RDR	
	Errs	Acc	Errs	Acc	Errs	Acc
All	1,112	86.7%	198	97.6%	236	97.2%
Known	549	91.0%	55	99.1%	78	98.7%
Unknown	563	74.6%	143	93.5%	158	92.9%

Table 4 shows the absolute number of errors (wrongly lemmatized words) and the accuracy on the corpus dataset for two lemmatisers, the baseline and RDR, computed on all the words, and divided into known and unknown words. The baseline is a trivial lemmatiser which simply assigns the wordform itself to the lemma. The baseline shows poor results, as it correctly lemmatises only one word in five overall, and one in three for unknown. RDR does significantly better, with performance overall at 91%, and just under 88% for unknown words.

7.4. Experiments using morpho-syntactic descriptions

The second experiment assumes that a part-of-speech tagger (in our case TnT) first assigns a morphosyntactic description to each word, and the lemmatiser can use this additional information. In particular, the lexicon is split into separate training sets, one per MSD, and the lemmatiser is trained on each one separately. When performing the lemmatisation, the lemmatiser uses the tagger assigned MSD to first select the appropriate model, and only then lemmatises the word. This was also the regime used in our previous work with CLOG, so it is possible to directly compare the results achieved by CLOG [10] to those of RDR.

Given that the test set contains hand validated MSDs, the evaluation could have been performed assuming perfect tags, showing errors that result solely in the lemmatiser. However, we prefer a more realistic scenario, so we used the MSDs actually output by the TnT tagger.

Table 5 gives the MSD accuracy of the TnT tagger and the lemmatisation accuracies of CLOG and RDR lemmatisation coupled with TnT output. As can be seen, the lemmatisation accuracies are higher than those of the tagger: the lemmatisers, to a great extent, recover from tagging errors. The reason for this is that many tagging errors do not affect the accuracy of lemmatisation; for example, a very common type of mistake made by the tagger is confusing accusative with genitive, or vice versa, for masculine singular nouns or adjectives. However, these two MSDs always correspond to the same surface form of the word - therefore the lemmatiser will predict the same lemma in either case.

The results of the CLOG lemmatiser, previously reported in [10] show an overall accuracy of 97.6%, which rises to 99.1% on known words. The high score on known words is not really surprising, as the CLOG learner (based on Inductive Logic Programming) always completely covers positive examples, i.e., it achieves 100% accuracy if tested directly against its training set; the 0.9% error rate given in the table is due to mistakes in tagging.

Comparing RDR with CLOG shows that RDR does somewhat worse, but mostly due to known words, as RDR does not necessarily cover the training examples. For unknown words it makes just 15 more mistakes than CLOG, falling by only 0.6% absolute or 10% relative to CLOG. However, the difference is statistically significant: according to McNemar's test, CLOG performs better than RDR, with significance level 0.001 on known words and 0.01 on unknown words.

7.5. Discussion

This section has presented an experiment and contrastive evaluation of RDR lemmatisation on a gold-standard dataset, which is meant to closely mimic conditions when RDR is used in practice: the lemmatiser is used on free text, but is given to lemmatise only those words that can be inflected, and are not function (closed-class) words. Two experiments were performed, which differ in the level of supporting software they require; the first one relies solely of the form of a word in text, making for an extremely knowledge-lean environment; the only resource needed to train the lemmatiser is a list of pairs (wordform, lemma), or a lemmatised corpus. The second experiment, where RDR is coupled with TnT, resulted in a much better accuracy, but presupposes a tagger – and to train the tagger, a MSD hand annotated corpus is needed; of course, the entries in training lexicon must also be triplets (wordform, lemma, MSD). Such training sets are harder and more expensive to come by: while manual lemmatisation is quite intuitive and fast, MSD tagging is a significantly slower process, and the human annotators must have detailed knowledge of the MSDs and how to apply them to the text.

The comparison with CLOG showed that CLOG slightly outperforms RDR; however, it should be noted that RDR can be used both with a tagger or without. CLOG, on the other hand, needs the tagger, as it always covers its training examples. And when the MSDs are removed from the training lexicon, CLOG's rules to always correctly predict its lemmatisations become so specific that they stop covering unknown words, defeating the whole exercise.

8. The lemmatisation Web service

We have developed an on-line lemmatisation service which implements the RDR and CLOG+TnT lemmatisers. Both lemmatisation models have been generated automatically by training the learners from the MULTEXT-East lexicon.

Lemmatisation for Slovene texts is available via the service at *http://nl2.ijs.si/analyze/*, where the text to be lemmatised should be copied into the input window of the interface, the lemmatisation activated, after which the lemmatised text is provided in the output window. The user can choose between using the RDR (learned without TnT tagging) or the TnT+CLOG model. If RDR is selected, the text to lemmatise should be provided in Slovene, while with CLOG the user can choose between Slovene, English, Czech, Estonian, Hungarian, and Romanian.

Several output formats are supported in either tabular or XML format. The user can choose between viewing the list of lemmas, the lemmatised text, or MSD tagged and lemmatised text.

The system imposes no limits on the size of the texts to be processed, except for http timeout.

9. Summary and further work

This paper presented a machine learning approach to automated word lemmatisation using a Ripple Down Rule learning algorithm, specially adapted to this task. It introduces the RDR lemmatiser, evaluates it by cross-validation in a lexicon and on a gold-standard annotated corpus, and compares it to previous results of two other inductive lemmatisers, ATRIS and CLOG. Using only wordforms at input, RDR achieves an unknown-word accuracy of 77.0% on the lexicon and 87.9% on the corpus. If coupled with a part-of-speech tagger, the accuracy on the corpus rises to 92.9% for unknown wordform tokens, and to 97.2% overall. We show that RDR outperforms ATRIS as well as having a simpler training regime, and is more flexible than CLOG, as it can, unlike CLOG, also work without prior part-of-speech tagging. The RDR lemmatiser is simple to train and use for new languages and is freely available via a Web service, which enables remote lemmatisation of Slovene by the RDR lemmatiser, and additional tagging and lemmatisation of some other language texts by TnT + CLOG.

The experiments show that RDR can be successfully used either directly with wordforms, or its performance boosted by using a tagger. While the former setting does have a greater error rate, this might be the only option available for languages for which hand-annotated morphosyntactically annotated corpora are not available; this, to our knowledge, currently includes also European national languages such as Croatian and Slovak.

Our motivation for developing a trainable lemmatiser was two-fold. As opposed to complex morphological lexica and attendant software, a training dataset is much easier to obtain and is more portable. The other advantage of a trainable lemmatiser is that it automatically deals with unknown words, a feature that is often not available with lemmatisers using hand-crafted rules. This is a necessary requirement, as open texts will show out-of-vocabulary words even when using the largest lexica.

In further work we plan to train the RDR lemmatiser on new languages, and extend the on-line Web service to offer lemmatisers for these. We also intend to improve the service by enabling user generation of new language models.

Acknowledgements

We would like to thank the three anonymous reviewers for their comments; all the remaining mistakes are the responsibility of the authors.

This work was supported by the Slovenian Ministry of Higher Education, Science and Technology program Knowledge Technologies, and the European IST projects ALVIS (IST-1-002068-STP), SEKT (IST-1-506826-IP) and PASCAL Network of Excellence (IST-2002-506778).

References

[1] T. Brants. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000*, pages 224–231, Seattle, WA, 2000. http://www.coli.uni-sb.de/~thorsten/tnt/.

[2] G. Chrupala. Simple data-driven context-sensitive lemmatization. In *Proceedings of SEPLN-2006*, Zaragoza, Spain, 2006.

[3] A. Clark. Memory-based learning of morphology with stochastic transducers. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, pages 513–520, 2002.

[4] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, pages 261–283, 1989.

[5] P. J. Compton and R. Jansen. A philosophical basis for knowledge acquisition. *Knowledge Acquisition*, 2:241–257, 1990.

[6] S. Džeroski and T. Erjavec. Learning to Lemmatise Slovene Words. In James Cussens and Sašo Džeroski, editors, *Learning Language in Logic*, number 1925 in Lecture notes in artificial intelligence, pages 69–88. Springer-Verlag, Berlin, 2000.

[7] T. Erjavec. MULTEXT-East Version 3: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC'04*, pages 1535 – 1538, Paris, 2004. ELRA. http://nl.ijs.si/ME/V3/.

[8] T. Erjavec. The English-Slovene ACQUIS corpus. In *Fifth International Conference on Language Resources and Evaluation, LREC'06*, pages 2138 – 2141, Paris, 2006. ELRA. http://nl.ijs.si/svez/.

[9] T. Erjavec and S. Džeroski. Machine Learning of Language Structure: Lemmatising Unknown Slovene Words. *Applied Artificial Intelligence*, 18(1):17–41, 2004.

[10] T. Erjavec and B. Sárossy. Morphosyntactic tagging of Slovene legal language. *Informatica*, 30(4):483–488, 2006.

[11] B. Gaines. Induction and visualisation of rules with exceptions. In *6th AAAI Knowledge Acquisition for Knowledge Based Systems Workshop*, pages 7.1–7.17, 1991.

[12] P. Kiparsky. "Elsewhere" in phonology. In Steven R. Anderson, editor, *Festschrift for Morris Halle*, pages 93–106. Holt, Rinehart and Winston, New York, 1973.

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

[14] S. Manandhar, S. Džeroski, and T. Erjavec. Learning Multilingual Morphology with CLOG. In David Page, editor, *Inductive Logic Programming; 8th International Workshop ILP-98, Proceedings*, number 1446 in Lecture Notes in Artificial Intelligence, pages 135–144, Berlin, 1998. Springer-Verlag.

[15] D. Mladenić. Combinatorial optimization in inductive concept learning. In *Proceedings of ICML*, pages 205–211, 1993.

[16] D. Mladenić. Learning word normalization using word suffix and context from unlabeled data. In *Proceedings of ICML*, pages 427–434, 2002.

[17] R. J. Mooney. Inductive logic programming for natural language processing. In *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 3–22, Berlin, 1997. Springer-Verlag.

[18] R. J. Mooney and M. E. Califf. Induction of First-Order Decision Lists: Results on Learning the Past Tense of English Verbs. *Journal of Artificial Intelligence Research*, 3(1):1–24, 1995.

[19] P. Nakov. BulStem: Design and Evaluation of Inflectional Stemmer for Bulgarian. In *Proceeding of the Workshop on Balkan Language Resources and Tools*, Thessaloniki, 2003.

[20] M. Porter. An algorithm for suffix stripping. In *ACM SIGIR Conference on Conference on Research and Development in Information Retrieval*, pages 318–327, 1980.

[21] R. L. Rivest. Learning decision lists. *Machine Learning*, pages 229–246, 1987.

[22] A. Srinivasan, P. Compton, R. Malor, G. Edwards, C. Sammut, and L. Lazarus. Knowledge acquisition in context for a complex domain. In *Proceedings of the European Knowledge Acquisition Workshop, Aberdeen*, 1991.

[23] N. Stroppa and Y. Francois. An analogical learner for morphological analysis. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 120–127. Association for Computational Linguistics, 2005.

[24] A. van den Bosch and W. Daelemans. Memory-based morphological analysis. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 285–292, Morristown, New Jersey, 1999. Association for Computational Linguistics.

[25] D. Yarowsky, G. Ngai, and R. Wicentowski. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research HLt-2001*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.