# Automatic Lemmatizer Construction with Focus on OOV Words Lemmatization*

Jakub Kanis and Luděk Müller

University of West Bohemia, Department of Cybernetics
Univerzitní 8, 306 14 Plzeň, Czech Republic
{jkanis,muller}@kky.zcu.cz

**Abstract.** This paper deals with the automatic construction of a lemmatizer from a Full Form - Lemma (FFL) training dictionary and with lemmatization of new, in the FFL dictionary unseen, i.e. out-of-vocabulary (OOV) words. Three methods of lemmatization of three kinds of OOV words (missing full forms, unknown words, and compound words) are introduced. These methods were tested on Czech test data. The best result (recall: 99.3 % and precision: 75.1 %) has been achieved by a combination of these methods. The lexicon-free lemmatizer based on the method of lemmatization of unknown words (lemmatization patterns method) is introduced too.

## 1 Introduction

A lemmatizer (which associates group of words with the same stem with one basic word form, base or lemma) or a stemmer (which associates group of words with their stem) is used in various kinds of natural language applications like information retrieval systems (IR systems) [1], TTS systems [2], word sense disambiguation, text classification [1], and translation (we intend to use a lemmatizer for translation from the Czech language to Sign Supported Czech and to the Czech Sign Language). However, a usage of a lemmatizer yields mixed results. For example, there are works dealing with IR systems where the usage of a lemmatizer leads to better performance of IR [4] on one hand and on the other hand there are works reporting no influence on the IR results [3]. In addition, Krovetz [3] has concluded that accurate stemming of English can improve performance of the IR system and Monz [4] has confirmed this conclusion. Thus, the accuracy or more exactly the recall (because generally it is possible to assign more than one lemma to one word) is an important issue of a lemmatizer. In this paper we introduce the automatic language independent construction of a high accurate lemmatizer. First, we describe the lemmatizer construction from a Full Form - Lemma (FFL) dictionary. We describe two methods for the lemmatization of full word forms which are not in the FFL dictionary but their lemma is present in the FFL dictionary. Then we introduce the method for lemmatization of unknown words, i.e. words whose neither full form nor lemma is in the FFL dictionary. In the next section we describe the experiments with the lemmatizer for several languages and show how the well made lemmatization of out-of-vocabulary (OOV) words can improve the lemmatization of a test data. The last section summarizes this paper.

* Support for this work was provided by GA AS Czech Republic, project No. 1ET101470416.

V. Matoušek et al. (Eds.): TSD 2005, LNAI 3658, pp. 132–139, 2005.

## 2 Construction of Lemmatizer

### 2.1 Lemmatizer

There are two main processes used for derivation of new words in a language: the inflectional and the derivative process. The words are derived from the same morphological class (for example the form CLEARED and CLEARS of the verb CLEAR) in the inflectional process while in the derivative process are derived from other morphological classes (CLEARLY). The creation of a new word can be reached by applying a set of derivation rules in the both processes. The rules provide adding or stripping prefixes (prefix rule) and suffixes (suffix rule) to derive a new word form. From this point of view, the lemmatization can be regarded as the inverse operation to the inflectional and derivative processes. Thus, we can obtain lemmatization rules via the inversion of the given derivation rules [2]. Alternatively, we can induce them simply from the FFL dictionary (see Section 2.2). The lemmatization rules induction is advantageous when derivation rules are given because the inducted rules and a lexicon of lemmas are error free contrary to the manually created ones which can contain some errors. The usual error is a mismatch between a derivation rule condition and the string that had to be stripped. We will compare the lemmatization rules generated trough the inversion of the handcrafted derivation rules and the lemmatisation rules induced from the FFL dictionary automatically in the experiment results section. The FFL dictionary was created by a morphological generator from the derivation rules. The generator produces the full word forms from the corresponding lemmas contained in the lexicon.

The set of derivation rules is a set of if-then rules (for example, a simple derivation rule is: if a word ends by $E$, then strip $E$ and add $ION$, i.e. in the symbolic form: $E > -E, ION$). The set of rules should cover all morphology events of the given language. The completeness of the lexicon strongly influences the successfulness of the lemmatization because a proper basic form (lemma) can be found only if it is included in the lexicon [2]. Unfortunately, there are a lot of OOV words in real natural language processing applications which should be also lemmatized. In addition, if the FFL dictionary is used for the lemmatization rules induction, there still can be some full forms of a word in the test corpora which are not in the dictionary. Therefore, in the next sections we describe two different methods for lemmatization of full forms which are missing in the FFL dictionary, and present a method for lemmatization of additional OOV words.

### 2.2 Induction of Lemmatization Rules from FFL Training Dictionary

The FFL dictionary consists of pairs: [full word form, lemma]. The induction of lemmatization rules is based on searching for the longest common substring of the full form and the lemma. We are looking for lemmatization rules in the form if-then rules described in the previous section. The algorithm of searching the longest common substring is based on dynamic programming. The detailed description of the algorithm is given in [5].

The form of the derived lemmatization rules depends on a position of the longest common substring in the full form and the lemma. The longest common substring can be at the beginning, in the middle, or at the end of the full form and the lemma. For

example, if we have a pair of strings $BAC$ and $ADE$, where $A$, $B$, $C$, $D$, and $E$ are their substrings (each substring is a sequence of characters, e.g. $A = a_1....a_n$), then we can derive two lemmatization rules, which transform the full form $BAC$ into the lemma $ADE$. The first one is the prefix rule $B > -B$ and the second one is the suffix rule $C > -C, DE$. The substring $B$ before and the substring $C$ after the longest common substring $A$ represents the condition of the prefix and the suffix rule, respectively.

We suppose that no more than two rules are applied to the lemma during the derivation process: the suffix and/or the prefix rule. To illustrate all possible forms of lemmatization rules, we show a table (Table 1) of pairs: [full word form, lemma] for all combinations of positions of the longest common substring $A$ in the pair of strings and the lemmatization rules derived from them.

**Table 1.** All possible forms of lemmatization rules used in the inductive process

| Full Form | Lemma | Prefix Rule | Suffix Rule | Alternative Rules | |
|---|---|---|---|---|---|
| **ABC** | **ADE** | | BC > -BC, DE | | |
| **ABC** | **DAE** | **A > D** | BC > -BC, E | . > D | |
| **ABC** | **DEA** | **A > DE** | BC > -BC | . > DE | |
| **BAC** | **ADE** | B > -B | C > -C, DE | | |
| **BAC** | **DAE** | B > -B, D | C > -C, E | | |
| **BAC** | **DEA** | B > -D, DE | C > -C | | |
| **BCA** | **ADE** | BC > -BC | **$a_n$ > DE** | | . > DE |
| **BCA** | **DAE** | BC > -BC, D | **$a_n$ > E** | | . > E |
| **BCA** | **DEA** | BC > -BC, DE | | | |

There are general lemmatization rule forms in the third and the fourth column, which we use in the inductive process of lemmatization rules. In the second, third, seventh and eighth rows there are also alternative rules in the last two columns (the dot in the rule means an arbitrary string). These alternative rules were derived in the same way as the rules in the example above ($BAC$ and $ADE$). Because there are no substrings before (row 2 and 3) or after (row 7 and 8) the common substring $A$, the derived lemmatization rules are stripped-only rules (see [2]) and therefore, they have to be replaced by other no stripped-only rules. The condition of a new prefix rule is the whole common substring $A$ and the condition of a new suffix rule is the last character $a_n$ of substring $A$. If there is no common substring then a rule which substitutes the full form for its lemma is created. For example, the pair of words $JE$ and $ON$ creates the rule: $JE > -JE, ON$. The absence of common substring is caused by the presence of irregular words in the language, i.e. words with the irregular inflectional and derivative process. Every induced rule has its identification code (*rule id* or *rule index*). Every lemma together with a set of ids of lemmatization rules which has been induced from this lemma (called "*lemma applicable rules*"), i.e. a sort list of the rule ids, is stored in the lemma lexicon [2].

### 2.3 Lemmatization of Missing Full Forms

When we want to construct the lemmatizer from the FFL dictionary, we have to cope with the following problem. The problem is the absence of some full forms in the FFL

dictionary, especially if we have the FFL dictionary which has not been created by the morphological generator.

**Generalization of Lemma Lexicon (GLL).** This method works with the lemma lexicon, which has been build in the inductive process of lemmatization rules. A lemma lexicon entry is a lemma with its "lemma applicable rules". Suppose that for some lemma and its relevant full forms the lemma lexicon contains only the information on rules which are used during the lemmatization of these full forms. This information is checked in the lemmatization process and thus the situation that the relevant full form is missing causes that this missing full form cannot be lemmatized. To provide the lemmatization of the missing relevant full forms, we need to add the ids of rules which lemmatize these missing forms to the lexicon. This problem can be viewed as a problem of automatic finding "lemma applicable rules" patterns or as lemma clustering based on the "lemma applicable rules". We assume that there are some lemmas with their all relevant full forms in the FFL dictionary which can be served as the patterns. Once we have the patterns, we assign them to the lemmas in the lexicon and create a new lemma lexicon consequently.

To find the patterns, we create a co-occurrence matrix $A$ of dimension $n$ x $n$, where $n$ is the number of lemmatization rules. The rule ids denote row and column indexes of $A$; the matrix element $a_{ij}$ comprises the information on how many times the rule $i$ together with the rule $j$ has been seen in the list of the rule ids. Now we go trough all the lemmas in the lexicon and count how many times the rule $i$ together with the rule $j$ has occurred in the lemma list of the rule ids. The rows in the matrix are treated as searched patterns, i.e. if we see the rule $i$ in a lemma list we enrich this list by adding the indexes of all columns (the rules) whose elements of the i-th row are positive. This enrichment brings a higher recall but a lower precision. A better way is to enrich the list by the column indexes which score is higher than some threshold. We used the threshold equal to one and obtained increasing in the recall by 2.58% but decreasing in the precision by 5.6% for our development data (for more detail see section 3). Because this method decreases the precision we develop another method: Hierarchical Lemmatization without Rule Permission Check.

**Hierarchical Lemmatization Without Rule Permission Check (HLWRPC).** The first problem of the previous method is that the enriched lexicon is used on all lemmatized words. The second one is that finding the right patterns which do not drop the precision is a very difficult task. We should make some changes in the lemmatization process to cope with these two problems. First, we try to use the lemmatization algorithm described into [2] on the investigated word. If it finds some lemma then this lemma is considered as the result otherwise the investigated word is the missing full form. In this case, we use a new lemmatization algorithm without the rule permission check, i.e. if we find some lemma in the lemma lexicon then we do not check if the lemmatization rules used during this lemmatization process are "lemma applicable rules". In this way, the lemma lexicon is totally generalized without the negative influences on a precision and a recall. This method increases the recall by 2.7% and the precision by 0.1% for development data (details are given in Section 3).

### 2.4 Lemmatization of OOV Words

The OOV words are words which are not in the FFL dictionary and therefore, we cannot lemmatize them by the lemmatization algorithm [2]. There are three types of the OOV words. The first type is the situation when the full form is missing in the FFL dictionary but its lemma is in the FFL dictionary. The lemmatization of the missing full forms has been described in the previous section. The second type is a word whose neither full form nor lemma is in the lexicon. This word is called an unknown word. The last type is a compound word whose partial word has its lemma in the lexicon. For lemmatization of compound words we use the same lemmatization algorithm (without rule permission check) as the one for the missing full forms. The difference is that we do not lemmatize the compound word directly. First, we remove some characters from the beginning of the compound word and subsequently the rest is lemmatized by the lemmatization algorithm without rule permission check. The question is: What is the minimal length of the rest which still can represent some full form? We provide several experiments and chose the length of six characters as the minimal length of the rest.

A set of "*word applicable rules*" (rules which condition is true for the given word) can be assigned to every word. One set can be assigned to several different words and hence it can be considered as the lemmatization pattern. In order to create the lemmatization patterns we go trough a whole FFL dictionary and for every pair [full form, lemma] we find "word applicable rules" (the lemmatization pattern). This lemmatization pattern together with the ids of winning rules and a *count of winnings* of every winning rule is saved to the pattern table. The winning rule is every rule which converts the full form to the lemma (if the full form is the same as the lemma then the winning rule is the empty rule but it is taken into account too). If the pattern already exists then we increase the score of the winning rules only. The winning rules are sorted by their count of winnings. We use two pattern tables - the prefix pattern table (word applicable prefix rules only) and the suffix pattern table (word applicable suffix rules only) specially. When we lemmatize the unknown word, we try to apply all the lemmatization rules and the applicable rules create prefix and suffix pattern. Then we find these patterns in the relevant table and apply the winning rules which have the highest count of winnings on the unknown word.

## 3 Experiments and Results

Two data sources have been used for the experiments, the Prague Dependency Treebank (PDT 1.0) and dictionaries and rule files from the Ispell [7]. The PDT 1.0 is a corpus of annotated Czech texts having three-level structure [6]: morphological, analytical, and tectogrammatical. For the construction and evaluation of the lemmatizer we have used only training, development, and test data from the morphological level. The data from the PDT 1.0 represent the FFL dictionary (PDT 1.0 FFL dictionary) with missing full forms. Ispell dictionary and rules file represent FFL dictionary (Ispell FFL dictionary) without missing full forms. The Ispell FFL dictionary is created by the morphological generator from the ispell dictionary and the derivation rules file. The Ispell FFL dictionary is used to compare the lemmatization rules acquired via inversion of manually handcrafted derivation rules with the lemmatization rules induced from a FFL dictionary. We have chosen the Ispell dictionaries and rules files for the Czech, English, and

Finnish language because these languages represent three basic types of language: inflective, analytical, and agglutinative language, respectively.

The lemmatizer output should be all lemmas from which the lemmatized word can be derived. This is a multiple output thus we have to count a recall ($R$) and a precision ($P$). The recall is computed as the number of the right lemmatized words to the number of all lemmatized words ratio. The word is lemmatized correctly when there is its reference lemma in the lemmatizer output. The reference lemma is the unambiguous lemma which is assigned to the given word by a human expert. The precision is computed as the ratio of the number of the right lemmatized words to the number of all lemmas generated by the lemmatizer for all correct lemmatized words.

We have created two lemmatizers to compare the inversion and the induced lemmatization rules, the first one (Lem_LM) is based on language morphology (lemmatization rules acquired via the inversion of manually handcrafted derivation rules) [2] and the second one (Lem_FFL) is induced from the Ispell FFL dictionary. We have used the Ispell FFL dictionary also for the comparison of the recall and the precision of these two lemmatizers. The results are shown in Table 2.

**Table 2.** The comparison of the lemmatizers

| Language | Czech | | | | | English | | | | | Finnish | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R [%] | P [%] | N. of errors | N. of prefix rules | N. of suffix rules | R [%] | P [%] | N. of errors | N. of prefix rules | N. of suffix rules | R [%] | P [%] | N. of errors | N. of prefix rules | N. of suffix rules |
| Lem_LM | 99.99 | 94.85 | 280 | 13 | 2533 | 96.56 | 95.23 | 4 750 | 3 | 43 | 99.62 | 94.27 | 19765 | 6 | 18612 |
| Lem_FFL | 100 | 95.63 | 0 | 52 | 1619 | 100 | 95.04 | 0 | 3 | 33 | 100 | 94.27 | 0 | 14 | 11271 |

We lemmatize the training data (Ispell FFL dictionary created by the morphological generator), so the recall should be 100 %. However, it is valid for the Lem_FFL only because there are errors in the ispell rules files [2]. The Lem_FFL (recall 100 %) can cope with these errors.

The methods for the lemmatization of missing full forms, unknown, and compound words have been tested on the development and test morphological data from the PDT 1.0. The results are given in Table 3.

**Table 3.** The results of the methods for the lemmatization of OOV words

| Method | Morphological development data | | | Morphological test data | | |
|---|---|---|---|---|---|---|
| | R [%] | P [%] | # of errors | R [%] | P [%] | # of errors |
| Lem_FFL | 95.06 | 76.42 | 5212 | 95.41 | 76.74 | 4736 |
| Lem_FFL_Gen_Dic | 97.64 | 70.8 | 2492 | X | X | X |
| Lem_FFL_Hierar | 97.8 | 76.53 | 2356 | X | X | X |
| Lem_FFL_Compound | 95.82 | 76.1 | 4411 | X | X | X |
| Lem_FFL_Unknown | 98.6 | 76.14 | 1481 | X | X | X |
| Lem_FFL_Hierar_Cmd_Unk | 99.31 | 74.59 | 726 | 99.3 | 75.1 | 712 |

In the first row the result for lemmatizer (Lem_FFL) trained on the PDT 1.0 FFL dictionary is given. In the next rows the results of the methods for the lemmatization missing full forms (GLL – Lem_FFL_Gen_Dic; HLWRPC – Lem_FFL_Hierar), compound (Lem_FFL_Compound), and unknown (Lem_FFL_Unknown) words used with

Lem_FFL lemmatizer are shown. The best result, which has been achieved by a combination of the methods, is in the last row.

The presented method for the lemmatization of the unknown words can lemmatize every word and works without a lemma lexicon. It can be used like a lexicon-free lemmatizer also. This lemmatizer has been tested on the ispell FFL dictionaries and on the development data from the PDT 1.0. The results are presented in Table 4.

**Table 4.** The lemmatizer based on the lemmatization of the unknown words only

| | Ispell FFL dictionaries | | | PDT 1.0 – morphological development data | |
|---|---|---|---|---|---|
| | Czech | English | Finnish | All development data | Lemmas not seen in training data |
| Lem_WR | R: 84.86 %<br>P: 68.89 % | R: 79.6 %<br>P: 96.94 % | R: 71.91 %<br>P: 98.08 % | R: 76.44%<br>P: 78.41% | R: 82.01 %<br>P: 82.37 % |
| Lem_All | R: 100 %<br>P: 8.16 % | R: 100 %<br>P: 48.89 % | R: 100 %<br>P: 31.98 % | R: 99.85 %<br>P: 10.32 % | R: 98.59 %<br>P: 9.36 % |
| Lemmatization patterns | | | | | |
| # of prefix patterns | 51 | 3 | 14 | 192 | |
| # of suffix patterns | 1241 | 33 | 8522 | 2177 | |

In the first row is the method (Lem_WR) which applies the winning lemmatization rules, with the highest count of winnings, of a relevant pattern on the lemmatized word. The second method (Lem_All) applies all winning lemmatization rules (independently on the count of winnings) of the relevant pattern.

## 4 Conclusions

We have introduced the method for the automatic construction of the lemmatizer from the FFL dictionary and the methods for lemmatization of three types of the OOV words (missing full forms, unknown (missing lemmas), and compound words). The methods have been evaluated on different types of data and the best result achieved on the test data had recall 99.3 % and precision 75.1 %. The baseline experiment was carried out with the lemmatizer induced from the PDT 1.0 FFL dictionary without applying any methods of lemmatization of OOV words. The baseline method achieved recall 95.41 % and precision 76.74 % (see Table 3). We have shown that the induction of the lemmatizer from a FFL dictionary is useful in case when a lemma lexicon and derivation rules are at disposal (see Table 2). The method for the lemmatization of the unknown words can be used as a lexicon-free lemmatizer (see Table 4).

## References

1. Gaustad, T.: Linguistic knowledge and word sense disambiguation. Groningen Dissertations in Linguistics, (2004), ISSN 0928-0030
2. Kanis J., Müller L.: Using the lemmatization technique for phonetic transcription in text-to-speech system, In Text, speech and dialogue. Berlin: Springer, (2004). s. 355-361.ISBN 3-540-23049-1.

3. Krovetz, R.: Viewing morphology as an inference process. In Horfhage,R., Rasmussen,E., and Willett, P., editors, Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 191-203, Pittsburgh (1993).
4. Monz, C.: From document retrieval to question answering. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam (2003).
5. Daniel Hirschberg's page: http://www.ics.uci.edu/ dan/class/161/notes/6/Dynamic.html
6. Böhmová, A., Hajič, J., Hajičová, E., Hladká, B.: The Prague Dependency Treebank: Three-Level annotation scenario. -In: A. Abeillé, editor, Treebanks: Building and using syntactically annotated corpora. Kluwer Academic Publishers (2001).
7. Ispell dictionaries and rules files: http://fmg-www.cs.ucla.edu/geoff/ispell-dictionaries.html