Using Lemmatization Technique for Automatic Diacritics Restoration

Jakub Kanis, Luděk Müller

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

Abstract

This paper is devoted to automatic construction of a lemmatizer from a Full Form - Lemma (FFL) training dictionary, and to 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. In addition, the application of lemmatizer automatic construction to the problem of automatic diacritics restoration is described.

1. Introduction

In contrast to English, almost every language with Latin alphabet uses some additional special characters. Those characters are called accented characters and are created from standard characters using diacritics. For example in Czech language there are fifteen (in lower case) accented characters (á, č, ď, é, ě, í, ň, ó, ř, š, ť, ú, ů, ý, ž) and three diacritics (´, ˇ, ˚). On the other hand there are only three accented characters (ä, ö, ü) and one diacritic (¨) in German.

The problem arises when diacritics are missing and we need to restore them - for example for speech synthesis of email messages written without diacritics. The missing diacritics are typical for email and SMS (Smart Message Service) communication. For example if one write an SMS message on the cellular phone then it is needed two characters for one accented character and the SMS message is then bigger than the message with unaccented characters. This phenomenon is undesirable because a standard SMS message has a limited size. The second problem is that the unaccented characters can be typed quicker than accented ones. Thus writing with unaccented characters is faster and people usually do not use diacritics.

The basic method for diacritics restoration uses a dictionary of unaccented word forms and corresponding accented forms. The problem is that for some unaccented forms exists more than one accented form. There are methods for selection of one accented form. The basic method is to select the form with the highest frequency of occurrence. The better method uses POS tagging for resolution of ambiguous accented forms [1]. The best method is based on usage of decision lists for homograph disambiguation [2].

In this paper we want to show that the lemmatizer with lemmatization patterns based on automatic induction of lemmatization rules from Full form - Lemma (FFL) dictionary can be used for diacritics restoration. The usage of lemmatizer has ability of processing out-of-vocabulary (OOV) words (see section 2.4) in contrast to the usage of the dictionary only.

In the next section we describe the lemmatizer construction from the FFL dictionary and the method used for OOV words lemmatization. The description of lemmatization and diacritics restoration experiments and its results are given in the third section. The last section summarizes this paper.

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 [3]. 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.

The set of derivation rules is a set of if-then rules (for example, a simple derivation rule is: if a word ends with 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 [3]. 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 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 [4].

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 [3]) 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 [3].

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.

2.3.1. 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 Table 2 in Section 3). Because this method decreases the precision we develop another method: Hierarchical Lemmatization without Rule Permission Check.

2.3.2. 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 [3] 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 Table 2 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 [3]. 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.

To every word a set of "*word applicable rules*" (rules which condition is true for the given word) can be found. One set can be assigned to several different words and hence it can be considered as a 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) separately. When we lemmatize the unknown word, we try to apply all the lemmatization rules, and the applicable rules then 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

The data source used for the experiments was the Prague Dependency Treebank (PDT 1.0). The PDT 1.0 is a corpus of annotated Czech texts having three-level structure [5]: 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. From the PDT 1.0 training data we extracted a set of pairs of full word form and lemma which represents the training FFL dictionary (PDT 1.0 FFL dictionary) for our experiments.

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 a ratio of number of the right lemmatized words to the number of all lemmatized words. 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.

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 2.

Table 2: *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.

For the diacritics restoration experiments we use the FFL dictionary which consists of pairs: [unaccented word form, accented word form]. This dictionary was extracted from morphological training data of PDT 1.0 but it can be extract from arbitrary raw accented texts because no additional annotation is needed (what is needed for annotation is diacritics removal). The results are given in Table 3.

In the first and the second row are the results for development data. In the rest of rows are the results for OOV words from development data only. The baseline for development data restoration is in the first row (Development 1 - usage of lemmatizer without methods for lemmatization of OOV words). In the third row is the baseline for OOV words restoration (OOV Words 1 - no diacritics restoration - a word is leaved in the same form). In the fourth row (OOV Words 2) is the result for the restoration of OOV words where we use all methods for OOV words lemmatization (missing full forms, compound words and

Table 3: *The results of the diacritics restoration.*

Data	R [%]	P [%]	# of errors
Development 1	95.05	76.88	5221
Development 2	97.47	75.40	2666
OOV Words 1	33.36	100	5093
OOV Words 2	66.79	59.98	2538
OOV Words 3	56.88	100	3295

unknown words) thus there are more possible results (P = 59.98 %). In the last row is then the result for the method where we choose only one possible result (P = 100 %).

4. Conclusions

We have introduced a method for the automatic construction of the lemmatizer with lemmatization patterns from the FFL dictionary and methods for lemmatization of OOV words. The methods have been evaluated on development data. The best result achieved on the test data had recall 99.3 % and precision 75.1 % (Table 2). In the next experiment we have shown that this method for automatic construction of lemmatizer can be used for the automatic diacritics restoration, especially in case of OOV words diacritics restoration. The results for diacritics restoration: recall 97.47 % and precision 75.4 %, are similar to ones for lemmatization of words. The main advantage of this method is diacritics restoration for OOV words. The results for OOV words diacritics restoration were: recall 66.79 % and precision 59.98 %. The precision, which is smaller than 100 %, means that there are more possible results. If we choose one result only then the recall drops to 56.88 % (Table 3).

The method for diacritics restoration based on automatic construction of the lemmatizer with lemmatization patterns from the FFL dictionary is language independent and can be simply adapted to another language or domain. For the adaptation we need the new FFL dictionary only. This dictionary can be prepared from arbitrary accented text easily. The preparation consists in diacritics removal.

5. Acknowledgment

This work was supported by the Ministry of Education of the Czech Republic under project MŠMT LC536.

6. References

[1] Tufiş, D., and Chiţu, A. Automatic diacritics insertion in Romanian texts. In Proceedings of the International Conference on Computational Lexicography COMPLEX ’99 (Pecs, Hungary, June 1999)

[2] Yarowsky, D., Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pp. 88–95 Las Cruces 1994

[3] 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.

[4] Daniel Hirschberg’s page: http://www.ics.uci.edu/ dan/class/161/notes/6/ Dynamic.html

[5] Böhmová, A., Hajič, J., Hajičová, E., Hladká, B.: The Prague Dependency Treebank: Three-Level annotation scenario. -In: A. Abeill, editor, Treebanks: Building and using syntactically annotated corpora. Kluwer Academic Publishers (2001).