ITM Web of Conferences **6**, 03013 (2016)
DOI: 10.1051/itmconf/20160603013

The fast vocabulary-based algorithm for natural language word form analysis

Alexey Rozanov[1,a]

[1]*Ryazan State Radioengineering University, Department of computational and applied mathematics, Ryazan*

In the field of Natural Language Processing, identifying word forms and, more precisely, identifying part-of-speech and grammatical information for each of the words in the input text usually comprises the very first level of text processing (or immediately follows splitting the text into words, should such task be non-trivial), therefore development of approaches to speed up the word form analysis pose significant interest. In this work, by using the work [1] as a basis, we present an approach to analysis of word forms for natural languages with postfix inflection, following the work done in [3]. We propose a way of representing the postfix inflection rules associated with a natural language and an algorithm for word form analysis based on it. In conclusion, we provide the benchmark data indicating the increase in speed compared to known analysis methods.

1 Introduction

In his works, A. Prutzkow proposed a universal model for representing the word form generation rules [1] and an algorithm for word form analysis of a natural language [2].

All rules in the aforementioned model are described as chains of elementary transformations of strings, each one possessing the properties of **unambiguity** (the same word is always transformed with the same result) and **reversibility** (for each transformation P, there is exactly one transformation P^{-1} such that $P^{-1}(P(s)) = s$ for any string s the transformation P is applicable to).

Albeit the model and the algorithm utilizing it, both proposed in [1], do have the advantages of being language-independent and bi-directional (i.e. allowing both generation and analysis), the speed of word form analysis is significantly lower compared to known analogues.

In this article we only consider word form analysis for languages with postfix inflection, i.e., languages in which word form generation rules only replace postfixes in the word, leaving the stem unchanged. As shown in [3], the model described above is redundant for such languages and can be reduced to a much simpler model, which allows using a more efficient algorithm for word form analysis.

Then we propose an approach to structuration of vocabulary for word form analysis software. It allows representing complex word paradigms efficiently. We consider the advantages of proposed vocabulary structuration and conclude the article with the performance test data.

2 Organizing the rule set

It is shown in [3] that, given the language inflection is strictly postfix, every word form generation rule can be represented by using a single postfix substitution operation, which is easily seen to be unambiguous (deterministic) and reversible.

Therefore, one might represent the whole set of such rules as distinct connections C_i between two prefix trees L (left) and R (right), correspondingly representing the set of postfixes the initial forms possess and the set of their replacements. Figure 1 displays the resulting data structure.

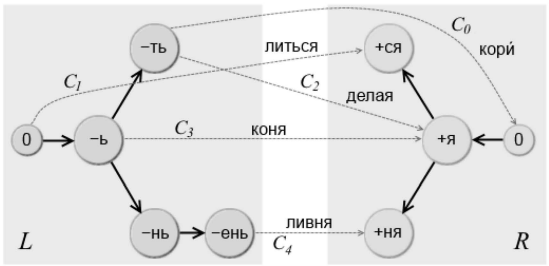


Figure 1. Connections between prefix trees defining rules for word form generation.

The data structure shown in Figure 1 illustrates how the rules for word form generation are stored in memory. In this example, a few rules for the Russian language are shown: C_0 represents the generation of imperative ("*кори́*") for the Russian verb "*кори́ть*" (Eng. *to blame*), as well as many other Russian verbs, C_1 is used to obtain the reflexive form ("*литься*") of the verb "*лить*" (Eng. *to pour*), C_2 provides the rule for obtaining the gerund form of the verb "*делать*" (Eng. *to do*), C_3 shows how to

[a] e-mail : thehdotx@gmail.com

generate genitive (or accusative, as said two forms of this word do not differ) for the noun "конь" (Eng. *male horse*), C_4 generates the genitive (but not accusative) for the noun "ливень" (Eng. *downpour*).

In the above figure, each node of the left tree L represents a certain valid Russian postfix (i.e. at least one word has it in the initial form). This postfix is what should be removed in order to obtain a word form that differs from the initial one. The "zero" node corresponds to the null postfix meaning that no remove is needed.

Similarly, each node of the right tree R represents a certain valid (at least for a single word in its non-initial form) Russian postfix. The said postfix is what should replace the removed postfix of the initial form of the word. The "zero" node represents the null postfix, meaning that no new postfix is needed.

Then, given the above mentioned data structure represents the complete inflection rule set, the algorithm for word form analysis, contains the following steps:

(a) obtaining the chain Z of right tree nodes that matches the characters of the input word taken in the reverse order, thus eliminating most of the rules as impossible for the given input word;

(b) taking the next possible way of splitting the input word into a [*stem+postfix*] pair by iterating thtough $\{Z_i\}$;

(c) scanning through all rules associated with node Z_i to filter out the rules whose inversions, being applied to the input word, do not produce the initial form they were initially associated with;

(d) the set of the rules obtained gives the information on all possible interpretations of the input word, as well as the initial word forms for each of the interpretations found at step (c), so the word is considered unknown if the resulting set is empty.

Thus, it can be seen (and is shown in [3]) that

• the set of the rules to iterate through when analyzing a word is reduced to a small (compared to the entire Russian rule set) set of the rules represented by the connections C_k linked to the nodes of a single chain of the tree R (and not the whole tree);

• the application of any given rule to a word is as trivial as the postfix replacement is in the programming language the programmer decided to implement the algorithm in;

• the memory overhead is negligible (around 2Mb for the trees built from the test vocabulary containing over 8000 words in initial form) in comparison to the vocabulary size (said vocabulary, when deserialized, took about 20Mb of RAM), therefore the analysis method is suitable for running on devices possessing small amounts of RAM;

• the analysis can be executed in parallel very well, as it requires no writing to shared memory, therefore the speed of analysis increases proportionally to the number of parallel execution threads.

3 Representing grammatical information

When planning the vocabulary structure of the word form analysis system, we usually face the problem of organizing the initial word forms into groups sharing similar (or equal) word form generation rule sets. In languages with complex inflection rules, the structure of a word paradigm may depend on a number of circumstances including the values of certain categories (e.g. perfective or imperfective aspect of a verb, comparability of an adjective) associated with the word in question. Therefore, we need to divide the whole set of vocabulary words so that the amount of duplicate rules in similar word groups would be as small as possible.

In [4], the author proposes a hierarchical representation of the vocabulary and demonstrates that it is sufficient to describe the full paradigm structure for Russian words. The proposed model is largely language-independent, as well as it does not depend on the word generation rule kinds existing in the language. The proposed model can be described using the following sequence of definitions:

1. The whole set of **grammatical categories** of the language is stored in a special global registry, i.e.

$$GValues = \{GValue_1, GValue_2, \ldots, GValue_N\}$$
$$GTypes = \{(GType_1, GValues_1), \ldots, (GType_2, GValues_2)\}$$

Examples of particular grammatical categories descriptions for the Russian language are

(*Case*, {*Nominative*, *Genitive*, ..., *Prepositional*})
(*Number*, {*Singular*, *Plural*})
(*Animacy*, {*Inanimate*,*Animate*})

2. When describing words and sets of words, it is imperative that the word or word group being described can have only one or zero GValues per GType. In the latter case we say that the grammatical category in question is not applicable (e.g. verb-related categories for nouns), or is undefined. The description (be it partial or complete) is called a **grammatical value map** (GValueMap). Examples for the Russian language are

(1) "*делами*":
{{*Case*:*Instrumental*}, {*Number*:*Plural*},
{*Animacy*:*Inanimate*}, {*Gender*:*Neuter*}}
(the initial word form "*дело*" is the Russian for "*deed*")

(2) *animate feminine nouns*:
{{*Animacy*:*Animate*},{*Gender*:*Feminine*}}

Grammatical value maps are used in the proposed model both for representing the analysis results and describing word groups.

3. Thus, the word form generation rules are represented as the (*Operation*, *GValueMap*) pairs where *Operation* is the exact low-level string manipulation procedure for obtaining the word form from the initial (vocabulary) form, and *GValueMap* is the grammatical information. For example, the word "*делами*" is generated by rule

R = (–о+ами):{{*Case*:*Instrumental*},{*Number*:*Plural*}}.

Note how no animacy is included in GValueMap, as it is already set for the word family containing the word.

4. GTypes often depend on each other: for example, in Russian, gender is undefined in plural forms; possible participles depend on aspect and transitivity. Due to that, listing all variable GTypes is not enough to efficiently describe a paradigm; therefore, the concept of **Inflection Maps** is introduced.

The idea behind inflection maps is to present the paradigm structure by splitting it into subsets formed by groups of independent GTypes. Each inflection map is composed of a GValueMap listing all invariant GValues,

a set of variable GTypes, and a set of restrictions forced upon particular variable GTypes.

To illustrate how the inflection maps can be used to describe the structure of the paradigm for a group of words, we consider the paradigm of Russian adjective "ясный" (Eng. *clear*) as an example (Table 1).

Table 1. Full Russian adjective paradigm structure.

Comparison degree, short/long forms, Cases \ Number, Gender, Animacy			Singular			Plural, Animate/ Inanimate
			Masculine Animate/ Inanimate	Feminine	Neuter	
Positive	Long form	Nom.	ясный	ясная	ясное	ясные
		Gen.	ясного	ясной	ясного	ясных
		Dat.	ясному	ясной	ясному	ясным
		Acc.	ясного/ ясный	ясную	ясное	ясных/ ясные
		Inst.	ясным	ясной/ ясною	ясным	ясными
		Prep.	ясном	ясной	ясном	ясных
	Short form		ясен	ясна	ясно	ясны
Comparative			яснее			

It is easily seen that GTypes shown in the table above are not independent, and for these six GTypes (*Number*, *Gender*, *Animacy*, *Comparison degree*, *Adjective form* (short or long), and *Grammatical Case*), there are just 30 distinct combinations of GValues that are grammatically correct (out of 288 theoretically possible, yielded as the product of GValue counters for each of the GTypes listed, i.e. 2×3×2×2×2×6). Therefore, we present the paradigm structure using 8 inflection maps:

$InflectionMap_1$: ({{*Number*:*Plural*},{*Form*:*Short*}},∅,∅)
(short plural form is unique)

$InflectionMap_2$:({{*Number*:*Singular*},{*Form*:*Short*}},{*Gender*},∅)
(short singular)

$InflectionMap_3$: ({{*ComparisonDegree*:*Comparative*}},∅,∅)
(comparative is unique too)

$InflectionMap_4$: ({{*Number*:*Plural*}},{*Case*},{*Case*≠*Accusative*})
(long form, plural)

$InflectionMap_5$:
({{*Number*:*Singular*}},{*Case*,*Gender*},{*Case*≠*Accusative*})
(long form, singular)

$InflectionMap_6$:
({{*Number*:*Plural*},{*Case*:*Accusative*}},{*Animacy*},∅)
(Inflection maps 6..8 describe all accusative forms)

$InflectionMap_7$:
({{*Number*:*Singular*},{*Case*:*Accusative*}},{*Gender*},
{*Gender*≠*Masculine*})

$InflectionMap_8$:
({{*Number*:*Singular*},{*Case*:*Accusative*},{*Gender*:*Masculine*}},
{Animacy},∅)

5. Next, the whole set of words in initial forms is divided into subsets sharing the same sets of word form generation rules. Each of such subsets is associated with a special shared GValueMap describing the invariant grammatical information such as gender and animacy for nouns.

At this point, we should have no trouble building the word form analyser, as the analysis is the process of determining the initial word form and the grammatical values map for the input word. Once an applicable rule is found, the latter goal is achieved by merging the GValueMap associated with the found rule with the GValueMap associated with the word subset the found rule belongs to, and the former goal is achieved by applying the reverse of the rule to the input word.

However, further structuration of the vocabulary is necessary because otherwise the process of filling it with new words requires much more redundant actions. The advantages of the proposed additional hierarchical vocabulary structuration include

(a) automatic generation of paradigm templates for new words based on their classification (for Russian verbs, given its aspect and transitivity, the complete paradigm template differs);

(b) guessing of particular rules for such templates can be done more precisely by taking into account how close to the new word was the existing rule in the hierarchy.

6. In the proposed model, the hierarchy of word types contains three levels (**Supertypes** – **Kinds** – **Families**) described below.

(a) **Supertypes** are the largest sets of words sharing the same invariant grammatical information structure. Also, at the supertype level, the set of all possible infection maps is described. For example, Russian nouns are presented as the supertype containing 2 invariant GTypes (Animacy and Gender) and a single inflection map (∅, {*Case*, *Number*}, ∅), which means that in Russian most nouns have 12 distinct forms, with some exceptions discussed below.

(b) **Kinds** are the subsets comprising supertypes, which are characterized by a GValueMap (shared by each of the words in the subset), a set of applicable inflection maps (some of the maps declared in the supertype might be inapplicable to some of the words comprising said Supertype), a set of words grouped in **families** (which are discussed below), and a set of **restrictions** (often empty) forced upon particular inflection maps. For example, in the supertype describing Russian nouns, there are 8 kinds, namely *Inanimate Masculine*, *Inanimate Feminine*, *Animate Masculine*, *Animate Feminine*, *Neuter*, *Plural Nouns*, *Singular Nouns*, *Indeclinable nouns*. The kind for *Plural Nouns*, for example, has a non-empty GValueMap {{*Number:Plural*}}, which reduces the amount of possible word forms to 6, (i.e. one per grammatical case).

(c) Families are the smallest subsets of words containing words that share the same word form generation rule set. For example, one of the biggest families of Russian nouns is the family of inanimate masculine nouns containing the word "*телефон*" (Eng. *phone*). The complete rule set associated with this family consists of the following 12 rules:

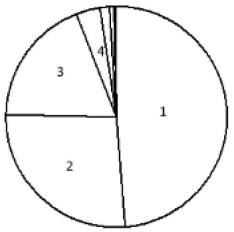
R_1 = 0:{{*Case*:*Nominative*},{*Number*:*Singular*}}
R_2 = (+а):{{*Case*:*Genitive*},{*Number*:*Singular*}}
R_3 = (+у):{{*Case*:*Dative*},{*Number*:*Singular*}}
R_4 = 0:{{*Case*:*Accusative*},{*Number*:*Singular*}}
R_5 = (+ом):{{*Case*:*Instrumental*},{*Number*:*Singular*}}
R_6 = (+е):{{*Case*:*Prepositional*},{*Number*:*Singular*}}
R_7 = (+ы):{{*Case*:*Nominative*},{*Number*:*Plural*}}
R_8 = (+ов):{{*Case*:*Genitive*},{*Number*:*Plural*}}
R_9 = (+ам):{{*Case*:*Dative*},{*Number*:*Plural*}}
R_{10} = (+ы):{{*Case*:*Accusative*},{*Number*:*Plural*}}
R_{11} = (+ами):{{*Case*:*Instrumental*},{*Number*:*Plural*}}
R_{12} = (+ах):{{*Case*:*Prepositional*},{*Number*:*Plural*}}

Note how rules R_1 and R_4 have zero transformation because Russian inanimate non-feminine nouns have the

same singular form in nominative and accusative cases, and nominative singular is considered the initial form for all Russian nouns that have it.

4 Performance comparison

In order to test the analysis speed, the author prepared a vocabulary containing some of the most frequently used Russian words (figure 2 shows the vocabulary statistics).



1 Nouns	4135
2 Verbs	2267
3 Adjectives	1603
4 Adverbs	304
5 Unchangeable words	104
6 Numerals	50
7 Pronouns	27

Figure 2. Test Russian vocabulary statistics.

Although the vocabulary only slightly exceeds 8000 initial word forms, the full bank of all recognizable word forms buildable from said initial forms would contain over 240000 words. As a result, more than 70% of the test input was recognized properly, thus allowing estimating the analysis speed and comparing it to that of the Mystem analyzer [5].

The performance tests were carried out on an Intel Core i7-4800MQ (2.7GHz), 16Gb RAM, running Windows 7. The algorithms were implemented in C# 6.0 (using .NET 4.5 framework).

In the test scenario involving a big corpora of words containing no duplicates (to exclude the effects of caching), the implementation of the proposed algorithm written by the author turned out to be slower than Mystem (the speed ratio varied from 1 to 0.5 depending on the amount of unknown words in the input text).

On the other hand, in the scenario involving a regular text (containing lots of repeating words), the implementation of the proposed algorithm has shown better speed (ratio varying from 1.5 to 2) and, thanks to caching, analyzing text B twice as big as text A never took twice as long as analyzing text A.

The roughly averaged sample absolute values are presented in table 2.

Table 2. Word analysis speed comparison.

	Proposed algorithm implementation	Mystem	Speed Ratio
No repeating words, 250000 words	55000 words/second	80000 words/second	0.69
Regular Russian text, 250000 words	180000 words/second	120000 words/second	1.5

5 Conclusion

In this article, we have demonstrated an approach to natural language word forms analysis using a general algorithm independent of particular language inflection rules, as long as all rules can be reduced to postfix replacement.

Given the fact that the performance tests were carried out using a high-level programming language, we could expect even better speeds, should the proposed method be implemented using a low-level programming language and a quality optimizing compiler.

Acknowledgements

The author is grateful to prof. Prutzkow Alexander Viktorovich of RSREU for his constant attention to this work, to prof. Mironov Valentin Vasilyevich from RSREU for his hospitality and beneficial discussions, and to Zavolokin Alexander Ivanovich from RSREU for his many valuable comments and helpful discussions.

References

1. A.V. Prutzkow, *Cloud of Science,* **1**, 88 (2014) [Rus]
2. A.V. Prutskov, Automat. Document. and Math. Ling., **45**, 232 (2011)
3. V.V. Mironov and A.K. Rozanov. Inform. of Education and Sci., **25** (2015)
4. A.V. Prutzkow, 2014 International conference on computer technologies in physical and engineering applications (ICCTPEA), p. 157 (2014)
5. https://tech.yandex.ru/mystem/