Towards Czech Morphological Guesser

Pavel Šmerk

Faculty of Informatics, Masaryk University
Botanická 68a, CZ-602 00 Brno, Czech Republic
smerk@mail.muni.cz

Abstract. This paper presents a morphological guesser for Czech based on data from Czech morphological analyzer `ajka` [1]. The idea behind the presented concept lies in a presumption that the new (and therefore unknown to the analyzer) words in a language behave quite regularly and that a description of this regular behaviour can be extracted from the existing data of the morphological analyzer. The paper describes both the construction of guesser data and the architecture of the guesser itself.

1 Introduction

An obvious disadvantage of traditional morphological analyzers is the finiteness of their dictionaries. There is no way to catch *all* words of the particular language in a dictionary of an analyzer, because new and new words continue to appear in the language. Thus, almost allways there will be some words on which the analyzer will not be able to return any information.

If we want to process even these words unrecognized by the analyzer, we have two possibilities. Either to guess possible lemma (or lemmata) and appropriate tags from the context of the word, or to guess it from the form of the word, i. e. from its resemblance to some word known to the analyzer.

In this paper we describe the second of these two possible approaches. First of all, a general idea will be introduced. In the next section, we describe the algorithm for construction of the data as well as the architecture of the guesser in section 4. At the end we make some remarks on possible future development of a guesser.

2 General Idea

The Czech language has many inflectional and derivational paradigms (especially if we count every “exception" as a separate paradigm, as the morphological analyzer `ajka` does), but only a smaller part of them is synchronically productive in the sense that there are appearing (or at least may appear) new words in the language which belong to that productive paradigms.

For instance, word *husa* (“goose") has an own, separate paradigm, because of a doublet in genitive plural, *hus* and *husí*. There is no other word with the same inflectional paradigm and no chance that any such word could appear in

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2008, pp. 1–4, 2008.

Czech. All new feminines which end with vocal *-a* will belong to the paradigm *žena* (“woman").

Of course, it is similar for the derivational relations. For instance, some older adjectives with a meaning “made of material/substance" are created with suffix *-en(ý)*, e. g. *dřevěný* (“wooden”). But nowadays this suffix is not productive and this meaning is expressed by productive suffix *-ov(ý)*.

Obviously, the non-productive paradigms are quite useless for the guessing the possible lemma and tags, or even more than that: these paradigms might serve as an undesirable bias if we would not ignore them. Unfortunately, we do not have the information, which paradigms are productive and which are not.

As was said above, we may assume that all Czech words unrecognized by the analyzer are regular, i. e. belong to some productive paradigm. Or, in other words, that the words with irregular behaviour are either already known to the analyzer or they are quite infrequent in real texts and thus irrelevant for our task. Moreover, we may assume that absolute majority of the actually used lexicon is covered by the dictionary of the analyzer, which means that there is enough examples of all productive behaviour in that dictionary. Then we may conclude that “productive" significantly correlates with “frequent in existing data" and our goal will be to sort out the frequent behaviour from the analyzer’s dictionary.

3 Construction of the Guesser Data

In this section, the algorithm of the guesser data construction will be described in detail.

- First of all, we count numbers of lemmata for each paradigm from the dictionary of Czech morphological analyzer `ajka`,
- then we pick out all lemmata (from that dictionary) which belong to any paradigm which has at least 20 lemmata (this will reduce the number of lemmata from 389,831 to 382,994, i. e. by 1.75 %, but the number of paradigms from 1,830 to 387, i. e. by 78.85 %),
- we let the morphology analyzer generate all word forms for each lemma picked out in the previous step, if the lemma belongs to some of the open POS categories (i. e. nouns, adjectives, verbs or adverbs — other POS categories are closed, so we need not expect any new words which would belong to them),
- moreover we discard all colloquial word forms, all negations and superlatives and all word forms of length 6 letters or less,
- each word form we turn into a triplet word form, lemma and tag in the following form: `akzeřuo:ka:ek:k1gMnSc2,k1gMnSc4` where the `akzeřuo` is the reversed (letter by letter) word form *ouřezka*, the next two strings `ka` and `ek` specifies a lemma (one has to remove the first string from the end of the word form and append the second string, i. e. lemma in the example is *ouřezek*) and the last is a list of possible morphological tags,

– we sort the triplets lexicographically (the actual sort order is not important),
– then we go through the list of triplets and collect information on possible word forms' ends of various length and corresponding lemmata and tags:
  - we use two arrays, both of length 10. The elements of the first array represent letters from the end of the processed word form so that each left "subarray" represents the end of the word form of the given length. The elements of the second array are hashes, in which the keys are triplets without the word form (e. g. `ka:ek:k1gMnSc2,k1gMnSc4`) and values indicates for how many word forms (from the list of triplets) with the given end the analyzer returns this lemma and tag(s),
  - it would be difficult to describe the following steps in general, that is why we illustrate it on example,
  - let us suppose that after the processing of the triplet `akzeřuo:ka:ek:k1gMnSc2,k1gMnSc4`, the content of the first array is

1	2	3	4	5	6	7	8	9	10
a	k	z	e	-	-	-	-	-	-

    (which means that the first hash from the second array stores possible interpretations [lemma and tag(s)] of words which end with *-a*, the second hash is information on words which end with *-ka* etc.),
  - and that the next two triplets are `akzeřýv:ka:ek:k1gMnSc2,k1gMnSc4` and `akzjapš:::k1gFnSc1`,
  - we assume that at least the first (or the last, in reversed string) three letters represent the root of the word form, that is, none of these letters are part of any derivational suffix of flectional ending and thus we can (and should) ignore them. In addition, we assume, that eventual combinations of suffixes and ending longer than 10 letters are not interesting for the guesser,
  - it follows that in our example we ignore *ouř*, *výř* and *špa* and take only the ends *ezka* (*akze*) and *jzka* (*akzj*),
  - for each of such ends we compare its letters with the letters in the first array. We do it from the highest index (10) to the lowest (1). There are three possibilities[1]:
    * both positions are empty ("-" signs emptiness). We do nothing,
    * both positions are nonempty, but equal. We take the rest of the triplet (e. g. `ka:ek:k1gMnSc2,k1gMnSc4`) as a key in the corresponding hash in the second array and increase the associated value,
    * positions are different. We empty the corresponding hash and replace the content of the array element with the letter from the end of the word. Before emptying the hash, we store the information it contains, but only if total sum of values exceeds 20 and at least one value exceeds 10. In such case we generate a string by joining the following data with some separator: given end of the word form, total sum of values and for each key its value and the key itself.

[1] In fact, things are even a little bit more complicated.

Moreover, all numbers, which we store in the generated string, has to be subtracted from the corresponding values in all hashes with lower index than the current hash has,
  - back to our example: the word end *akze* (word form *výřezka*) is the same as the letters in the first array, so we only update the first four hashes by adding 1 to the value associated with key `ka:ek:k1gMnSc2,k1gMnSc4`. But the next word end *akzj* (word form *špajzka*) differs at the fourth position. Therefore we empty the hash, generate something like `akze 50 30:ka:ek:k1gMnSc2,k1gMnSc4 20:::k1gFnSc1` (just an example, not the real data) and put *j* into the fourth element of the first array,
- we sort the generated strings lexicographically, do some optimizations (merge some very similar data etc.), strip the numbers, reverse the sort order and the data for the guesser are done.

4 Guesser Architecture

The guesser itself is rather simple. During the start, it reads the data and creates a regular expression from word ends and a hash, in which the keys are the word ends and values are the information about possible lemmata and tags. Words, which are not recognized by the morphological analyzer, are reversed and their last three letters are cut off. The rest is matched against the regular expression, the eventual match (which is always the longest possible one, because of reversed sort order of the data) is found in the hash and the corresponding value is sent to the output.

5 Future Development

The main problem of this approach is that it works well only for word forms which have some derivational suffixes or inflectional endings, i. e. which are integrated into the Czech language system. It will not success in guessing indeclinable words such as expressions from other languages, foreign proper names etc. It seems that proper handling of these words will require some contextual information.

Acknowledgements. This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET200610406, by the Ministry of Education of CR within the Center of Computational Linguistics LC536, by the Czech Science Foundation under the project GA407/07/0679, and in the National Research Programme II project 2C06009.

References

1. Radek Sedláček and Pavel Smrž. 2001. *A New Czech Morphological Analyser* `ajka`. In Proceedings of the 4th International Conference TSD 2001. LNCS 2166, Springer-Verlag, pp. 100–107.