A Disambiguator for Pymorphy2 Morphological Analyzer

Nicolás Cortegoso Vissio[a], Victor Zakharov[a]

[a] *St Petersburg State University, 9 Universitetskaya Emb., Saint-Petersburg, 195299, Russia*

Abstract

Pymorphy2 is a morphological analyzer implemented in Python for Russian. The parser takes a word and, based on its morphology, produces a series of classification hypotheses regarding class, gender, number, case, etc. However, the analysis of the isolated word rarely occurs without any ambiguity. This article presents an implementation of a trigram tag model that works on top of the morphological parsing performed by pymorphy2 and uses the sequence of words in the sentence to choose the most probable morphological interpretation for each word.

Keywords

Russian language, pymorphy2, NLP, tagger, part-of-speech, morphological disambiguation

1. Introduction

Part-of-speech tagging for Russian language is a well-researched field and many taggers have been developed applying different approaches. When it comes to working with NLP, Python is arguably the most widely used programming language today and pymorphy2, a popular choice for performing the morphological analysis of Russian words. According to the official documentation [1] pymorphy2 is capable of:

- transforming a word to its dictionary form (lemma), for example, "люди → человек", or "гулял -> гулять";
- converting a word to the desired form, for example, change its grammatical case, put it in plural, etc;
- providing grammatical information about a word (number, gender, case, part-of-speech, etc.).

To parse a word form, pymorphy2 relies on modified version of the dictionary of the OpenCorpora project [2] that was optimized for speed and memory saving. The OpenCorpora dictionary is structured around lexemes. A lexeme consists of all the forms of a word and the labels with the grammatical information, where the first word form in the list corresponds to its dictionary form. For example, the lexeme for "ёж" (hedgehog) looks like figure 1:

```
ёж      NOUN,anim,masc sing,nomn
ежа     NOUN,anim,masc sing,gent
ежу     NOUN,anim,masc sing,datv
...
ежами   NOUN,anim,masc plur,ablt
ежах    NOUN,anim,masc plur,loct
```

Figure 1: Example of a lexeme [3]

If the word form does not exist in the dictionary, pymorphy2 conducts a predictive analysis on the unknown word, identifying suffixes, prefixes and applying other strategies that could provide a criterion to classify it.

EMAIL: st082534@student.spbu.ru (A. 1); v.zakharov@spbu.ru (A. 2)
ORCID: 0000-0003-1683-7270 (A. 1); 0000-0003-0522-7469 (A. 2)

Taken separately, Russian word forms often allow for more than one grammatical interpretation.[2] Since pymorphy2 parses one word at a time, it returns a list of possible parses with an associated probability. When working with bag-of-words models, a common practice is to simply select for each word form the parse with the highest probability. In this case, assuming only the dictionary form of the word is required, pymorphy2 will get the correct part-of-speech about 92% of the time. However, if the grammatical case of the word is taken in consideration, the precision will drop to 82% (see the experiment's results in section 6) and the parsing will often produce ungrammatical sequences.

In order to improve the percentages shown in the previous paragraph, the proposal presented in this article is to implement a trigram part-of-speech model to disambiguate the morphological analysis that pymorphy2 performs on the isolated word. The part-of-speech tags of those word classes that have declensions are augmented with the grammatical case, while other features such as number or gender are discarded in order to keep the trigram model at a reasonable size and prevent data sparseness when training it on a rather small corpus.

Since this paper proposes a method to disambiguate pymorphy2 parsing results, it will focus exclusively on this morphological analyzer and measure its performance before and after the suggested extension. It does not intend to be a superior solution to other morphological analyzers available for Russian, but rather a helper tool for pymorphy2 users. For state-of-the-art taggers or comparisons between the performance of pymorphy2 and other available options (mystem3, TreeTagger, FreeLing, etc.), the reader is referred to the work of Kuzmenko [4] or Kotelnikov et al. [5].

The remainder of the paper will cover: 2. A pymorphy2's parsing example, 3. Trigram hidden Markov Model, 4. Training the trigram model, 5. Code implementation example, 6. Testing the model and 7. Conclusions and further work.

2. A pymorphy2's parsing example

As noted in the introduction, pymorphy2 processes each word separately and returns one or more "Parse" objects containing the possible parses for the given word form. For example, the morphological analysis of the word forms "мама", "мыла", "раму" produces the following lists (1), (2), (3):

```
[
  Parse(word='мама', tag=OpencorporaTag('NOUN,anim,femn sing,nomn'), normal_form='мама', score=1.0,
methods_stack=((<DictionaryAnalyzer>, 'мама', 1907, 0),))                                              (1)
]
```

```
[
  Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut sing,gent'), normal_form='мыло', score=0.333333,
methods_stack=((<DictionaryAnalyzer>, 'мыла', 54, 1),)),
  Parse(word='мыла', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'), normal_form='мыть',
score=0.333333, methods_stack=((<DictionaryAnalyzer>, 'мыла', 1813, 8),)),
  Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut plur,nomn'), normal_form='мыло', score=0.166666,   (2)
methods_stack=((<DictionaryAnalyzer>, 'мыла', 54, 6),)),
  Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut plur,accs'), normal_form='мыло', score=0.166666,
methods_stack=((<DictionaryAnalyzer>, 'мыла', 54, 9),))
]
```

```
[
  Parse(word='раму', tag=OpencorporaTag('NOUN,inan,masc,Geox sing,datv'), normal_form='рам', score=0.5,
methods_stack=((<DictionaryAnalyzer>, 'раму', 32, 2),)),
  Parse(word='раму', tag=OpencorporaTag('NOUN,inan,femn sing,accs'), normal_form='рама', score=0.5,       (3)
methods_stack=((<DictionaryAnalyzer>, 'раму', 55, 3),))
]
```

With the exception of "мама", the other word forms have more than one possible interpretation. In the case of the noun "раму" the ambiguity arises in gender and case, while "мыла" can be analysed as

2 For example, the nominal and accusative plural cases endings for nouns like "сталь" (declension type 8a according to A. A. Zaliznyak's classification) are identical for the singular cases of the genitive, dative and locative; most of the singular feminine adjectives in the genitive, dative, locative and instrumental cases share the same inflection; a word form can even belong to different word classes, like "мыла", that can be analyzed as a noun or a verb.

a verb or noun. Every parse object inside the list has a parameter "score" with an associated probability. Korobov [3] states that the score corresponds to the conditional probability *p (analysis | word)* estimated on the basis of the OpenCorpora corpus. This is obtained by counting how many times a certain analysis has been associated with a given word form and, based on these frequencies, its conditional probability is calculated using Laplace smoothing. The parse objects within the list are sorted according to this probability in descending order, therefore picking the first item in the list is equivalent to selecting the parse object with the most probable interpretation for the given word form. For example, the parse objects with the highest score for each of the word forms analysed in (1), (2) and (3) would be:

```
Parse(word='мама', tag=OpencorporaTag('NOUN,anim,femn sing,nomn'), normal_form='мама', score=1.0,
methods_stack=((<DictionaryAnalyzer>, 'мама', 1907, 0),))                                          (4)
```

```
Parse(word='мыла', tag=OpencorporaTag('NOUN,inan,neut sing,gent'), normal_form='мыло', score=0.333333,
methods_stack=((<DictionaryAnalyzer>, 'мыла', 54, 1),))                                            (5)
```

```
Parse(word='раму', tag=OpencorporaTag('NOUN,inan,masc,Geox sing,datv'), normal_form='рам', score=0.5,
methods_stack=((<DictionaryAnalyzer>, 'раму', 32, 2),))                                            (6)
```

If "мама", "мыла", "раму" are no longer treated as separate tokens, but as words in the sentence "мама мыла раму", the parse object with the highest score incorrectly classifies the last two. When working with bag-of-words models and lemmas, the misclassification in case is usually not harmful (most of the time it will still provide the right dictionary form)[3], but a wrong part-of-speech attribution will produce a different interpretation of the lemma. The next section suggests how the score values can be combined with the trigram tag model to obtain better results.

3. Trigram hidden Markov model

Hidden Markov models (HMM) are probabilistic sequence classifiers that have been widely used in NLP tasks like part-of-speech tagging and word class disambiguation. The task of the model is to find for any string of word forms of the set Ψ (the observable states) the most probable sequence of part-of-speech tags of the set Ω (the hidden states). For a better understanding of HMM the reader is referred to Jurafsky [6] or Bocharov et al. [7]. Although, nowadays POS-taggers are build using more advanced techniques, for example those based on neural networks, for the purpose envisaged here of eliminating the ambiguity of the analysis previously carried out by pymorphy2, a modified HMM model would be an easy solution to implement. The HMM is briefly described below along with the intended modification to disambiguate the analysis from pymorphy2.

To train an HMM model, it is necessary to calculate two parameters in a tagged corpus: the emission and the transition probabilities.

The emission probabilities

$$p(w_i \vee t_i) \tag{7}$$

where p is the conditional probability that the word w_i corresponds to the tag t_i. This assumes that the probability of an output observation w_i depends only on the state that produced the observation t_i and not on any other states or observations.

The transition probabilities

$$p(t_t \vee t_{i-1}, t_{i-2}) \tag{8}$$

where p is the probability that the tag t_i occurs, provided that is preceded by the tags t_{i-1} and t_{i-2}. This assumes that the probability of a particular tag depends only on the previous two tags (trigram).

3 Here the missclassification of “раму” produces a different dictionary form.

The HMM tagging algorithm chooses as the most likely tag sequence the one that maximizes the product of two terms: the probability of the sequence of tags (the transition probability) and the probability of each tag generating a word (the emission probability).

Trigram HMM

$$argmaxp(x_1, \dots, x_n, y_1, \dots y_{n+1}) \approx argmax \prod_{i=1}^{n+1} p\,(y_i \vee y_{i-1}, y_{i-2}) \prod_{i=1}^{n} p\,(x_i \vee y_i) \qquad (9)$$

where $p(y_i|\ y_{i-1},\ y_{i-2})$ is the transition probability and $p(x_i|y_i)$ is the emission probability.

The approach taken here is to implement the tagging algorithm on top of the pymorphy2 parsing results and use the score values from the Parse object as the emission probabilities.

Modified trigram HMM

$$argmaxp(x_1, \dots, x_n, y_1, \dots y_{n+1}) \approx argmax \prod_{i=1}^{n+1} p\,(y_i \vee y_{i-1}, y_{i-2}) \prod_{i=1}^{n} p\,(y_i \vee x_i) \qquad (10)$$

where the last term of the equation, the emission probability of a word given a tag, is replaced by the score value from the pymorphy2 parse: the probability of the analysis given a word.

4. Training the trigram POS tag model

Pymorphy2 not only implements the OpenCorpora dictionary, it also adopts the same set of tags. To take advantage of this fact, the model was trained on the OpenCorpora labeled subcorpus with homonyms removed [8] (26011 sentences, 256311 tokens, 188319 words), so no modifications on the tags were required. As mentioned in the previous section, the emission probabilities are directly replaced by the score values from the parse objects. Therefore, only the transition probabilities in the corpus were calculated on the basis of the following 49 tags, which were obtained by combining the 20 basic part-of-speech tags from pymorphy2/OpenCorpora and the grammatical case (where applicable):

Table 1
Part-of-speech tags without case declension

part-of-speech	tag
adverb	ADVB
comparative	COMP
conjunction	CONJ
gerund	GRND
infinitive	INFN
interjection	INTJ
particle	PRCL
predicative	PRED
preposition	PREP
verb	VERB
short form adjective	ADJS
short form participle	PRTS

As table 2 shows, those part-of-speech that have declensions, were augmented with the grammatical case. Gender and number were not taken into account to keep the tag set within reasonable limits. It was assumed that case is a good predictor to be included in the transition probabilities (although this hypothesis remains to be proven). Table 4 presents a fragment of the resulting matrix with the transition probabilities. The cells contain the conditional probability for the tag in the row when it is preceded by the two tags from the columns. For example, the probability that an adjective in the nominative case appears at the very beginning of the sentence is 0.0731339900. Those combinations that were not observed in the corpus were assigned a very small probability of 0.000000001; for instance an adjective in any other case than is not instrumental after an adjective in the instrumental case at the beginning of the sentence. The transition probabilities were stored in a JSON-format file.

Table 2
Part-of-speech tags with case declension

case	noun	adjective	participle	pronoun	numeral
nominative	NOUN nomn	ADJF nomn	PRTF nomn	NPRO nomn	NUMB nomn
genitive	NOUN gent	ADJF gent	PRTF gent	NPRO gent	NUMB gent
accusative	NOUN accs	ADJF accs	PRTF accs	NPRO accs	NUMB accs
dative	NOUN datv	ADJF datv	PRTF datv	NPRO datv	NUMB datv
locative	NOUN loct	ADJF loct	PRTF loct	NPRO loct	NUMB loct
instrumental	NOUN ablt	ADJF ablt	PRTF ablt	NPRO ablt	NUMB ablt
vocative	NOUN voct				
2nd genitive	NOUN gen2				
2nd locative	NOUN loc2				

Table 3
Other tags

part-of-speech	tag
Latin word or character	LATN
roman number	ROMN
unknown class	UNKN

Table 4
Fragment of the matrix with the transition probabilities

	<*>_<S>[4]	<S>_ADJF ablt	<S>_ADJF accs	<S>_ADJF datv	...
<E>[5]	0.006326730	0.05263158	0.04285714	0.052631589	...
ADJF ablt	0.007155230	0.09473684	0.000000001	0.000000001	...
ADJF accs	0.005272275	0.000000001	0.1	0.000000001	...
ADJF datv	0.001431046	0.000000001	0.000000001	0.052631589	...
ADJF gent	0.003765910	0.000000001	0.000000001	0.000000001	...
ADJF loct	0.000150636	0.000000001	0.000000001	0.000000001	...
ADJF nomn	0.073133990	0.000000001	0.000000001	0.000000001	...
...	...	...	...	...	...

5. Code implementation example

The code written in python[6] implements the Viterbi algorithm to find the most probable sequence of parse objects from the morphological analysis performed by pymorphy2. It takes as input the JSON file with the transition probabilities and a list that contains all the possible parse objects that pymorphy2 returns for each word. The output is a new list with only one parse object per word. The code and the file with the transition probabilities are available for download from a GitHub repository [9].

4 Symbol for "start of sentence".
5 Symbol for "end of sentence".
6 System requirements: python3, pymorphy2 version 0.8.

Code implementation example

```
from pymorphy2 import MorphAnalyzer #1

from hmmtrigram import MostProbableTagSequence #2

morph = MorphAnalyzer() #3

token_list = ['Мама', 'мыла', 'Раму', '.'] #4                                    (11)

pymorphy2_parsed = [morph.parse(token) for token in token_list] #5

mpts = MostProbableTagSequence('transition_probabilities.json') #6

mpts.get_sequence(pymorphy2_parsed) #7
```

#1: imports the class for morphological analysis from the package "pymorphy2"
#2: imports the class for calculating the most probable tag sequence from "hmmtrigram"
#3: instantiates the object "morph" from the class "MorphAnalyzer"
#4: any list of tokens
#5: the "parse" method of the "morph" object parses each token in the list and stores the parsing results in the new list "pymorphy2_parsed"
#6: instantiates the object "mpts" from the class "MostProbableTagSequence" with the name of the json file that contains the transition probabilities as argument
#7: the "get_sequence" method of the "mpts" object takes the list with the parse objects stored in "pymorphy2_parsed" and returns a new list with the most probable parsing for the sequence of tokens.

Code output

```
[
Parse(word='мама', tag=OpencorporaTag('NOUN,anim,femn sing,nomn'),
normal_form='мама', score=1.0, methods_stack=((<DictionaryAnalyzer>, 'мама', 1907,
0),)),
Parse(word='мыла', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'),
normal_form='мыть', score=0.333333, methods_stack=((<DictionaryAnalyzer>, 'мыла',
1813, 8),)),                                                                     (12)
Parse(word='раму', tag=OpencorporaTag('NOUN,inan,femn sing,accs'),
normal_form='рама', score=0.5, methods_stack=((<DictionaryAnalyzer>, 'раму', 55,
3),)),
Parse(word='.', tag=OpencorporaTag('PNCT'), normal_form='.', score=1.0,
methods_stack=((<PunctuationAnalyzer>, '.'),))
]
```

The most probable sequence correctly disambiguates 'мама' as a noun in the nominative case, 'мыла' as verb and 'раму' as a noun in the accusative case.

6. Testing the model

The test was conducted on the OpenCorpora subcorpus without homonyms and unknown words of 10966 sentences, 72671 tokens and 50433 words. The experiment presents the results for 55 275 tokens (no punctuation marks).

The baseline summarizes the results of selecting the parse object with the highest score for each independent word form (the first item in the list that pymorphy2 generates).

Table 5
Pymorphy2 part-of-speech tagging results (most probable parse object)

method	precision	recall
baseline (simple tags)	0.92	0.91

Table 5 shows the averaged precision and recall for the 20 part-of-speech tags (the base POS tags without case). This outlines the performance that can be expected of getting the right part-of-speech

(POS) tag for a given word form. When grammatical case is also taken into account (augmented tags), these values drop significantly (compare with the baseline in table 6).

Table 6
Baseline and trigram model extension using augmented tags

method	precision	recall
baseline (augmented POS tags)	0.82	0.80
trigram model	0.94	0.89

Table 6 contrasts the averaged precision and recall for the 49 augmented tags (part-of-speech + case) for the baseline with those for the implementation of the trigram model to choose between the parse objects returned by pymorphy2.

Table 7
Long form adjectives with case for the baseline and the trigram model

POS tags + case	F-score (baseline)	F-score (3-gram model)
ADJF ablt	0.68	0.97
ADJF accs	0.73	0.92
ADJF datv	0.64	0.88
ADJF gent	0.81	0.95
ADJF loct	0.63	0.93
ADJF nomn	0.91	0.96

Table 7 shows the F-score for the long form adjectives + case for the baseline and the trigram model. When considering case declensions, the trigram model improves clearly over the baseline. However, those differences are less extreme for word classes that do not have case declensions (see table 8).

Table 8
POS tags without case declension for the baseline and the trigram model

POS tags	F-score (baseline)	F-score (3-gram model)
ADJS (short adjectives)	0.90	0.98
ADVB (adverbs)	0.94	0.96
CONJ (conjunctions)	0.91	0.90
INTJ (interjections)	0.90	0.88

Conjunctions and interjections constitute the two cases within the 49 tags where the implementation of the trigram model performs slightly below the baseline.

7. Conclusions and further work

The testing results support the benefit of applying the trigram model to disambiguate the analysis preformed by pymorphy2. However, if pymorphy2 is being used only to get the part-of-speech or the dictionary form of the words within a bag-of-words model, the suggested extension will produce little improvement over the baseline. For that purpose, selecting the first parse object in the list will be enough. The implementation of the trigram model to choose the most probable sequence of parse objects is useful for those tasks that require valid POS tag sequences or greater precision in determining the grammatical case of a word. In Russian, by disambiguating the grammatical case of a word form, most of the time its gender and number are also obtained correctly. The effects of broadening the combination of part of speech + case with respect to number and / or gender remain to be explored.

8. References

[1] Documentation. Morphological analyzer pymorphy2. URL: https://pymorphy2.readthedocs.io/en/stable/user/index.html
[2] OpenCorpora. URL: http://opencorpora.org/
[3] M. Korobov, Morphological Analyzer and Generator for Russian and Ukranian Languages, in: Analysis of Images, Social Networks and Texts, 2015, pp. 320-332.
[4] E. Kuzmenko, Morphological Analysis for Russian: Integration and Comparison of Taggers, in: D. Ignatov et al. (eds) Analysis of Images, Social Networks and Texts. AIST 2016. Communications in Computer and Information Science, vol. 661, Springer, Cham, 2017.
[5] E. Kotelnikov, E. Razova and I. Fishcheva, A Close Look at Russian Morphological Parsers: Which One Is the Best? in: Communications in Computer and Information Science, 2018.
[6] D. Jurafsky and J. Martin, Speech and language processing: An introduction to natural language processing, computional linguistics, and speech recognition, 2nd. ed. Pearson, Upper Saddle River, New Jersey, 2009, pp. 139-151.
[7] V. Bocharov and O. Mitrenina, Computer morphology, in: I. Nikolaev, O. Mitrenina and T. Lando (eds), 2nd. ed., URSS, Saint Petersburg, 2017, pp. 14–33.
[8] Downloads. OpenCorpus. URL: http://opencorpora.org/?page=downloads
[9] N. Cortegoso-Vissio. Github repository. URL: https://github.com/nicolascortegoso/morphological_analyzer_for_russian