Building NLP Pipeline for Russian with a Handful of Linguistic Knowledge

Nikita Medyankin and Kira Droganova

School of Linguistics, Faculty of Humanities,
Higher School of Economics, Moscow, Russia
nikita.medyankin@gmail.com,kira.droganova@gmail.com
https://www.hse.ru/en/

Abstract. This work addresses the issue of building a free NLP pipeline for processing Russian texts all the way from plain text to morphologically and syntactically annotated conll. The pipeline is written in python3. Segmentation is provided by our own module. Mystem with numerous postprocessing fixes is used for lemmatization and morphology tagging. Finally, syntactical annotation is obtained with MaltParser utilizing our own model trained on Syntagrus, which was converted into conll format for this purpose, with morphological tagset being converted into Mystem/NRC tagset with numerous special fixes.

Keywords: natural language processing, dependency parsing, syntactic relations for Russian

1 Introduction

This work builds upon the works of J. Nivre et al. [1] dedicated to training MaltParser models on Syntagrus, and the research conducted by S. Sharoff in 2011, [2] which addressed an issue of building assorted NLP tools for Russian language by utilising fully statistical approach. Our initial goal was more modest and at the same time more ambitious as we were set to build a reasonably accurate NLP pipeline including all stages from text segmentation to syntactic annotation, resorting to whatever tools looked reasonable to use at each stage. The point was that the pipeline in question should ideally be designed so that any person lacking advanced programming skills would be able to use it. In the process, we created our own rule-based segmentation module, mobilized Mystem [3] to serve as both tokenizer and morphology tagger, created post-mystem correction module to improve the results of Mystem doing its job, and trained a bunch of models for MaltParser building upon research conducted by Kira Droganova [4]. Fig. 1 provides the general scheme of the resulting pipeline.

The article structure is as follows. Sections 2—4 cover segmentation, morphology, and syntax modules respectively, with basic outline of workflow, results regarding accuracy, and most prominent errors for each module. Section 5 is dedicated to overall pipeline quality. Section 6 addresses web interface and source code. Section 7 contains conclusions and further plans.

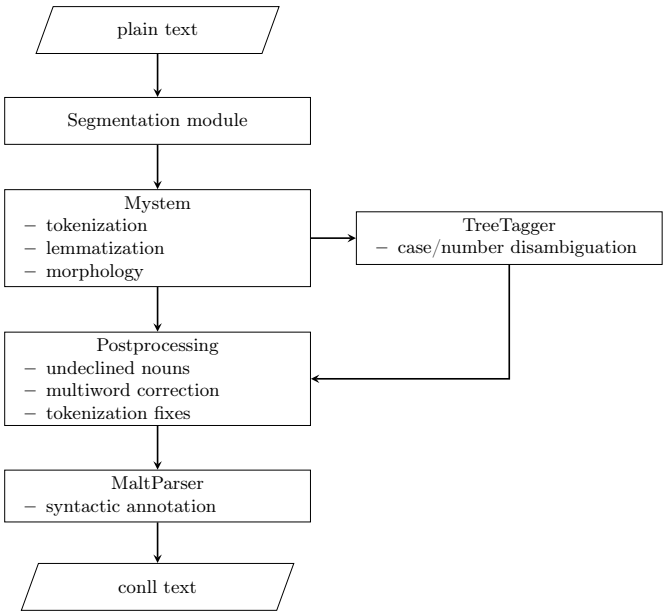

Fig. 1. Pipeline workflow

2 Segmentation

MaltParser operates on a sentence-to-sentence basis. Therefore, correct segmentation is crucial for correct syntactic parsing. For this task we designed our own rule-based segmentation module, which was written in python3 with rule patterns determined using perl-style regular expressions. Since MaltParser model is trained on Syntagrus, we kept in mind that segmentation rules should mostly stick to what is prominent for Syntagrus, yielding us, e.g., semicolons as ends of sentences.

Segmentation module workflow is as follows:

1. Detect all potential ends of sentences:
   - Always considered end of sentence:
     - semicolon;
     - colon followed by dash;
     - line break.
   - Considered candidate end of sentence:
     - Any number of dots, question marks, and exclamation marks in any combination followed by a capital letter.
2. Override special cases:
   - Abbreviation patterns:
     - [.,]

  - [т.е] — any lower-case/upper-case letters with dot in between.
  - [т. е] — any lowercase letter after dot and whitespace.
  - [П. И. Чайковский], [Чайковский П.И.] — dot preceded by a single uppercase letter.
- Quoted speech and explanation patterns involving quotations and parentheses: *«Прекрати!» — воскликнул Геннадий. / "Stop it!" — Gennady exclaimed.*

Accuracy was measured manually on 1,000 sentences designed specifically for testing of segmentation modules, and proved 99.5%. The test set was provided by O. Lyashevskaya.

The most typical patterns that may cause wrong segmentation are as follows:

- addresses or amounts of money: 25 *руб.* 33 *коп.*
- emoticon and emoji patterns.

3 Morphology

Morphology module as a whole works the following way:

1. Mystem [3] provides the bulk of segmentation, morphology tagging, and lexical disambiguation.
2. Undeclined nouns and abbreviations are detected and given special 'nonflex' tag in place of case and number.
3. TreeTagger [5] model trained by A. Fenogenova et al. [6] is used for case and number disambiguation for nouns and adjectives.
4. Multiword expressions are fixed using a dictionary obtained from Syntagrus.
5. A number of minor fixes mainly concerning correcting tokenization of punctuation marks is applied.

3.1 Undeclined Nouns and Abbreviations

Undeclined nouns such as *кофе / coffee* are given special treatment because being what they are they do not really have case and number based upon their word form. In fact, their case and number can only be somewhat determined from their relations with other words in the sentence. Since we wanted to base our syntactical annotation on morphology and not vice versa, the decision was made to consider case and number of undeclined nouns undetermined (using special 'nonflex' tag in place of case and number). The same goes for abbreviations such as *СССР / USSR, в.* (*век / century*) or *П.* (*Петрович*) / *P.* (*Petrovich*) in a way that Mystem does not reconstruct the full form for the abbreviation, thus basically rendering them as undeclined nouns.

A noun is considered undeclined using a simple heuristic. If Mystem produces 12 or more possible annotations for one word, it is 'nonflex' for both case and number. If the number of annotations is less than 12 but more than 6 for either plural or singular number, it is 'nonflex' for case but not for number.

3.2 Resolving Morphological Ambiguity

The issue with Mystem is that it only comes equipped with lexical disambiguation feature, but not with morphological one, e.g., it can perfectly tell preposition *в / in, into* from abbreviation *в* (*век / century*) (though it does not reconstruct the original lemma for abbreviation), but is not designed to choose the correct case for noun from homonymous forms, e.g., 'S inan nom sg' vs. 'S inan acc sg' for *труп / corpse.*

To address this problem, we use TreeTagger with parameter file trained by A. Fenogenova et al. [5] on disambiguated part of RNC to choose morphological annotation for nouns and adjectives from those provided by Mystem. This fix provides roughly 5% increase in morphological annotation accuracy, as opposed to just using the first morphological annotation available from Mystem.

3.3 Fixing Multiword Expressions

We also fix multiword expressions, e.g., *как бы то ни было.* Mystem generally does not recognize them as such therefore lemmatizing and annotating each word separately. To resolve this issue, we had extracted a list of frequent multiword tokens from Syntagrus and created a dictionary. During postprocessing, the tokens are stacked up and given morphological annotation according to this dictionary. Honestly saying, that feature somewhat backfired on us while testing the pipeline on Syntagrus as not all instances of such multiword expressions are even annotated as multiword expressions in Syntagrus itself yielding a slight drop in accuracy.

A number of minor fixes mainly concerning correcting the tokenization of punctuation marks is also applied during postprocessing.

3.4 Accuracy and Common Mismatches

Common mismatches and accuracy scores for morphological tagging were measured for the partial pipeline going from plain text to morphologically annotated conll. were obtained on the joint development and final test parts of Syntagrus (see section 4.1 below) and converted to plain text. Strict accuracy, i.e., only complete match of annotations for a token is a hit, is 88%. Part of speech accuracy reached 97%. Most common mismatches extracted automatically using python3 script are shown in Table 2. The first column shows the percentage of specific mismatch among all tokens, the second shows the percentage of specific mismatch among all mismatches. As can be clearly seen, the most prominent mismatches can be divided into four distinctive groups:

1. Wrong case for nouns and adjectives.
2. Brevis adjective mistaken for adverb, e.g., *нужно, должно, известно, трудно, необходимо.*
3. Conjunction instead of particle, with the worst offender being *и / and.*
4. Conjunction instead of adverb, e.g., *однако, как, пока, когда.*

The third and the fourth columns refer to how particular mismatch in morphological annotation directly affects syntactical labelling. LES and UES are exact opposites of LAS and UAS (with E standing for *error*) meaning the third column shows how many tokens with particular morphology error are labeled with wrong head and/or relation type, and the fourth column shows how many tokens with that morphology error are labeled with wrong head. For example, it can be figured that wrong case for adjective is relatively harmless for syntactical labelling, wrong case for noun is much worse, and erroneous CONJ instead of PART or ADV is the most severe. Of course, wrong morphology tagging should also have indirect impact on syntax, i.e., tokens with correctly detected morphology might be affected by their less fortunate peers, but this effect is much harder to measure.

Fig. 2. Most Common Morphology Mismatches

% **tok**	% **err**	**LES**	**UES**	**tagged as**	**in syntagrus**
0.49%	4.04%	98.30%	76.74%	CONJ	PART
0.30%	2.45%	78.92%	40.05%	S m inan nom sg	S m inan acc sg
0.30%	2.44%	66.82%	58.59%	ADV -	A - - sg brev n -
0.29%	2.35%	86.59%	35.37%	S m inan acc sg	S m inan nom sg
0.26%	2.13%	95.15%	91.11%	CONJ	ADV -
0.23%	1.89%	10.00%	7.88%	A - nom pl plen - -	A - acc pl plen - inan
0.22%	1.80%	85.99%	51.27%	S n inan nom sg	S n inan acc sg
0.21%	1.72%	15.72%	11.37%	A - nom sg plen m -	A - acc sg plen m inan
0.16%	1.31%	10.53%	8.33%	A - acc sg plen m inan	A - nom sg plen m -
0.15%	1.26%	24.55%	19.55%	A - acc pl plen - inan	A - nom pl plen - -
0.15%	1.23%	5.14%	4.21%	S f inan loc sg	S f inan dat sg
0.14%	1.12%	85.71%	41.84%	S n inan acc sg	S n inan nom sg
0.12%	0.95%	22.89%	18.67%	A - nom sg plen n -	A - acc sg plen n -
0.12%	0.95%	80.00%	43.03%	S m inan nom pl	S m inan acc pl
0.11%	0.86%	75.33%	31.33%	S m inan acc pl	S m inan nom pl
0.10%	0.83%	37.50%	22.22%	S m anim gen sg	S m anim acc sg

4 Syntax

4.1 Training MaltParser Model

Unlike rule-based segmentation module and mish-mash morphology module, syntax module is fully statistical. It utilizes MaltParser with model trained on Syntagrus. For the purposes of fine-tuning training settings and testing, Syntagrus was split into three parts: training set (80%), development test set (10%) and final test set (10%).

A series of experiments was conducted using different types of projective and non-projective algorithms [4]. The most valuable results have been achieved

with pseudo-projective transformations provided by MaltParser functionality and Nivre arc-eager algorithm. Accuracy of the obtained models was measured with MaltEval [8] using Labeled Attachment Score and Unlabeled Attachment Score evaluation metrics.

Current model for MaltParser has labeled attachment score (LAS) of 85.0% and unlabeled attachment score (UAS) of 90.7%, which is an improvement over the best figures reported by Sharoff with them being 83.4% LAS and 89.4% UAS [2]. This was achieved via improved automated conversion of Syntagrus morphological tagset into Mystem/NRC tagset.

4.2 Tagset Conversion

For the purpose of training MaltParser, xml format of Syntagrus was conversed into conll, which is MaltParser operational format. Syntagrus morphological tags [9], which come from ETAP-3 [10], were converted into Mystem/NRC tags [3, 7]. Each annotation was processed the following way:

1. Special fixes were applied (see 4.3 for details).
2. POS-specific tags were detected using regular expressions.
3. Detected tags were replaced by their Mystem/NRC equivalents.
4. Some tags normally omitted in Syntagrus were reinstated (e.g., non-short form for adjectives).
5. The resulting tags were output in regular POS-specific order.

4.3 Special Fixes

A number of Syntagrus peculiarities was fixed during conversion, the most notable being the following:

1. Personal pronouns are treated as nouns in Syntagrus and as such are always treated as having gender and no person due to being a legacy feature from the times when ETAP was French–Russian. They were converted into nominal pronouns having person with gender marked only where appropriate.
2. 3rd person possessive pronouns such as *их / their* are always considered Genitive form of corresponding personal pronouns in Syntagrus, as the forms are homonymous in Russian. They were converted into proper adjectives based on their relation label: quasi-agentive and attributive relations always constitute adjective pronoun, others are personal pronouns.
3. Some frequent adverbs annotated as particles, which indeed can be the case depending on semantics but is relatively rare, were uniformly converted into adverbs.
4. *Which / который* annotated as noun was converted into adjective.
5. Single number numeral *один / one,* which is the only Russian numeral to retain number, is converted into noun for the purpose of not accounting for number for other numerals.
6. Participles are converted into separate part of speech as their syntactical behavior is much closer to adjectives than verbs.

7. Undeclined nouns were detected using the same procedure as in 3.1 and given exactly the same treatment.
8. The same as above goes for abbreviations.

4.4 Common Mismatches

Common mismatches in syntactic labeling for the whole pipeline going from plain text to fully annotated conll were obtained on joint set of development and final test parts of Syntagrus. As can be seen from the Table 3, they are quite usual for MaltParser models trained on Syntagrus. The most prominent types are:

1. Mixed up predicative and 1-completive relations.
2. Mixed up quasi-agentive and 1-completive relations.
3. Mixed up circumstantial and completive relations.
4. Wrong positional number for completive relation.

Errors of the first type exist mostly due to wrong case detection for nouns, specifically 'acc' instead of 'nom' and vice versa, i.e., it is basically the question of subject vs. direct object. Errors of the second type are almost exclusively nouns in Genitive with other noun as head. The distinction is mostly semantic, resulting in confusion for MaltParser. The third type consists mostly of prepositions with verb as head. The distinction is also largely semantic with such relation considered completive if corresponding semantic role is considered part of verb frame, and circumstantial otherwise, e.g., *переезжал на дачу / moved to the villa* vs. *переезжал на лето / moved for summer.* Errors of the fourth type have their roots in the fact that positional number for completive relation in Russian is mostly determined by the semantics of verb frame, and as such is easily messed up by MaltParser and even by annotators of the corpus themselves.

5 Overall Pipeline Quality

Total accuracy of the whole pipeline was also estimated, and though the figures are not shocking high being 76.7% LAS and 84.1% UAS, their increase over the course of development looks promising. It should also be noted that the authors of this work are unaware of any other tool offering NLP pipeline for Russian going from plain text to syntactic annotation working out of the box and at the same time being free to use. The only one that might be comparable is the pipeline put together by Sharoff himself in 2011, but it works out of the box only for Linux-based systems, and we could find no reported results regarding its overall accuracy.

6 Web Interface and Source Code

A deliberately simplistic web interface has been implemented on top of the pipeline, which basically allows the user to upload plain text file in Russian

Fig. 3. Most Common Syntax Mismatches

% **tok**	% **err**	**tagged as**	**in syntargus**
0.55%	7.45%	quasi-agent	1-compl
0.48%	6.44%	1-compl	predic
0.43%	5.74%	1-compl	2-compl
0.38%	5.12%	predic	1-compl
0.37%	4.97%	circ	1-compl
0.32%	4.33%	1-compl	quasi-agent
0.32%	4.26%	circ	2-compl
0.26%	3.51%	1-compl	circ
0.25%	3.41%	2-compl	1-compl
0.22%	3.00%	quasi-agent	attr
0.22%	2.92%	1-compl	attr
0.18%	2.42%	attr	1-compl
0.15%	2.05%	2-compl	circ
0.11%	1.45%	circ	restr
0.09%	1.18%	2-compl	3-compl
0.09%	1.15%	attr	quant

and get it annotated in conll. It is available for unconditional use at `http://web-corpora.net/wsgi3/ru-syntax/`.

Offline version is supplied as a python3 library with command line interface. The source code can be obtained from github at `https://github.com/tiefling-cat/ru-syntax`. In order to use it, one would also need to download our MaltParser model at `http://web-corpora.net/wsgi3/ru-syntax/download`. Other requirements, as well as the detailed installation process, are listed on the github page readme.

7 Conclusions

In this work, we have presented NLP pipeline for Russian going from plain text to morphologically and syntactically annotated conll that does not require any specific technical knowledge to use, is free to use and (mostly) open-source.

Concerning future research and development, the following directions might be considered. First, segmentation module calls for more tests. Second, conjunction vs. particle and conjunction vs. adverb problem should be addressed due to their dramatic impact on the quality of syntactical annotation.

8 Acknowledgements

The authors are grateful for computational capabilities provided by Andrey Kutuzov and mail.ru group and appreciate support and feedbacks from Olga Lyashevskaya from School of Linguistics, Faculty of Humanities, Higher School of Economics, Moscow.

References

1. Nivre J., Hall J., Nilsson J., Chanev A., Eryigit G., Kübler S., Marinov S., Marsi E.: MaltParser: A language independent system for data-driven dependency parsing. In: Natural Language Engineering, Vol. 13, pp. 95–135 (2007)
2. Sharoff S., Nivre J.: The proper place of men and machines in language technology. Processing russian without any linguistic knowledge. In: Computational Linguistics and Intelligent Technologies. Proceedings of the International Workshop "Dialogue 2011". Vol. 10 (17), pp. 657–670. RGGU, Moscow (2011)
3. Segalovich I.: A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine, MLMTA-2003, `https://tech.yandex.ru/mystem/`
4. Droganova K.: Building a Dependency Parsing Model for Russian with MaltParser and MyStem Tagset. In: Proceedings of the AINL-ISMW FRUCT, Saint- Petersburg, Russia, 9–14 November 2015, ITMO University, FRUCT Oy, Finland (2015)
5. Schmid H.: Probabilistic Part-of-speech Tagging Using Decision Trees. In: International Conference on New Methods in Language Processing (1994)
6. Fenogenova A., Kayutenko D., Dereza O.: Mystem+, `http://web-corpora.net/wsgi/mystemplus.wsgi/mystemplus/`
7. Russian National Corpus, `http://www.ruscorpora.ru/en/index.html`
8. Nilsson J.: User Guide for MaltEval 1.0 (beta), `http://www.maltparser.org/malteval.html` (2014)
9. Syntagrus Instruction, `http://www.ruscorpora.ru/instruction-syntax.html`
10. Iomdin L., Petrochenkov V., Sizov V., Tsinman L.: ETAP parser: state of the art. In: Computational Linguistics and Intelligent Technologies. Proceedings of the International Workshop "Dialogue 2012". Vol. 11 (18), pp. 830–848. RGGU, Moscow (2012)
11. Depparse Wiki, `http://depparse.uvt.nl/DataFormat.html`