Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2020”

Moscow, June 17–20, 2020

AN EXPERIMENTAL RULE-BASED PARSER FOR RUSSIAN EMPLOYING THE NLP RESOURCES OF THE ETAP SYSTEM[1]

Inshakova E. S. (e.s.inshakova@gmail.com)
Laboratory of Computational Linguistics, A. A. Kharkevich Institute for Information Transmission Problems;
Institute of Linguistics RAS, Moscow, Russia

Sizov V. G. (victor.sizov@gmail.com)
Laboratory of Computational Linguistics, A. A. Kharkevich Institute for Information Transmission Problems, Moscow, Russia

This paper presents a rule-based dependency parser for Russian based on bottom-up approach. Its rules are partially rewritten ETAP syntagms, organized into groups that constitute a single pipeline. We demonstrate that such an organization enhances the performance of our parser relative to the ETAP system's and enables it to successfully process long phrases (more specifically, heavy nominal and prepositional phrases at the current experimental stage of our work).

Keywords: rule-based parser, dependency grammar, natural language processing

DOI: 10.28995/2075-7182-2020-19-387-399

[1] This paper was supported by the RSF grant No. 16-18-10422-P.

ЭКСПЕРИМЕНТАЛЬНЫЙ ПРАВИЛОВЫЙ ПАРСЕР РУССКОГО ЯЗЫКА НА МАТЕРИАЛЕ ЛИНГВИСТИЧЕСКИХ РЕСУРСОВ СИСТЕМЫ ЭТАП

Иншакова Е. С. (e.s.inshakova@gmail.com)
Лаборатория компьютерной лингвистики,
Институт проблем передачи информации
им. А. А. Харкевича РАН;
Институт языкознания РАН, Москва, Россия

Сизов В. Г. (victor.sizov@gmail.com)
Лаборатория компьютерной лингвистики,
Институт проблем передачи информации
им. А. А. Харкевича РАН, Москва, Россия

1. Introduction

In this paper, we present a pilot version of a Russian language parser that uses the parsing rules of the ETAP linguistic processor [Boguslavsky et al. 2008], [Iomdin et al. 2012] as its material. It creates similar dependency structures with the same tree link labels (syntactic relations), but is based on crucially different principles.

ETAP is one of the oldest existing knowledge-based NLP applications for Russian. Its Russian language processor had been being developed for over 20 years and accumulated a lot of important linguistic information, especially in Russian combinatorial dictionary (RCD). However, complex as it is, the ETAP parser is unable to meet some challenges. First of all, it cannot reliably work on long sentences because of inability to split such sentences into linguistically acceptable chunks and overgeneration of hypothetical links that causes combinatorial explosions [Iomdin et al. 2012], [Tsinman 2011]. This, actually, is sometimes true for not-so-long sentences and other types of phrases. In our paper, we show that ETAP's resources (rules and RCD) can be used to create a higher-performance parser. This is demonstrated through the example of heavy nominal and prepositional phrases that the ETAP parser cannot process.

The paper is structured as follows. In **Section 2**, we present our parser's architecture: in **subsection 2.1**, we list its major differences from the regular ETAP; in **subsection 2.2**, we describe in detail the parsing pipeline. In **Section 3**, we provide the results of evaluation of the experimental parser's performance and compare them to the ones of the current ETAP-4. In **Section 4**, we briefly discuss the advantages and drawbacks of our algorithm.

2. The parser's architecture

2.1. General principles and differences from the ETAP parser

1. *Tree-building technique*. The regular ETAP parser is neither bottom-up nor top-down. Its binary rules that connect head and dependent words (syntagms) first construct all the possible hypothetical links, and then other rules filter them [Iomdin et al. 2012]. Our parser uses the bottom-up technique to build dependency trees. As opposed to other relatively new rule-based parsers of Russian [Anisimovich et al. 2012], [Antonova and Misyurev 2012], [Boyarsky and Kanevsky 2015], [Moskvina et al. 2016] and other languages [Gamallo 2015], [Korzeniowski and Mazurkiewicz 2017], it consistently applies the linguistic principle of constituents' hierarchy. It starts from parsing subtrees that correspond to the smallest constituents, and then at each new stage proceeds to larger ones.

2. *Dictionary vs. grammar.* The regular ETAP is dictionary-oriented. It heavily relies on the words' features from the RCD entries, e.g. those that encode specific constructions a given lexeme can participate in. Such syntactic information as word order, projectivity and punctuation marks, though also very important for ETAP, often cannot be made the most of. This is partially because the primary matrix, or network, of hypothetical links is often too noisy to check these parameters, partially because word/constituent order is not the main concern of the Meaning—Text theory that underlies the ETAP system. The filtering rules (INTER-SYNT) generally favour linear distance restrictions and discard long-distance links [Tsinman 2011], but the problem is that the linearly closest words are not necessarily the structurally closest ones.

   On the contrary, the architecture of our parser that simulates bottom-up constituency tree derivation makes it more grammar-oriented. It parallels dependency (sub)trees to constituents, and relies on their order, projectivity and punctuation when merging those subtrees together.

3. *Ordering of the rules*. In the regular ETAP, the order syntagms are applied in is irrelevant. In our parser, all syntagms are strictly ordered. Every next rule makes use of the information about links built by previous rules.

4. *The continuity condition*. In our parser, each rule, while linking two words, presupposes that i) these words are heads of subtrees that are already built (a minimal subtree is defined as a single node); ii) those subtrees are contiguous, i.e. for each node *x* between the subtrees *A* (with head *a*) and *B* (with head *b*) it is true that at least one of the homonyms of *x* (variants of *x*'s morphological analysis) is connected via a dependency chain to either *a* or *b*. However, this condition must be weakened for parenthetical expressions between the heads *a* and *b*, because they can contain such syntactic strictures that haven't yet been parsed at a given stage (e.g. if *a* is an adjective, *b* is a noun, and a parenthetical expression between *a* and *b* contains a prepositional phrase—see the parsing pipeline in **subsection 2.2**). To be able to connect *a* and *b* in such cases, we provide that words that belong to parenthetical groups between *a* and *b* need not be connected

to *a* or *b* via a dependency chain. We should also add to our definition that the heads *a* and *b* must belong to the same parenthetical group or to no parenthetical groups at all, except for one case. It is the JUXTAPOSE relation between a head word and the head of a clarifying expression in brackets (in this case, *a* and *b* belong to different parenthetical groups). The continuity condition with all these exceptions is implemented in a new binary predicate IS-CONTINUOUS(X,Y) of ETAP's formal language FORET.

As can be seen, the principle of continuity is based on the idea of projectivity. However, it can be also applied to non-projective strictures:

i) to a displaced subtree *A* and its right neighbor *B*, as in (1);
ii) to parts *A* and *B* of comparative constructions as they are presented in ETAP (2).

(1) [*Kakuyu knigu*]$_A$ [*vash prepodavatel'*]$_B$ *khotel, chtoby vy prochitali?*
‘[What book]$_A$ <did> [your teacher]$_B$ want you to read?’

(2) *takie* [*oblasti znaniya*]$_A$, [*kak*]$_B$ *istoriya filosofii*
‘such [areas of knowledge]$_A$ [as]$_B$ history of philosophy’ [in ETAP's syntax, the head *takie* ‘such’ governs the conjunction *kak* ‘as’]

2.2. The process of parsing

2.2.1. Stages of parsing

In our parser, rules are grouped into classes that correspond to the categories of constituents (= subtrees) they create, and are mostly named after these categories: Abbr[eviature], Add[itive], **AdvP, AP, DP** [determiner phrase], **NP, PP, TP** [tense phrase], TPFin[ite], TPSub[ordinate]. These big classes of rules are divided into subclasses: ∅ ‘bare’, -Attach, -Coord, -Postcoord, -With-Br[ackets] and -With-Par[enthesis]. Below we present the sequence of stages the experimental parser works in, and explain the functions of the groups of rules. Because our parser builds only NP/PPs so far, we will not give here the outline of full-sentence parsing.

Stage of parsing		Its output	Examples[1]
ABBR	ABBR	Linking parts of composite words	neftenepronitsaemyj ‘oil-tight’
	ABBR-COORD	Coordination links between parts of composite words	nefte- i gazonepronitsaemyj ‘oil- and gas-tight’

[2] In the examples, we show only the links that are built at the given stage of parsing, and do not show the links built at the preceding stages.

Stage of parsing		Its output	Examples[1]
ADDP	ADDP	Linking parts of compound numerals, or parts of many idiomatic expressions	pyat'desyat tri 'fifty-three' drug druga 'each other'
	ADDP-COORD	Linking numerals into coordinate chains	sorok dva ili tridtsat' vosem' 'fourty-two or thirty-eight'
ADVP	**Building up subtrees headed by Adv (adverbial phrases)**		
	ADVP	Linking linearly closest dependent words to Adv	ne ochen' bystro 'not very quickly'
	ADVP-COORD	Linking AdvP's into coordinate and comparative constructions	bystro ili medlenno 'quickly or slowly'
	ADVP-POSTCOORD	Building links that bypass the ready AdvP coordinate chains	ochen' i ochen' bystro lit. 'very and very quickly
	ADVP-WITH-BR	Creating long-distance links where the host (Adv) and the modifier are separated by a parenthetical expression of any kind	ochen' (200 km/ch) bystro lit. 'very (200 km/h) quickly
AP	**Building up subtrees headed by Adj (adjective phrases)**		
	AP	Linking linearly closest dependent words to Adj	ne ochen' bystryj 'not very quick

Stage of parsing		Its output	Examples[1]
AP	AP-COORD	Linking AP's into coordinate and comparative constructions	bystryj ili medlennyj 'quick or slow
	AP-POSTCOORD	Building links that bypass the ready AP coordinate chains	samyj ili ne samyj bystryj lit. 'the most or not the most quick'
	AP-WITH-BR	Creating long-distance links where the host (Adj) and the modifier are separated by a parentheti-cal expression of any kind	naskol'ko shiroko (u nas) izvestnyj lit. 'how widely (over here) known'
DP	**Building up subtrees headed by N, without complements (≈ determiner phrases)**		
	DP	Linking linearly closest dependent words to N	bol'shaya gonochnaya mashina 'big racing car' dve ili tri mashiny 'two or three cars'
	DP-COORD	Linking DP's into coordinate and comparative constructions	tvoya mashina ili moy velosiped 'your car or my bicycle'
	DP-POSTCOORD	Building links that use or bypass the ready DP coordinate chains	zelenye shapka i sharf 'green-PL hat and scarf'

Stage of parsing		Its output	Examples[1]
DP	DP-WITH-BR	Creating long-distance links where the host (N) and the modifier are separated by a parentheti-cal expression of any kind	zelenye (yey takie nravyatsya) grushi ‘green (she likes this sort) pears’
NP	**Building up subtrees headed by N, with complements and nominal modifiers (nominal phrases)**		
	NP1–4	Attaching complements and nominal modifiers to N’s	yego otvet mne ‘his-N answer [to] me’ Misha Ivanov, nash sisadmin ‘Misha Ivanov, our sysop’
	NP-COORD	Linking NP’s into coordinate and comparative constructions	tarelka salata i chashka kofe ‘a plate of salad and a cup of coffee’
	NP-POSTCOORD	Building links that bypass the ready DP coordi-nate chains	yego i yeyo otchety ‘his-N and her-N reports’
	NP-WITH-BR	Creating long-distance links where the host (N) and the modifier are separated by a parentheti-cal expression of any kind	yeyo dlinnoe (do pyat) plat’ye ‘her-N long (down to her ankles) dress’
PP	**Building up subtrees headed by Prep (prepositional phrases)**		
	PP	Building links from prepositions to N/A/Num’s	v lyubom sluchaye ‘in any case’

Stage of parsing		Its output	Examples[1]
PP	PP-WITH-BR0	Building links from prepositions that bypass parenthetical expressions	nad (ili pod) chastnoy zemley ‘above (or under) private land’
	PP-ATTACH1–5	Attaching prepositional complements and modifiers to N/A/Adv/Num/Pr	yego uverennost’ v pobede nad sopernikom ‘his confidence in his win against his rival’ v chastnosti o probleme vybora ‘in particular about the problem of choice’
	PP-COORD	Linking PP’s into coordinate and comparative constructions	v devyat’ utra ili v sem’ vechera ‘at nine a.m. or at seven p.m.’ temno, kak v pogrebe ‘as dark as in a vault’
	PP-POSTCOORD	Building links that bypass the already attached PP’s	interesnye dlya nas predlozheniya ‘offers interesting for us’ (lit. ‘interesting for us offers’)
	PP-WITH-BR1–5	Creating long-distance PP-internal links where the host and the modifier are separated by a parenthetical expression of any kind	vstrecha s nim (chtoby obsudit’ eto) v chetyre chasa ‘meeting with him (to discuss this) at four o’clock’
	PP-WITH-PAR	Drawing links to parenthetical expressions	na dva (ili na tri) chasa ran’she ‘two (or three) hours earlier’

2.2.2. Attaching multiple dependents

The above-mentioned rules, like in the regular ETAP, are binary: they connect two subtrees into a bigger (sub)tree. To enable the parser to process such groups where a head word has multiple dependent words, we subdivided the NP, PP-Attach, TP, TPFin stages into NP1–4, PP-Attach1–5, TP1–5, TPFin1–5 respectively. The result of such a subdivision partially resembles that of slot-filling parsing grammars [McCord et al. 2012], [Anisimovich et al. 2012], where each lexeme or word class has a fixed number of slots for complements, adjuncts etc. around the head word. But our way of linking multiple dependents to heads can be rather called 'layer-by-layer' attachment. Each 'layer' is such a subtree (or a pair of subtrees) to the right and/or left of the head that constitutes a projective tree with this head. The closer a layer is to the core word, the earlier it is added.

There are basically two ways of layering dependents around the head word (an N in our experiment). The first one is creating recursive rules that attach adjectives or particles to N (from the 1st to the nth one):

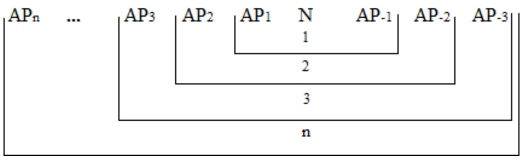

Fig. 1. 'Layers' of AP's around a nominal head

These rules apply to the same head N until the most distant AP/particle is attached to it.

The second way is multiplying and ordering the rules that attach dependent NP's, PP's, AdvP's and nominalized AP/NumP's to core words. The number of such copied rules (up to 4 so far) mirrors the maximal number of 'layers' around the core N.

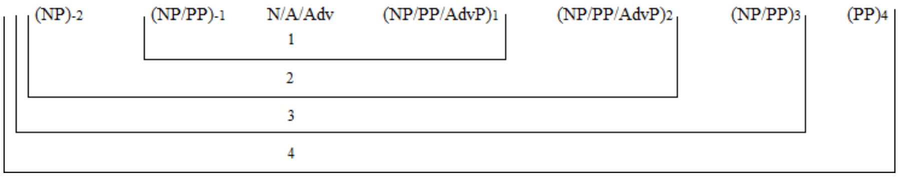

Fig. 2. The maximal number of 'layers' of dependent subtrees around a nominal head

Because dependents of the two types can intersperse, we multiplied the rules of the first type: they first apply to a bare head word, and then to the same word after its arguments have been attached. There is also another cause to multiply rules: we need links that 'circumvent' different types of subtrees. For instance, the syntagm MODIF.01 that connects adjectives to head N's should be repeated at DP, NP, and PP stages to 'circumvent' first AP's, then DP's, NP's and finally PP's (*eto ves'ma interesnoe predlozhenie* 'this very interesting offer'; *eto yego predlozhenie* 'this offer of his', lit. 'this his-N offer'; *interesnoe dlya nas predlozhenie* 'an offer interesting for us', lit. 'interesting for us offer'; *nashe s Petey predlozhenie* 'me and Petya's offer', lit. 'our with Petya offer', etc.).

3. Evaluation of the parser's performance

At the current stage of our parser's development, we restricted ourselves to constructions that are characterized by:

- POS repertory of the nodes: N, A, Adv, Particle, Conj, Prep;
- syntactic relations between the nodes: 1–5-COMPL (argument relations); QUASIAGENT (a relation between an action nominal and its genitive argument that denotes an agent or possessor); ATTRIB (it links head N's with non-argument nominal and prepositionl modifiers); MODIF (a relation between N's and their adjectival modifiers); PREPOS (a relation between a preposition and a word that it governs); RESTR (it links restrictive particles or PP's to words of any POS); LOCUT (a relation between parts of highly idiomatic expressions); COORDIN (a relation between heads of two coordinate phrases or between the head of a conjunct and a coordinate conjunction); COORD-CONJ (a relation between a coordinate conjunction and head of the last conjunct); COMPAR (a relation between a head word and a comparative conjunction or a noun in the Genitive of comparison); COMP-CONJ (a relation between a comparative conjunction and the head of the phrase denoting the standard of comparison); JUXTAPOSE (a relation between a head word and the head of a clarifying expression in brackets)[3].
- length: NP's and PP's can be as heavy as possible (in our corpus, the maximal length is 45 words).

We ran two evaluation experiments. In the first one, we used a very limited corpus that was 100 phrases long (the average phrase length was 16.6 words). The selection principle was the regular ETAP's inability to correctly parse a given NP or PP. We selected almost ½ of the phrases (49) on the basis of a certain feature—long-distance links that 'circumvent' parenthetical expressions in brackets and present a challenge for the regular ETAP parser. Other sources of errors (in 62 sentences) are very diverse: e.g. unknown words, long phrase length, incorrect PP-attachment or conjunct attachment, very long linear distance between the head and the dependent word, and constructions undescribed in ETAP rules. The phrases were taken from the Russian National Corpus (some of them had been originally sentences that we nominalized).

[3] See the full tagset of the ETAP parser in http://ruscorpora.ru/new/instruction-syntax.html.

Table 1. Results of the 1st evaluation experiment (100 phrases)

Parameters	The regular ETAP parser	The experimental parser
Number/percentage of correct heads (UAS)	1,377 (82%)	1,528 (92%)
Number/percentage of correct dependency labels	1,427 (85%)	1,568 (94%)
Number/percentage of correct heads and dependency labels (LAS)	1,327 (79%)	1,498 (90%)
Number/percentage of correct syntactic structures (without dependency labels)	3 (3%)	37 (37%)
Number/percentage of correct syntactic structures (with dependency labels)	0	29 (29%)
Processing time	64 sec.	41 sec.

For the second evaluation experiment, we extracted from the SynTagRus treebank all subtrees: 1) headed by N or Prep, 2) featuring only the POS and syntactic relations from the list given above, 3) being ≥ 10 words long (2,838 phrases in total). 2,500 phrases comprised the testing corpus, and 338 were used as a development corpus. The evaluation results for our parser and the regular ETAP are presented in Table 2.

Table 2. Results of the 2nd evaluation experiment (2,500 phrases)

Parameters	The regular ETAP	The experimental parser
UAS	91.21	93.45
LAS	88.68	90.94
Number/percentage of correct structures (without dependency labels)	1185 (47.40)	1354 (54.16)
Number/percentage of correct structures (with dependency labels)	922 (36.88)	1048 (41.92)
Processing time	948 sec.	738 sec.

4. Conclusion

The evaluation results show that at the current stage our experimental parser outperforms the regular ETAP parser. However, as can be seen from the difference between its performance in the 1st and in the 2nd experiments, our parser is still somehow 'tuned' to cope with a certain type of errors, which is frequent in the 1st testing corpus and infrequent in the 2nd one. Besides that, the experimental parser faces some other problems. Firstly, its pipeline architecture is fraught with the risk of accumulation of errors with every step; secondly, the multiplication of rules (by approximately

10 times for NP's / PP's so far) can potentially increase the processing time of whole sentences. Thirdly, some issues listed in [Iomdin et al. 2012] still hold for the experimental parser, e.g. it cannot yet correctly parse structures with gapping, phrases with orthographical errors and other kinds of non-standard spelling.

One may also note that the regular ETAP and our parser often make different types of errors. For instance, if ETAP fails to find a correct host for some word, it is likely to attach this word to some of the linear closest ones (e.g. via the ATTRIB relation), whereas our parser attaches 'hostless' words to the root of the phrase. Consequently, if the experimental parser is given a whole sentence, it will pick out all the NP/PP/AP's and correctly parse them (this capability makes it possible to use it as an NP/PP extractor).

References

1. *Anisimovich K. V. et al.* (2012) Syntactic and Semantic Parser Based on ABBYY Compreno Linguistic Technologies. In: Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". Issue 11 (18). P. 90–103.
2. *Antonova A. A., Misyurev A. V.* (2012) Russian dependency parser SyntAutom at the Dialogue-2012 parser evaluation task. In: Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". Issue 11 (18). P. 104–118.
3. *Boguslavsky I. M. et al.* (2008) Parser of the ETAP system and its evaluation with the aid of a deeply annotated corpus of Russian texts [Sintaksicheskiy analizator sistemy ETAP i yego otsenka s pomoshchyu gluboko razmechennogo korpusa russkikh tekstov]. In: Proceedings of the international conference 'Corpus linguistics—2008' (St. Petersburg). P. 56–74.
4. *Boyarsky K., Kanevsky E.* (2015) SemSin semantic and syntactic parser [Semantiko-sintaksicheskiy parser SimSin]. In: Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 15, no. 5, pp. 869–876.
5. *Gamallo P.* (2015) Dependency Parsing with Compression Rules. In: The 14th International Conference on Parsing Technologies (IWPT-2015). P. 107–117.
6. *Iomdin L. L. et al.* (2012) ETAP parser: state of the art. In: Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". Issue 11 (18). P. 119–131.
7. *Korzeniowski M., Mazurkiewicz J.* (2017) Rule Based Dependency Parser for Polish Language. In: Rutkowski L., Korytkowski M., Scherer R., Tadeusiewicz R., Zadeh L., Zurada J. (eds) Artificial Intelligence and Soft Computing. ICAISC 2017. Lecture Notes in Computer Science, vol. 10246. P. 498–508. Springer.
8. *McCord M.C., Murdock J. W., Boguraev B. K.* (2012) Deep parsing in Watson. In: IBM Journal of Research and Development, vol. 56 3.4: IBM, pp. 3–1.
9. *Moskvina A. D. et al.* (2016) Development of the Core for Syntactic Parser for Russian based on NLTK libraries [Razrabotka yadra sintaksicheskogo analizatora dlya russkogo yazyka na osnove bibliotek NLTK]. In: Computer linguistics and computational ontologies. Proceedings of the XIX joint scientific conference.

10. *Tsinman L. L.* (2011) Ranking syntactic hypotheses in the syntactic parser of ETAP-3 linguistic processor [Ranzhirovanie sintaksicheskikh gipotez v sintaksicheskom analizatore lingvisticheskogo protsessora ETAP-3]. In: Word and Language [Slovo i yazyk]. Festschrift for Yu. D. Apresyan's 80th anniversary. Moscow, 2011. P. 573–587.