Parsing Russian: a Hybrid Approach

Dan Skatov, Sergey Liverko,
Vladimir Okatiev, Dmitry Strebkov
Russian Federation, Nizhny Novgorod
Dictum Ltd
{ds,liverko,oka,strebkov}@dictum.ru

Abstract

We present an approach for natural language parsing in which dependency and constituency parses are acquired simultaneously. This leads to accurate parses represented in a specific way, richer than constituency or dependency tree. It also allows reducing parsing time complexity. Within the proposed approach, we show how to treat some significant phenomena of the Russian language and also perform a brief evaluation of the parser implementation, known as *DictaScope Syntax*.

1 Introduction

A syntactic parser inputs a sentence and produces information on syntactic relationships between parts of the sentence. It is an open question which method is the most convenient one to represent these relationships. In this paper, we are focusing on two of those methods. The first one, a *constituency tree (CT)*, is a representation of a sentence by a set of nested fragments — groups, each group corresponding to a syntactically coherent phrase. The second one, a *dependency tree (DT)*, expresses relationships by a set of syntactic links between pairs of tokens.

Figure 1 demonstrates correspondence between CT and DT: one is clearly derivable from another. In applications, one usually needs to transform CT into DT due to the following fact: if a tree is correct, then subjects, objects and adverbials of some predicate X are always direct children of the node X in DT. With a traditional CT framework these children can be obtained in much less intuitive way by browsing up and down through constituents, as shown in Figure 1 by dotted lines. According to this comparison, DT transparently maps onto the level of semantic representation, thereby DT-s are considered most appropriate in applications (Jurafsky and Martin, 2009) like sentiment analysis and fact extraction.

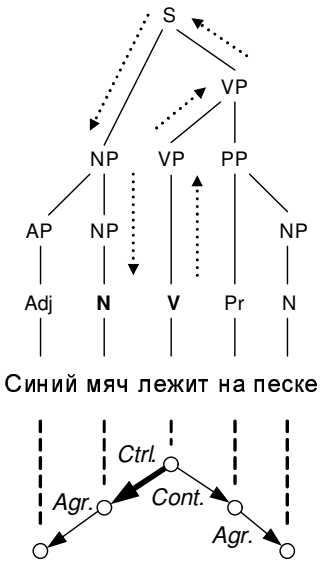

Figure 1: A constituency tree (upper) and a dependency tree (lower) for a sentence *"A blue ball lies on the sand"*.

Constituency parsing. Despite the usefulness of DT-s, CT-s have a longer history of application as a computational model. For now, probabilistic constituency parser by (Charniak, 1997) and its derivatives are considered the state of the art for English. Unfortunately, the framework of constituency parsing, taken alone, is not productive for languages such as Russian. It turns out that the number of rules in a grammar start to grow fast if one tries to describe an inflecting language with a free word order explicitly. As a result, pure constituency parsers are not well known for Russian. It has recently been confirmed by a Russian syntactic parsers task at the Dialogue conference (see http://www.dialog-21.ru), at which several parsers were presented and all of them used DT formalism as a basis.

Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing, pages 34–42,
Sofia, Bulgaria, 8-9 August 2013.

Dependency parsing. Modern algorithmic approaches to dependency parsing are based on machine learning techniques and are supported by open-source implementations. Unfortunately, large DT-s corpora are not widely available for Russian to train these parsers. The need for the corpora also brings complications when one wants to achieve high precision parsing a given subject domain, and then to switch to parse another domain: eventually one will need a separate corpus for each domain. There is a consent among researches that a "long tail" of special error cases is definitely hard to fix in pure machine learning frameworks (Sharov and Nivre, 2011), while it is necessary for high precision. In contrast to English, dependency parsing is traditional for Russian computational linguistics. As a result, modern Russian parsers produce DT-s. These parsers are mainly rule-based with an optional statistical component (Toldova et al., 2012) and standard expert-verified data sets such as verb subcategorization frames, which are called "control models" or "set of valences" in Russian. None of the rule-based parsers that were presented at the Dialogue task are freely available.

Unfortunately, the practice of using DT-s has revealed some of their significant deficiencies. The most frequently discussed one is the representation of homogenous parts of the sentence (Testelets, 2001). Figure 2 shows some known methods. One can observe that there must be a syntactic agreement between the group of homogenous parts `1-3` and their parent `4-6`[1] by `Number`, which is `Plural`, but it is impossible to capture this relation in a DT where only words can hold grammar values. No representation among A-E in Figure 2 keeps this information for the group. Things get worse if one tries to represent an agreement for two groups of homogenous parts, like in 2.F. In addition, it is common to modify the parsing algorithm, but not the set of decision rules, directly in order to get nonprojective[2] DT-s (McDonald et al., 2005). The list of open problems can be extended further: paired conjunctions, nested sentences, introductory phrases etc.

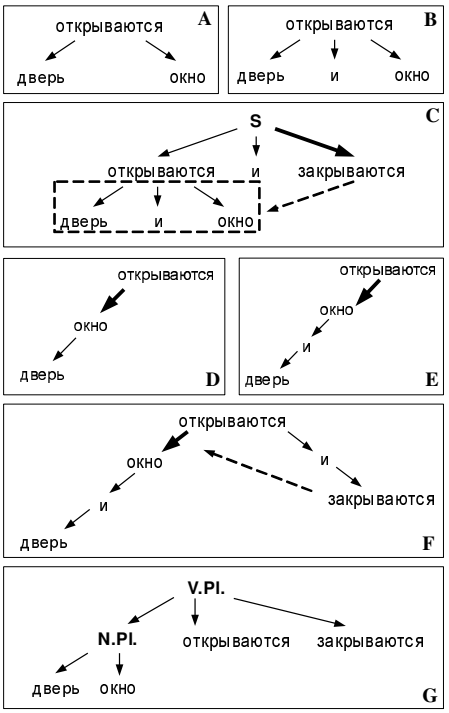

Figure 2: Uncertainty with the representation of homogenous parts in dependency trees for *"дверь$_1$ и$_2$ окно$_3$ открываются$_4$ и$_5$ закрываются$_6$"* *"door$_1$ and$_2$ window$_3$ open$_4$ and$_5$ close$_6$"*

Toward the hybrid approach. It would be possible to resolve problems with homogenous parts if additional vertices could be added to the DT, like in Figure 2.G, representing supplementary constituents with synthesized grammar values. Unfortunately, known approaches for dependency parsing assume that all vertices are predefined before the algorithm starts, so it is impossible to include new vertices on the fly without inventing new parsing methods.

The idea of combining dependencies and constituencies directly is not a new one. For Russian, (Gladkij, 1985) suggested designating a standard relation between predicate and subject by a syntactic link, along with adding a separate constituent for compound predicative groups like *"должен уступить"* from *"Дорогу$_1$ должен$_2$ уступить$_3$ водитель$_4$" "The driver$_4$ must$_2$ give$_3$ way$_1$ (to smb.)..."*, which has a nonprojective DT. This solution immediately reduces the number of overlapping dependency links for compound predicates, because links that tend to over-

[1] In examples, we put indices for words in correspondence with English translation (often with omitted articles *"a"*, *"the"*), refer to any word by its index, and to a phrase by indices of its starting and finishing word.

[2] Dependency tree is called *projective* if each subtree corresponds to a continuous fragment of the source sentence. There is evidence that more than 80% of the sentences are usually projective in European natural languages. A famous example of nonprojectivity for Russian is *"Я$_1$ памятник$_2$ себе$_3$ воздвиг$_4$ нерукотворный$_5$" "I$_1$'ve raised$_4$ a monument$_2$ for myself$_3$ not made by hands$_5$"* from Pushkin, where link `4→1` overlaps `2→5`.

lap are packed inside the constituent. In (Kurohashi and Nagao, 1994) constituents for groups of homogenous parts are prebuilt to be treated as single units during the next step by a dependency parser for Japanese. In (Okatiev et al., 2010) preprocessing of certain tokens connected to constituents is performed before the dependency parsing. In (Wang and Zong, 2010) a third-party black-box dependency parser is used to improve the results of author's constituency parser, achieving 10% boost in F_1. Additionally, there is evidence that ABBYY Compreno (Anisimovich et al., 2012) tracks the order in sequences of consecutive tokens so it actually exploits some kind of constituents throughout the process of dependency parsing.

In this paper, we propose a method of representing a parse tree, which is a combination of CT and DT and eliminates some disadvantages of both. Then we describe syntactic rules that guide the process of simultaneous bottom-up acquisition of multiple syntactically ambiguous DT-s and CT-s for a given sentence, ranked by syntactic relevance. In addition, we discuss some properties of the rule base that is built in the framework of proposed rules description language. After that, we show that it is possible to extend the classical Cocke-Younger-Kasami algorithm (Kasami, 1965) to use grammar rules of arbitrary arity without grammar binarization and to exploit intermediate DT-s to increase efficiency. Moreover, we demonstrate how to achieve a reasonable ranking of solutions without using any additional statistical information. In conclusion, we discuss possible extensions of the approach.

2 Representing the Parse Tree

Our goal is to achieve the most complete representation of syntactic relations between words and/or chunks of a given sentence. The subject of semantic representation, e.g. semantic labeling of predicate participants or questions on the edges, are not discussed further. We assume that such information can be surfaced either during the process of analysis or directly afterwards by semantic interpretation of resulting parse trees.

As it was mentioned, it is possible to avoid disadvantages of DT-s if one finds a way to bring additional vertices to these trees. Though it is not natural for DT-s to have such vertices, it is common for constituents in CT-s to acquire derived grammar values. Thereafter, instead of building a parse tree of a completely new type, we obtain the representation that consists of three components: 1) constituency tree — CT, 2) dependency tree — DT, 3) hybrid tree — HT. CT strictly corresponds to a system of syntactic rules given preliminarily, and DT is built by instructions from rules at each step of the bottom-up analysis driven by this system. A hybrid representation, HT, that is based on DT but depicts homogenous parts, dependent clauses and nested sentences in a particular way. As a next step, we declare a balanced system of agreements between these different representations.

Assumptions for a constituency tree. We only require that CT correctly corresponds to the given sentence, and phrases of CT do not need to correspond to the expected expressions of VP, NP, AP etc. E.g., one can declare a specific rule that generates a constituent that simultaneously corresponds to a subject and a verb, e.g. *"должен уступить водитель"*. Such a phrase corresponds both to VP and NP and is atypical, but the corresponding rule can produce a proper fragment of a nonprojective DT and HT, which is the main goal in this context. To summarize, in the proposed model constituencies play an utilitarian role, they are treated like carriers for parts of DT and are borrowed at certain decision points to form a resulting HT.

Assumptions for a dependency tree. During the parsing, a *hypothesis of syntactic locality* is considered, that states: tokens that are close linearly in a sentence are more likely to have a syntactic relationship. If there are several methods to represent a syntactic phenomenon, we choose a method that fits it best. Recalling Figure 2, one can observe that in 2.A and 2.B two links correspond to different linear distances between corresponding vertices, while in C–F each link corresponds to a unity distance. Let us call the latter kind of homogeneity representation "a chain scheme" and require syntactic rules to follow it.

The following assumptions are made: 1) prepositions become roots of the corresponding prepositional phrases; 2) subordinate conjunctions become roots of the corresponding dependent clauses; 3) punctuation marks, quotes and coordination conjunctions are removed from DT and will be properly presented in HT.

The following link types are expected: agreement (`Adj+Noun`, `Noun+Noun`, etc), control

(Verb+Noun, prepositional phrases etc), contiguity (for adverbs, particles etc.), isolation for dependent clauses and coordination for chains of homogenous parts. For cases when a synthesized grammar value is needed like in Figure 2.G, we do not expect a special new vertex to be introduced in DT. Instead, each vertex of DT contains a reference to the corresponding constituency in CT, which holds its own grammar value. By default, this value is inherited from the head element of the constituent, but can be overridden, which is the case for homogeneity.

Assumptions for a hybrid tree are revealed by the example in Figure 3, where an XML-file is represented for a HT of a sentence with two dependent clauses 8–11 and 12–19, a nested sentence 1–19 and homogenous nouns 17 and 19. Only significant details are shown in the figure — shortened grammar values (N for Noun etc) as GV and wordforms as W. Full representation also includes normal forms, detailed grammar values, tokens' positions and link types.

Subordinate clauses are presented under Subord tag as a nested group, while nested sentences are located under S tag. The group of two homogenous nouns is placed under Coord, but, unlike corresponding DT, homogenous members do not form a chain and are located on a single level inside a special Group tag. Such Group can have any number of dependent elements which are placed between </Group> and </Coord> (shown in Figure 4 below). There, the agreement by number between the group of two nouns 2–4 and the adjective 1 is taken into account, and the adverb 10 adjoins to the group of verbs 6 and 9 but not to a single verb.

An intensive research has been performed by (Okatiev et al., 2010) on the role of punctuation for Russian. According to it, one has to distinguish roles of punctuation marks and conjunctions (which together are called junctions) to deduce correct parses. For this reason roles are marked in a hybrid XML tree. One punctuation mark or a conjunction can possess several roles: it can be an isolator (Isol, bounds dependent phrases), a separator (Sepr, is used to separate homogenous parts inside coordinated groups) and a connector (Conn, indicates nested sentences, e.g. quotation marks). Isolators and connectors always play an opening or a closing role for one or several clauses. In the sentence from Figure 3, the

```
<S T="«Я не могу … он">
<V W="объясняет" GV="V">
  <S T="Я не могу требовать…">
    <Conn W='«'/>
    <V W="могу" GV="V">
      <V W="Я" GV="Pron">
      <V W="не" GV="Part">
      <V W="требовать" GV="V">
        <V W="бережливости" GV="N">
          <V W="от">
            <V W="людей">
              <Subord GrV="V">
                <Isol W=","/>
                <V W="работают GV="V">
                  <V W="которые" GV="Pron">
                  <V W="на" GV="Pr">
                    <V W="меня" GV="Pron">
                  </V>
                </V>
                <Isol W=","/>
              </Subord>
            </V>
          </V>
        </V>
      </V>
      <Subord GrV="V">
        <Isol W=","/>
        <Isol W="если"/>
        <V W="буду" GV="V">
          <V W="проводить" GV="V">
            <V W="время" GV="N">
              <V W="в" GV="Prep">
                <Coord>
                  <Group GrV="N">
                    <V W="роскоши" GV="N">
                    <Sepr W="и"/>
                    <V W="комфорте" GV="N">
                  </Group>
                </Coord>
              </V>
            </V>
          </V>
        </V>
      </Subord>
    </V>
    <Conn W='»'/>
  </S>
  <Conn W="—"/>
  <V W="он">
</V>
</S>
```

Figure 3: A HT for *"«Я$_1$ не$_2$ могу$_3$ требовать$_4$ бережливости$_5$ от$_6$ людей$_7$, которые$_8$ на$_9$ меня$_{10}$ работают$_{11}$, если$_{12}$ буду$_{13}$ проводить$_{14}$ время$_{15}$ в$_{16}$ роскоши$_{17}$ и$_{18}$ комфорте$_{19}$», — объясняет$_{20}$ он$_{21}$"*
"«I$_1$ can$_3$ not$_2$ require$_4$ thrift$_5$ of$_6$ people$_7$ who$_8$ work$_{11}$ for$_9$ me$_{10}$ if$_{12}$ I spend$_{14}$ time$_{15}$ in$_{16}$ luxury$_{17}$ and$_{18}$ comfort$_{19}$», — he$_{21}$ explains$_{20}$".

comma between 11 and 12 is a closing isolator for clause 8–11 and also and opening isolator for 12–18. Therefore this comma is mentioned twice in shown XML file in the beginning and the ending of the corresponding <Subord>-s. In general, HT contains at least as many vertices as there are tokens in a sentence, and possible duplicates correspond to punctuation marks and conjunctions.

Another case of multiple roles for the punctuation is shown by the example *"Круг$_1$, закрашенный$_2$ синим$_3$, равнобедренный$_4$*

треугольник$_5$ *и*$_6$ *жёлтый*$_7$ *квадрат*$_8$*"* *"Circle*$_1$*, shaded*$_2$ *in*$_2$ *blue*$_3$*, isosceles*$_4$ *triangle*$_5$ *and*$_6$ *yellow*$_7$ *square*$_8$*"* where the comma between `3` and `4` is an ending isolator for `2-3` and also a separator for parts `1` and `5` of a group of homogenous nouns. In addition, the case of an isolating group of a comma and a following conjunction, like *", если" ", if"*, is always being discussed: is it a single token, should one remove a comma or leave it, etc. In this case, this group is just a pair of consecutive isolators in HT XML, see Figures 3 and 4.

```
<S>
<Coord>
   <Group GV="Verb">
      <Sepr W="как"/>
      <V W="открываются" GV="Verb"/>
      <Sepr W=","/>
      <Sepr W="так и"/>
      <V W="закрываются" GV="Verb"/>
   </Group>
   <Coord>
      <Group GV="Noun">
         <V W="дверь" GV="Noun"/>
         <Sepr W="и"/>
         <V W="окно" GV="Noun"/>
      </Group>
      <V W="Большие" GV="Adj"/>
   </Coord>
   <V W="тихо" GV="Adv"/>
</Coord>
</S>
```

Figure 4: A HT for *"Большие*$_1$ *дверь*$_2$ *и*$_3$ *окно*$_4$ *как*$_5$ *открываются*$_6$*, так*$_7$ *и*$_8$ *закрываются*$_9$ *тихо*$_{10}$*"* *"The large*$_1$ *door*$_2$ *and*$_3$ *window*$_4$ *both*$_5$ *open*$_6$ *and*$_{7,8}$ *close*$_9$ *silently*$_{10}$*"*.

The way in which paired conjunctions are treated in HT is synchronized with the representation of single coordinative conjunctions. E.g., *"как . . . , так и . . . "* *"both . . . and . . . "* is followed by a rule *"*$S \rightarrow$ *как* S*, так и* S*"*, where *"S"*-s denote clauses, yielding the parse in Figure 4, which is difficult to obtain in a pure DT.

3 The Rule Base

Linguistic data. We use a set of subcategorization frames for about 25 000 verbs, deverbatives and other predicative words, which is collected at our side through standard techniques of collocations and lexics aquisition from (Manning and Schütze, 1999). A morphological dictionary consists of about 5 mln wordforms.

Morphological hierarchy. In many cases, it is useful to unite different parts of speech into one metapart which possesses some common properties. E.g., participles share properties of verbs and adjectives. For verbs, the class `ComVerb` is introduced, which includes `Verb` in both finite and infinite forms, `Participle` and `Transgressive`, for adjectives — the class `ComAdj` with `FullAdj`, `ShortAdj` and `Participle`. This concept leads to the reduction in the number of syntactic rules.

The language of syntactic rules is shown by an example that describes a coordination relation between an adjective and a noun:

```
// Высокая спинка "high back"
AgreeNounAdj {
  T: [ComAdj] [ComNoun];
  C: NumberGenderCaseAgree (PH1, PH2);
  Main: 2; L: 2=>Agreement=>1;
}
```

Section `T` declares that a template should check grammar values of consecutive phrases `PH1` and `PH2`, while section `C` checks required properties of them. If phrases fit `T` and `C`, then a DT is built according to `L` section by a link from the main word of the second constituent to the main word of the first, plus trees of `PH1` and `PH2`. The second phrase is declared the head one (`Main: 2`) for the new phrase built by `AgreeNounAdj` rule.

Coordination. It is possible to override grammar value of a new phrase `PH`, which is shown by an example of homogenous nouns:

```
// Яблоко, груша "apple, pear"
CoordNounComma {
  T: [ComNoun] <,> [ComNoun];
  C: (PH1.Case == PH2.Case) && !PH1.IsCoord
     && PH2.NormalForm != "который";
  Main: 1; L: 1=>Coord=>2; J: 1<=Sepr;
  A: PH.Number = NUMBER_PL;
     PH.IsCoord = true;
}
```

`Number` of the entire phrase is set to `Plural`. In addition, the role of the comma is set to `Sepr`. `IsCoord` property is introduced to phrases to prune the propagation of different bracketings. E.g., for a chain of nouns *"A и B, C и D"* a large number of bracketings are possible: *"[[A и B], [C и D]]"*, *"[A и [B, [C и D]]]"* etc. To prune this to exactly one correct bracketing, we deny chaining phrases that both contain chains of nonunity length and allow only left-to-right chains by a check `!PH1.IsCoord`.

Ambiguous compounds. Some compounds in Russian have their own grammar values in certain contexts. E.g., *"по причине" "by reason of"* is sometimes a preposition equivalent to *"из-за" "because of"* (as preposition): *"[судили [по [причине и следствию]]]" "judged on reason and consequence"* vs. *"[опоздал [по причине]*

пробок]" "was late by reason of traffic jams". In contrast to context rules for such constructions, we use pure syntactic rules for testing both versions, introducing a compound version by the rule:

```
CompoundPrep {
  T: [Any] [Any];
  C: IsCompoundPrep (PH1, PH2);
  Main: 1; L: 1=>Compound=>2;
  A: PH.Type = PHRASE_PREP;
}
```

Nonprojective parses for compound predicates are processed by the following rule which produces a nonprojective DT. Despite nonprojectivity, it brings no problem for CT parsing process:

```
// Дорогу должен уступить…
ControlNonProjectLeft {
  T: [Any] [Any] [Any];
  C: PredicModel (PH2, PH3) &&
     IsFreeValence (PH2, PH3) &&
     PredicModel (PH3, PH1) &&
     IsFreeValence (PH3, PH1);
  Main: 2;
  L: 2=>Control=>3; 3=>Control=>1;
  A: FillValence (PH, PH3);
}
```

Punctuation. Roles of junctions guide the parsing process. E.g., we consider that a junction that has exactly one role, which is ending isolator, is equivalent to a whitespace. Let us see how the rule `AgreeNounAdj` will parse the example under this assumption: *"синий$_1$, как$_2$ море$_3$, оттенок$_4$" "shade$_4$, as blue$_1$ as$_2$ a sea$_3$"*. One can verify that, due to consideration, the second comma no longer prevents this phrase from being covered by `AgreeNounAdj`. Another way to track these roles is to reject false control in genitive, e.g. *"много$_1$ [кругов$_2$, заполненных$_3$ белым$_4$, квадратов$_5$], эллипсов$_6$" "lots$_1$ of$_1$ circles$_2$, shaded$_3$ in$_3$ white$_4$, squares$_5$, ellipses$_6$"*.

Overall base. For Russian, we have built a rule base with 90 rules, divided into 7 groups, one for each type of syntactic relations to be included into DT. These rules exploit 20 predicates in criterion sections and 10 additional phrase properties.

4 The Algorithm

We provide a modification of the Cocke-Yanger-Kasami algorithm (CYK) to find DT-s by corresponding CT-s that can be derived by rules described above and that are best in a specified way.

In our interpretation, CYK has the following structure. It inputs a sentence of n tokens. Empty $n \times n$ matrix M is set up and its main diagonal is filled with one-token phrases, each phrase at a cell $M[i][i]$ takes some grammar value from the i-th token of the sentence, $i = 1, \ldots, n$ as its head. CYK iterates all diagonals from the main diagonal of M to its upper-right cell $M[1][n]$. Each cell on the diagonal of length $k \in \{n-1, \ldots, 1\}$ will contain only phrases of exactly k consecutive tokens, so $M[1][n]$ will contain exactly those phrases that correspond to consecutive tokens, i.e. the entire sentence.

For CYK, it is traditionally assumed that each rule has exactly two nonterminals, and grammars formed by such rules are called *binarized grammars* (also known as *"in Chomsky normal form"*). Now consider the binarized rule $R : Ph_1 Ph_2 \rightarrow Ph$. If one wants to derive a phrase Ph of consecutive tokens by this rule, then one should look at consecutive phrases Ph_1 and Ph_2 with properties defined by R and of j and $k - j$ tokens correspondently. So, standing at $M[i][i + n - k]$, $i \in \{1, \ldots, k\}$, CYK searches for Ph_1 in j cells in the current row to the left and then for Ph_2 in $k-j$ lower diagonals in the current column to the bottom, checking them by R if both Ph_1 and Ph_2 are found for some j and storing corresponding Ph in $M[i][i+n-k]$. Figure 5 shows M at the moment when CYK has finished for a particular input.

		N$_5$= Adj+N$_4$
Adj высокая	**N$_3$=** Adj+N$_1$	**N$_6$=** N$_3$+N$_2$
	N$_1$ спинка	**N$_4$=** N$_1$+N$_2$
		N$_2$ стула

Figure 5: The CYK matrix after *"высокая$_1$ спинка$_2$ стула$_3$" "high$_1$ chair$_3$ back$_2$"*.

Rules of arbitrary arity. Surprisingly, the extension of CYK for grammars that are not binarized is not widely discussed in literature. Instead, issues of binarization and special classes of grammars that do not lead to exponential growth of the binarized version are proposed (Lange and Leiß, 2009). Indeed, due to the author's experience, researchers argue to reject using CYK, because the increase in the size of the grammar through binarization degrades the performance significantly. Although it is true, we further show that it is not necessary at all to binarize grammar to use CYK.

To use the rule system proposed earlier, we modify CYK in the following way. Consider the rule $R : Ph_1\ Ph_2\ \ldots\ Ph_r \rightarrow Ph$. One can treat it as a rule $R' : Ph_1\ PhL \rightarrow Ph$, where $PhL = Ph_2\ Ph_3\ \ldots\ Ph_r$. In this way, the problem reduces to the former case of binarized grammar. When a reduction is applied to PhL recursively and finally some Ph_r is fixed, a set of $Ph_1, \ldots, Ph_r$ can be checked against R. This check is performed as described in Section 3. One can verify that modification increases the worst run time of CYK by a polynomial multiplier $\mathcal{O}(n^r)$, and it is always an overestimation for natural languages, for which the case $r \geq 4$ is rare. Moreover, a big room for optimization is left. E.g., it is possible to extract checks from R that correspond to $Ph_1, \ldots, Ph_m$, $m < r$, and apply them before all r phrases are collected to be checked.

Equivalency of phrases. In Figure 5 two final parses are derived by two different ways but these parses correspond exactly to the same phrase with no syntactic ambiguity. When $n > 5$, matrix cells' content becomes too large due to this effect, which leads to a significant decrease in CYK performance. Let us recall that the main goal of the process is to obtain correct DT-s. Let us notice then that two parse results in $M[1][3]$ from Figure 5 carry the same DT. Therefore it is necessary to merge phrases in a cell if they carry identical DT-s. Let us assume that CYK had already put S phrases in a cell, and a new phrase P is pending. CYK then checks P against all elements of S and declines to add P in S at the first time when it finds some $p \in P$ that carry the same DT as P.

Notice that it is insufficient to merge two phrases only by properties of the head. E.g., for *"Я*$_1$ *не*$_2$ *посещал*$_3$ *палаты*$_4$ *мер*$_5$ *и*$_6$ *весов*$_7$*"* *"I*$_1$ *did*$_3$ *not*$_2$ *attend*$_3$ *the Chamber*$_4$ *of Weights*$_5$ *and*$_6$ *Measures*$_7$*"* the first possible phrasal coverage is `[1 2 3 [4 [5 6 7]]]`, the second is `[1 2 3 [[4 5] 6 [7]]]`, and it is not known which is correct at a syntactic level. For both coverages, the head of the group is the same verb with subject and object slots filled, while the underlying DT-s differ.

Shallow weighting. Due to a high level of ambiguity in natural languages, a huge amount of phrases can be obtained in subcells even when the merging of phrases takes place as described above. Therefore it is necessary to delete some portion of irrelevant phrases from subcells.

For every phrase that arises in any step of CYK, let us add a weight to this phrase by the following scheme. For each edge $e = (i, j)$ of the corresponding DT that forms a link from i-th to j-th word of a sentence, we attach a weight $|j - i|^q$. The weight W of the phrase is a sum of weights of all of the edges of its DT.

For every cell of M, after the set of phrases S is complete by CYK, S is sorted by the weights of phrases. After S is sorted, it turns out to be separated into layers by the weights. Finally, only α top layers are left in a cell. Our evaluation has showed that $q = 2$ and $\alpha = 2$ are sufficient.

Processing mistypings as if the correction variants were additional grammar values of tokens, being incorporated into the algorithm by a scheme given by (Erekhinskaya et al., 2011), improves F_1 up to 20% on real-world texts from the Internet without significant loss in the performance.

Partial parses in case there is an error in an input that are good enough and look much like ones from pure dependency parsers can be obtained by the proposed algorithm, in contrast to shift-reduce approaches, in which only some left part of the sentence with an error is parsed.

5 Evaluation

There is a lack of corpora for Russian to evaluate parsers. In 2012, a task for Russian syntactic parsers was held during the Dialogue conference. The evaluation was conducted as follows: every parser processed a set of thousands of separate sentences from news and fiction, and then a "golden standard" of 800 sentences was selected and verified by several assessors. During evaluation, some mistakes, such as prepositional phrase attachment, were not taken into account as syntactic parsers are originally not intended to deal with semantics. Ignoring this, the method of evaluation was exactly UAS (*unlabeled attach score*, i.e. a number of nodes in a DT that have correct parents, see (McDonald et al., 2005)).

Our previous version of *DictaScope Syntax* parser, which was based on a modification of Eisner's algorithm (Erekhinskaya et al., 2011), took part in that task in 2012, resulting with 5-th place out of 7 (systems have been ranged by F_1), with 86,3% precision, 98% recall and 0,917 F_1. Our current evaluation of the new version of *DictaScope Syntax* parser, based on methods proposed in this paper, follows the technique from the

Dialogue-2012 task (Toldova et al., 2012). We took 800 entries from the same set of sentences and marked them up in HT XML format. In evaluation we followed the same principles as described in (Toldova et al., 2012), reaching 93.1% precision, 97% recall and 95% F_1, which correspond to the 3rd place out of 7, with a lag of half percent from the second place. We have also marked up a corpus from Internet-forums and Wikipedia of 300 sentences, reaching 87% F_1.

Note on complexity. It is known that for a sentence of n tokens CYK is $\mathcal{O}(n^3)$ algorithm by worst case complexity, and this complexity can be reduced to $\mathcal{O}(n^{2.38})$ by algebraic tricks (Valiant, 1975). We have performed a time complexity evaluation of our parser on a corpus of 1 mln Russian sentences from Internet-news, averaging the time for every fixed length of the sentence. We evaluated sentences with lengths from 3 to 40 tokens, 12 tokens average length. Evaluation has showed the performance of 25 sentences per second average for one kernel of 3GHz Intel Quad. The evaluation has also led to a plot given in Figure 6.

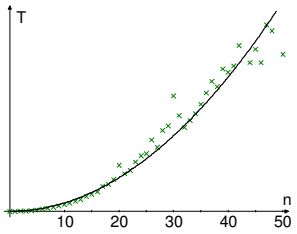

Figure 6: Average complexity of parsing as a function of the number of tokens

It can be verified that the plot corresponds only to An^2 for some A, but not to An^3. We can explain it in the following way. With our grammar and Russian language, we have noticed that for a completed upper-triangular matrix of CYK, nonempty cells on each row are denser near the main diagonal. Following this, for the part of the row that forms m cells to the right of the main diagonal, the density of nonempty cells in it is $p_m \leq \frac{c}{m}$ for some c. Now assume that 1) the maximum cost of the rule checking operation and 2) the maximum number of phrases' combinations that need to be verified against the rule base are some constants which depend on rules, 3) τ is the number of rules which are stored in a vocabulary with a key formed by grammar values from templates. Then, the total number of rule checks is

$$T_{total} \leq c \cdot \sum_{k=n-1}^{1} \sum_{i=1}^{k} \sum_{j=1}^{n-k} p_{n-k} \cdot \log \tau \leq$$
$$\leq c \cdot \log \tau \cdot \sum_{k=1}^{n-1} \sum_{i=1}^{k} \sum_{j=1}^{n-k} \frac{C}{n-k} =$$
$$= c \cdot C \cdot \log \tau \cdot \sum_{k=1}^{n-1} k = \mathcal{O}\left(n^2 \log \tau\right) \ .$$

6 Discussion

In this paper we proposed a method of parsing Russian, based on a hybrid representation of the result, which is derived from a dependency tree with elements of the corresponding constituency tree to model phenomena like homogenous members, nested sentences and junction roles. This approach led to the elimination of some disadvantages of both representations. We also presented a rule system and an algorithm to acquire a ranked set of syntactically ambiguous representations of that kind for a given sentence. Properties of the Cocke–Younger–Kasami algorithm and its modifications, remarkable for natural language parsing, are particularly discussed. The *DictaScope Syntax* parser, based on the proposed results, is embedded in a commercial NLP system, that is adopted in *Kribrum.ru* — a service for Internet-based reputation management.

The natural question is whether this approach can be extended to parse other languages. We perform the development of rule systems for English and Arabic, and preliminary evaluation demonstrates results comparable to those for Russian.

We also intend to propose the described HT XML format as a standard markup language for syntax parse trees by building the freely available corpus for languages that lack such linguistic resources, e.g. for Russian.

References

Konstantin Anisimovich, Konstantin Druzhkin, Filipp Minlos, M. Petrova, Vladimir Selegey, and K. Zuev. 2012. Syntactic and Semantic parser based on ABBYY Compreno linguistic technologies. *In Proceedings of the International Conference "Dialogue-2012"*, 2:91–103.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. *In Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

Tatiana Erekhinskaya, Anna Titova, and Vladimir Okatiev. 2011. Syntax parsing for texts with misspellings in DictaScope Syntax. *In Proceedings*

of the International Conference "Dialogue-2011", pages 186–195.

Alexey Gladkij. 1985. *Syntactic structures of natural language in automated systems for human-machine interaction*. Science, Moscow, USSR.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. 2-nd edition*. Prentice-Hall.

Tadao Kasami. 1965. *An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report AFCRL-65-758*. Air Force Cambridge Research Lab, Bedford, MA.

Sadao Kurohashi and Makoto Nagao. 1994. Japanese dependency/case structure analyzer.

Martin Lange and Hans Leiß. 2009. To CNF or not to CNF? An Efficient Yet Presentable Version of the CYK Algorithm. *Informatica Didactica*.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. *In Proc. of the Joint Conf. on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Vladimir Okatiev, Tatiana Erekhinskaya, and Tatiana Ratanova. 2010. Secret punctuation marks. *In Proceedings of the International Conference "Dialogue-2010"*, pages 356–362.

Sergey Sharov and Joakim Nivre. 2011. The proper place of men and machines in language technology. processing Russian without any linguistic knowledge. *In Proceedings of the International Conference "Dialogue-2011"*, pages 591–604.

Jacov Testelets. 2001. *Introduction to general syntax*. RSUH, Moscow, Russia.

Svetlana Toldova, Elena Sokolova, Irina Astaf'eva, Anastasia Gareyshina, A. Koroleva, Dmitry Privoznov, E. Sidorova, L. Tupikina, and Olga Lyashevskaya. 2012. NLP evaluation 2011–2012: Russian syntactic parsers. *In Proceedings of the International Conference "Dialogue-2012"*, 2:77–90.

Leslie Valiant. 1975. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 2(10):308–314.

Zhiguo Wang and Chengqing Zong. 2010. Phrase structure parsing with dependency structure. *International Conference on Computational Linguistics*.