ОПЫТ СОЗДАНИЯ СИНТАКСИЧЕСКОГО АНАЛИЗАТОРА АРАБСКОГО ЯЗЫКА ДЛЯ ПРОМЫШЛЕННОГО ПРИМЕНЕНИЯ

Стребков Д. Ю. (strebkov@dictum.ru),
Хилал Н. Р. (hilal@dictum.ru),
Руджеймийя А. (redjaimia@dictum.ru),
Скатов Д. С. (ds@dictum.ru)

ООО «Диктум», Нижний Новгород, Россия

Ключевые слова: синтаксический анализ, синтаксический анализатор, семитские языки, арабский язык

THE EXPERIENCE OF BUILDING INDUSTRIAL-STRENGTH PARSER FOR ARABIC

Strebkov D. Y. (strebkov@dictum.ru),
Hilal N. R. (hilal@dictum.ru),
Redjaimia A. (redjaimia@dictum.ru),
Skatov D. S. (ds@dictum.ru)

Dictum Ltd., Nizhny Novgorod, Russia

We present a propagation of a hybrid approach for natural language parsing on Semitic languages on the example of the Arabic language. The hybrid approach proposes a way for acquiring dependency and constituency parses simultaneously at every step of the analysis. The result of the propagation is represented by a syntactic parser for Arabic language and the fact that the parser shows quite satisfactory results and belongs to the group of rule-based parsers actually forms scientific novelty of this article. We give a short review of Arabic Natural Language Processing (NLP) technologies and their current state and then describe steps that were required for our propagation: choosing of morphological analyzer, morphological index compression scheme, description of rule base system that is used by the parser, modifications that were needed for tuning in the core parsing algorithm. We also designate problems that we faced during the propagation and the results that we finally achieved. In the end we provide results of brief evaluation of the parser and give information on its current usage.

Keywords: syntax parsing, syntax parser, Semitic languages, Arabic language

1. Introduction

Arabic, which is the mother tongue of more than 300 million people, has received substantial attention by modern computational linguistics basing on its morphology and flexible sentences construction. The scale of Arabic-related research work is now orders of magnitude beyond what was available a decade ago [10]. At the same time, the language presents significant challenges to many natural language parsing applications for several reasons. Arabic sentences are syntactically ambiguous and complex due to the frequent usage of grammatical relations, order of words and phrases, conjunctions, and other constructions such as diacritics (vowels), which are known in written Arabic as "altashkiil" [1].

Result of the above interest can be presented as several applications. Their main goals are parsing Arabic language and providing some helper features for that. In this article we briefly describe some of such applications. Among them are three syntactic parsers (Stanford Parser [9], Berkeley Parser [16] and LFG Rule-basic Parser [2]), two morphological analyzers (Buckwalter Morphological Analyzer [7] and ElixirFM [18]), and Part-of-speech tagger (POS tagger) application developed by Stanford NLP Group [23].

Having these Arabic NLP applications available, the most significant motivation for the development of another parser are the existing NLP modules that we have developed and which are used industrially:

- Dictum's syntactic parser is based on the hybrid approach for NLP and has language-independent core component designed to support right-to-left (RTL) languages as well;
- "key-value" model for compact store of linguistic information that supports efficient access;
- opinion mining application which also has language-independent core and has been developed to deal with syntactic parser results presented in a specified format.

These applications are designed to be flexible for tuning and extending. Finally our model for syntactic rules representation supports semantic information marks.

2. Linguistic resources

In this paragraph we give a brief description of the existing syntactic modules for Arabic.

2.1. Syntax Parsers

It is necessary to mention that most accurate Arabic parsers are based on data-driven approach and assume using treebanks to learn probabilistic context-free grammars (PCFG) which assign a sequence of words the most likely parse tree [9]. Among them are Stanford Parser and Berkeley parser.

Stanford Parser is a statistical parser created by Stanford Natural Language Processing Group. Used to parse input data written in several languages such as English, German, Arabic and Chinese, it has been developed and maintained since 2002. The Arabic component takes the text as input and returns part-of-speech tagged text (the parser uses Stanford POS tagger for that) and a context-free phrase structure grammar representation:

```
Your query: هذا الرجل هو سعيد .
Tagging: هذا/DT الرجل/DTNN هو/PRP سعيد/NNP ./PUNC
Parse:
(ROOT
  (FRAG
    (NP
      (NP (DT هذا))
      (NP (DTNN الرجل)))
    (NP (PRP هو))
    (NP (NNP سعيد))
    (PUNC .)))
```

Fig. 1. Example of Stanford parser's phrase structure grammar representation

Arabic version of Stanford parser is based on the Penn Arabic Treebank (PATB) and uses phrasal category set of it [15]. Also the parser assumes precisely the tokenization of Arabic used in the PATB. There is no grammatical relations analysis available for Arabic. As for performance of Stanford parser, the dependency accuracy of the parser is around 83.5%.

Berkeley Parser is The Berkeley Natural Language Processing Group's parser; it is based on PCFG as well. Just like the Stanford parser, it returns a phrase structure representation of the input text in terms of PATB phrasal category set. Berkeley's PCFG is created using split-and-merge training strategy: splitting provides a tight fit to the training data, while merging improves generalization and controls grammar size. The resulting grammar is remarkably good at parsing [16]. According to the [9], that parser shows most state-of-the-art performance and leaves Stanford Parser behind: the accuracy of the Berkeley's parser is around 84%.

In addition to PCFG based parsers there is a rule-based parser for Arabic language, and it is the only one to our knowledge.

Arabic LFG Rule-basic Parser is the first Arabic rule-based parser available for Modern Standard Arabic (MSA). It was implemented using Xerox Linguistics Environment (XLE). Since the parser is based on LFG grammar [2], its output is represented by its special structures as shown on example below:

Your query : على الطريق رجل
Parse:

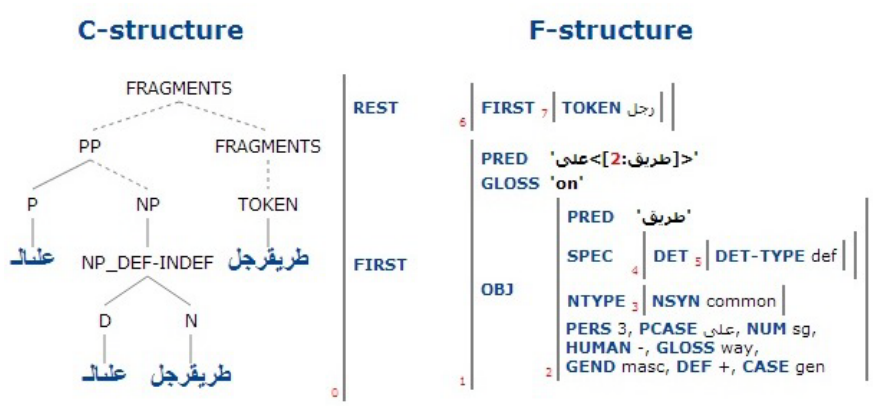

Fig. 2. LFG Rule-basic Parser's result

Figure 2 shows us parsing result in terms of phrase structure (c-structure) and grammar attribute-value pairs (f-structure).

According to the evaluation results, the accuracy of the parser is around 87% [2]. At the same time it is necessary to highlight that the result was got on a rather small corpus that consisted of 69 manually collected sentences only. M. Attia also noticed that he concentrated on short sentences and used robustness techniques to increase the coverage. All of these use hand-crafted grammars, which are difficult to scale to unrestricted data [22].

There are also Bikel Parser [12] and Malt Parser [13] which also belong to the group of data-driven parsers, so that approach is the most popular in case of Arabic language parsing.

2.2. Morphology

Morphological ambiguity in Arabic is an acute problem due to the richness and complexity of Arabic morphology.

The deficiencies of Buckwalter Morphology. Despite the fact that Buckwalter Morphology is a stem-based database and has been considered as the "most respected lexical resource", it includes a large number of entities which are not used in contemporary Arabic texts and this fact reduces the benefit of Buckwalter Morphology in analyzing the modern language. In addition, Buckwalter has some significant problems [2]:

- Absence of imperative state of almost every verb.
- Not all verbs have their correct passive form in correct tense.
- Large number of obsolete words.
- Misspelled words which lead to a massive increase in the ambiguity level for correct words.

For the reasons below, we decided to use the Elixir FM program for generating our own morphology instead of Buckwalter's:

- The lexicon's format considers the diacritics, and it means that for each of the entities ElixirFM program sets the correct vowel marks.
- In ElixirFM each verb has its correct passive form, tense and state.
- ElixirFM does orthographic analysis to get correct grammar meaning for (two, three-token) entry. For example, the word (فهم) can be interpreted as one-token "bare entity" or as tow-token "entity involved conjunction". ElixirFM includes both of these two variants in our morphology.
- ElixirFM uses the features of both word segments and the root to determine the morpho-syntactic features of the input inflected word.

Fig. 3. ElixirFM output

Figure 3 shows us analysis results of a given word: stem, transcription, grammar values and vowel reconstruction.

New morphological groups as expansion of ElixirFM issuance. Despite all advantages of the ElixirFM, the set of grammatical meanings which it gives do not cover the whole syntax of the Arabic language. In order to fill this gap we had to expand the list of grammatical meanings and add groups invented by us such as Condition, Special Function Word, Emotional Interjection and Preference name.

Also, we had to correct errors in the output of ElixirFM connected with some functional and frequency words, that were the reason for fault in the previous syntactic analysis. For example, entry (و) had two homonyms with different grammatical meanings, the first and the correct one = Conjunction, and the second erroneous = Preposition.

In case of adverbs, most of them were mistakenly identified as adjectives. To fix this error we added a check for both the case and the last letter. If the adjective was in accusative case and ended with letter Alif "ا", it became automatically adverb.

ElixirFM does not give complete information about irregular genders and does not have genders for such nouns as Broken Plurals. We assembled in lists all Broken Plurals with their numbers and genders that resulted in a full actuation of the syntactical rules with successful checking both the gender and number of nouns and, therefore receive the correct parsing.

It is necessary to mention that ElixirFM does not provide any information about control models of Arabic verbs, so currently acquisition of that very valuable information is a plan for further extension of our system.

As it was mentioned in the Introduction part, we use an efficient "key-value" model for compact storing of linguistic information (DAWG) [19]. After choosing ElixirFM as a source of morphological information, the next task was to find a set of Arabic words that could be passed to ElixirFM. It would have provided required morphological information that could have been stored in a text file. That morphological dump actually is an intermediate representation of the parser's morphology component: after being generated, it could be modified later by adding new morphological groups. As for the source of linguistic data, finally we fix on a combination of these two resources:

- Arabic Wordlist for Spellchecking that contains 9 million words [5];
- Twitter archive. We extracted all unique words from it getting around 1 million words.

Morphology data storage problem. Having that morphological dump created, we use a morphology index generation program that stores linguistic information from the dump to the DAWG [19]. The subset of grammar value and normal form being stored gave us the serialized representation of 140 Mbytes, while having 2 Mbytes for English, and 8 for Russian, so the size needed to be fixed. The structure of Arabic morphological system could be presented as a combination of two layers [3]. The former, derivation layer, is non-concatentative and opaque in the sense that it is a sort of abstraction and does not have a direct explicit surface manifestation. The latter, inflection layer, applies concatentative process by using prefixes and suffixes to express morphological syntactic features. The derivation uses interdigitation—a process when Arabic words are formed through the amalgamation of two tiers, namely, a root and a template. A root is a sequence of three consonants, and a template is a pattern of vowels with slots into which the consonants of the root are inserted:

Table 1. Interdigitation example

Pattern	**$R_1aaR_2iR_3$**			
Root	KTB كتب	QTL قتل	FHM فهم	SRB شرب
Stem	KaaTiB كَاتِب	QaaTiL قَاتِل	FaaHiM فَاهِم	SaaRiB شَارِب

The example above shows how four different stems could be formed from one pattern ($R_1aaR_2iR_3$) using corresponding roots. As for the number of different patterns in the Arabic language, there are around 500 of them [4] and it is possible to get all stems for the root by applying to it all available patterns.

Taking into account the fact that DAWG is better compressed if the keys do have many common prefixes and suffixes [19], which is not true for Arabic by default, the

decision for optimization was to split each stem on two basic parts: `root` part and `pattern` part:

$$KaaTiB \rightarrow KTB,\ R_1aaR_2iR_3$$

Secondly, ElixirFM provides information from inflection layer, i.e. shows `prefixes` and `suffixes` that were used during word formation. Due to it, we can change the conception of the key in our morphology index: for each Arabic word we store it in the following format instead of storing it as it is:

pattern|prefix|suffix|root

The figure 4 shows the advantage in compactness of such keys representation in comparison with straightforward approach when Arabic words act as keys:

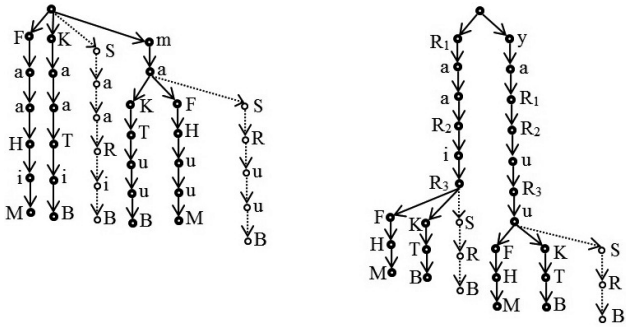

Fig. 4. Example of two approaches for keys representation

Both prefix trees are formed of four words: KaaTiB, FaaHiM, maKTuuB, maFHuuM. The left tree shows straightforward approach of keys representation, and the right one—approach that uses splitting technique mentioned above. As it could be seen from the figure 4, adding new roots (such as SRB) is more efficient from tree's size point of view.

Having splitted approach implemented, we reached the size of morphology index to be around 50 Mbytes.

3. Parser

In this chapter we mainly focus on modifications of the core of our language-independent syntactic parser that were required to get it working with Arabic language. The detailed description of the hybrid approach that was an inspiration of our parser is available in [20].

3.1. Rule base

As it was mentioned in the Annotation, our parser is a rule-based one. We used principles and approaches listed in our paper [20] to compile syntactic rules for Arabic, and at the moment their number is 193. These rules reflect and consider the specifics of the Arabic syntax. For example, in Arabic we can find, with the same frequency, (subject—verb) and (verb—subject) and this means the existence of two symmetric rules with the same priority. But with objects the (object—verb) version is more often than (verb—object) version, and therefore only one symmetric rule takes the priority which is determined by a check (!PH.InvertedLinksCount). If the rule doesn't take the priority we use check (PH.InvertedLinksCount), as mentioned in figure 5:

```
//التفاحة أكل   "apple eat"
Action+EntityObject {
  T: [Action] [Entity]
  C: LI1.Voice == VOICE_ACTIVE && ((LI2.Case == CASE_ACC &&
  !LI1.HasPersonalPron) || (LI2.Case == CASE_GEN && LI2.HasPrep)) &&
  PH2.Type != PHRASE_RELATIVE_PRON
  && PH2.Type != PHRASE_PERSONAL_PRON;
  Main: 1; L: 1=>VerbControl=>2; S: 1=>Object=>2;
  A: PH.InvertedLinksCount = 1;
}
```

Fig. 5. Description of "Action+EntityObject" rule

Arabic grammar has special categories for words that shift one or more elements of a clause into the accusative case. One of these categories is particle "Inna and her sisters" "ان واخواتها" which is usually used as subordinating conjunctions. It requires that the subject of the subordinate clause is in the accusative case and the predicate in the nominative case. In our morphology we have identified these particles in a separate group called "Special Function Word" and tried to describe it through our syntactic rules as follows:

```
//كأن المطر سيهطل - ليت الشباب يعود  "as if the rain come – wish
youth returned"
Action+Entity+SpecialFuncWord  {
  T: [Action] <> [Entity] <> [SpecialFuncWord]
  C: LI2.Case == CASE_ACC && PH1.Type != PHRASE_IMPERATIVE_ACTION
  && LI1.Gender == LI2.Gender;
  Main: 1; L: 1=>PredSubj=>2; 2=>Auxiliary=>3;
}
```

Fig. 6. Description of "Action+Entity+SpecialFuncWord" rule

Similarly, the category of verbs "Kana and its sisters" "كان و أخواتها" has the effect of shifting the predicate (خبر كان) from the nominative case to the accusative case. These verbs all denote existential states of being (or not being), becoming and remaining. We put these verbs in a group named Special Verbs and described it in syntactic rules.

Another special category of words and particles is the exclamation of wonder "اسلوب التعجب". It forms from particle (ما) and elative form which is identical with a verb form IV (af3ala). To identify the verb form IV in the whole morphological dictionary we put each adjective beginning with letter (j) in special group named Preference Name. Then, we described the exclamation of wonder in syntactic rule as follows:

```
//ما أجمله !   "how lovely it is !"
Wonder With (ما) {
  T: [PreferenceName] <> ["ما"]
  C: LI1.Case == CASE_ACC && PH2.Type == PHRASE_RELATIVE_PRON;
  Main: 1; L: 1=>Quantifier=>2;
  S: 1=>Quantifier=>2;
}
```

Fig. 7. Description of "WonderWith" rule

Also, vocative particles which come before noun are often used in Arabic and they can place noun into one of two cases (nominative or accusative). In our syntactic rules we described a vocative particle "yaa" (يا) and got a new phrase named "PHRASE_REQUIRED", that inherits the properties of one of the components PH1 or PH2 and can be used in other rule templates, as shown in figure 8 and figure 9:

```
// يا ولد   "Oh boy"
Entity+Call  {
  T: [Entity] <> ["يا"]
  C: LI1.Case != CASE_GEN && PH2.Type == PHRASE_EMOTIONAL_INTERJ;
  Main: 1; L: 1=>Auxiliary=>2;
  A: PH.Type = PHRASE_REQUIRED; LI.Gender == LI1.Gender && LI.Number
  == LI1.Number;
}
```

Fig. 8. Description of "Entity+Call" rule

```
//يا ولد ، اذهب    "Oh boy, go"
Any+Coma+Required {
  T : [Action] ({,} | {،}) [Required]
  C: LI1.Gender == LI2.Gender && LI1.Number == LI2.Number &&
  LI1.Person != PERSON_1ST;
  Main: 1; L: 1=>Parenth=>2;
  J: 1<=IsolEnd; 2<=IsolBegin;
}
```

Fig. 9. Description of "Any+Comma+Required" rule

Also, we devised a syntax rules that are able to analyze more complex structures such as the Subordinate Clause by using functions. In figure 10 we can see the relation between action and relative pronoun, which introduces a relative clause. As a result, we get entity equipped with specific function named ClauseEmbedded(PH) in the section A:

```
// الـذي نـجح    "который succeeded"
Action+Relative=Entity {
  T: [Action] <> [Relative]
  C: LI2.Gender == LI1.Gender && LI1.Person == PERSON_3RD;
  Main: 2; L: 2=>Subord=>1;
  A: PH.Type = PHRASE_ENTITY; ClauseEmbedded(PH);
}
When this function is needed we invoke it as IsClauseEmbedded(PH1)
in section C:
 //؟ من الـذي نـجح    "кто есть который succeeded ?"
Question+ActionWith(من)=Subject  {
  T: {؟}  [Entity] <> "من"
  C: IsClauseEmbedded(PH1) && PH2.Type == PHRASE_INTERROG_PART
  && LI1.Person != PERSON_1ST && !LI2.HasPrep;
  Main: 1; L: 1=>Quantifier=>2;
}
```

Fig. 10. Description of "Action+Relative=Entity" and "Question+ActionWith=Subject" rules

In general, our rule base covers all commonly used language constructs such as nominal and verbal sentence, compound sentences, conditional expressions.

3.2. The algorithm

The structure of the algorithm. Our parser uses an algorithm which can be treated as a combination of Cocke-Yanger-Kasami and Eisner's parsing algorithms ([17] and [8] correspondingly), to find dependency trees by corresponding phrase trees created by rules described above. The figure 11 shows an example of CYK's interpretation. As in usual implementation of the algorithm, it starts with upper-triangular matrix `M[2][2]` and fills its main diagonal with one-token phrases; each phrase from some cell is created from some grammatical value of corresponding token. Phrases $\texttt{Noun}_1$, $\texttt{Adj}_1$ and $\texttt{Noun}_2$ are created on that iteration.

An important moment here is the choice of destination cell for the phrase. Since tokens are numbered from right to left in case of the Arabic language, the algorithm creates phrase $\texttt{Noun}_1$ as the first one, $\texttt{Adj}_1$ and $\texttt{Noun}_2$ only after that. Therefore, if we will not take that fact into account, the phrase $\texttt{Noun}_1$ will be placed into the leftmost cell—`M[0][0]`, and that will look confusing because the phrase actually corresponds to the rightmost token of the sentence. To prevent that effect, we made a RTL-specific modification in the algorithm that adds phrases to the matrix `M` in reverse order, so phrase $\texttt{Noun}_1$ is placed into `M[1][1]` cell:

اللاعبون المخلصون

^{2}المخلصون : **$\texttt{Adj}_1$** ^{3}المخلصون : $\texttt{Noun}_2$	**$\texttt{Noun}_3$: Rule=Adj+Entity** $\texttt{Noun}_4$: Rule=Entity+Noun
	^{1}اللاعبون : **$\texttt{Noun}_1$**

Fig. 11. CYK matrix

That reverse filling process affects all diagonals of M, not only the main one.

On the next iteration the algorithm starts moving from the main diagonal to the next diagonal in upper-right direction, and we start creating phrases that cover 2 consecutive tokens. Just in that iteration we start using our rule base: from that moment for each cell that we are filling we iterate all rules from rules base, and if a rule passes template and criterion checks, we create a new phrase and add it to the cell that we are currently filling (phrases $Noun_3$ and $Noun_4$ are created that way). That iteration is the last one for our example; and phrases from upper-right edge cell cover entire input sentence.

4. Evaluation

The current evaluation of the described syntactic parser is based on the technique given in [21].

We have marked up and verified our own corpus, which consists of 300 "golden standard" sentences collected from classical texts, news and the Internet. Each sentence is unique in its syntax and lexical structure. The length of each sentence ranges in between 2–15 words. We specifically chose sentences for our corpus from various thematic sources such as banks, airlines, religion and literature. Also, we made the corpus cover all the most important language constructions, including coordinated and subordinated clauses.

The results have given the F-score of 82% UAS (unlabeled attach score [14]) with the parsing speed of ~ 2.17 Kbytes of plain text per second.

Figure 12 shows shortened assumption of hybrid tree for a sentence with isolation 2–3 and homogeneous nouns 4–6.

```
<S T="أكلت ، عند صديقي ، تفاحتين و برتقالة">
     <VW="أكلت" GV="V">
          < VW ="عند" GV ="Prep ">
               < VW ="صديقي" GV ="N />
          <Coord>
               <Group GrV="N">
                    <VW="تفاحتين" GV="N"
                    <Sepr Token="و" />
                    <VW="برتقالة" GV ="N"
               </Group>
          </Coord>
     </V>
</S>
```

Fig. 12. A hybrid tree for sentence "أكلت[1] ، عند[2] صديقي[3] ، تفاحتين[4] و[5] برتقالة[6]"
"I_ate[1] with[2] my_friend[3] two_apples[4] and[5] orange[6]"

5. Industrial usage

Described Arabic syntactic parser is currently used as an internal component of Dictum's Opinion Mining system (OMS), an application that is used as a primary component of the social media monitoring service named Kribrum [11].

The workflow looks like the following: OMS receives a review on some topic, passes it to the syntactic parser that analyses it and returns corresponding hybrid trees with semantic information marks provided by rule base system back to the OMS. Then OMS works with hybrid trees to get the summary tonality of the review, collects information about positive/negative aspects of the estimation and finally provides all gathered information to the Kribrum, so it becomes available to users.

6. Discussion

In the paper we shared our experience in building industrial-strength rule-based parser for Arabic. As it was mentioned in the Introduction, the majority of parsers for Arabic use data-driven approach, that is why good performance of our rule-based parser presents new experience in Arabic NLP.

The closest plans are the following:

- Make dictionaries that contain terms taking into account regional specificity and rebuild syntactic structure for expansion the opportunities of the analyzer.
- Add new grammatical characteristics as transitivity of verbs and animateness of nouns to make syntactic analyzing of long and complicated sentences more accurate.

References

1. *Al-Taani A., Msallam M., Wedian S.* (2010), A Top-Down Chart Parser for Analyzing Arabic Sentences, Department of Computer Science, Yarmouk University, Jordan.
2. *Attia M.* (2008), Handling Arabic Morphological and Syntactic Ambiguity within the LFG Framework with a View to Machine Translation, PhD Thesis, School of Languages, Linguistics and Cultures, the University of Manchester.
3. *Attia M., Pecina P., Tounsi L., Toral A., van Genabith J.* (2011), A Lexical Database for Modern Standard Arabic Interoperable with a Finite State Morphological Transducer, Mahlow, Cerstin, Piotrowski, Michael (Eds.) Systems and Frameworks for Computational Morphology. Second International Workshop, SFCM 2011, Zurich, Switzerland.
4. *Attia M., Pecina P., Tounsi L., Toral A., van Genabith J.* (2011), Lexical Profiling for Arabic. Electronic Lexicography in the 21st Century, Bled, Slovenia.
5. *Attia M., Pecina P., Samih Y., Shaalan K., van Genabith J.* (2012), Improved Spelling Error Detection and Correction for Arabic, COLING, Bumbai, India.
6. *Blinov A. A.* (2009), Territorial'nye varianty arabskogo litaraturnogo jazyka i ih otrazhenie v presse, PhD Thesis, Institute of Oriental Studies of the RAS, Moscow.

7. *Buckwalter T.* (2004), Buckwalter Arabic Morphological Analyzer Version 2.0, Linguistic Data Consortium, Philadelphia, USA.
8. *Eisner J.* (1998), Three new probabilistic models for dependency parsing: An exploration, In: Proceedings of the 16th International Conference on Computational Linguistics (COLING).
9. *Green S., Manning C. D.* (2010), Better Arabic Parsing: Baselines, Evaluations, and Analysis. In COLING 2010.
10. *Habash N. Y.* (2010), Introduction to Arabic Natural Language Processing, Morgan & Claypool, Toronto.
11. Kribrum service website, http://www.kribrum.ru
12. *Kulick S., Gabbard R., Marcus M.* (2006), Parsing the Arabic Treebank: Analysis and Improvements, Treebanks and Linguistic Theories 2006.
13. *Marton Y., Habash N. Y., Rambow O.* (2010), Improving Arabic dependency parsing with lexical and inflectional morphological features, Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages.
14. *McDonald R., Pereira F., Ribarov K., Hajic J.* (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).
15. Penn Arabic Treebank project, http://www.ircs.upenn.edu/arabic/
16. *Petrov S., Barrett L., Thibaux R., Klein D.* (2006), Learning Accurate, Compact, and Interpretable Tree Annotation, Proceedings of COLING-ACL 2006.
17. *Shamshad A.* (2012), CYK Algorithm, International Journal of Scientific Research Engineering & Technology (IJSRET), Volume 1 Issue 5.
18. *Smrz O.* (2007), Functional Arabic Morphology: Formal System and Implementation, Doctoral Thesis, Institute Of Formal and Applied Linguistics,
19. Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic.
20. *Skatov D. S., Gergel V. P.* (2013), Efficient Storage Structure Of A Dictionary With String Keys And Associated Values, Vestnik Nizhegorodskogo universiteta im. N. I. Lobachevskogo, Nizhnij Novgorod, Russia.
21. *Skatov D. S., Liverko S. V., Okatiev V. V., Strebkov D. Y.* (2013), Parsing Russian: a Hybrid Approach, Association for Computational Linguistics (ACL), Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing.
22. *Toldova S., Sokolova E., Astaf'eva I., Gareyshina A., Koroleva A., Privoznov D., Sidorova E., Tupikina L., and Lyashevskaya O.* (2012), Ocenka metodov avtomaticheskogo analiza teksta 2011–2012: sintaksicheskie parsery russkogo jazyka [NLP evaluation 2011–2012: Russian syntactic parsers]. In Computational linguistics and intellectual technologies. Proceedings of the International Workshop Dialogue'2012. Vol. 11 (18), Moscow, Russia.
23. *Tounsi L., Attia M., van Genabith J.* (2009), Parsing Arabic Using Treebank-Based LFG Resources, LFG09: 14th International LFG Conference, Trinity College, Cambridge, UK.
24. *Toutanova K., Manning C. D.* (2000), Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger, In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000).