A tool for morphologically ambiguous text processing

E. Klyshinsky, N. Kochetkova
Department of Computer Engineering
National Research University "Higher School of Economics"
Moscow, Russia
eklyshinsky@hse.ru

***Abstract*—The main course of preliminary natural texts processing is tagging and disambiguating texts. Hence, most of modern language tools are specified for such purposes. In our projects, we carry out a shallow syntax of untagged texts. For this purpose we developed a new query language based on regular expressions. This language allows write queries according to words' ambiguity.**

Keywords—shallow parsing; morphological ambiguity; ambiguous texts; query language

I. INTRODUCTION (*HEADING 1*)

There is two ways of creating software tools for natural language processing. The first one is creating widely used common tools, which are applicable for a variety of languages. The second one is considering of properties and peculiarities of the given language. In this paper, we have chosen the second way during the development a software tool for extraction of syntactically connected group of words.

Texts written in the Russian language have up to 60-80% words that are unambiguous by part of speech [1]. As result, there is possibility extracting some information without text disambiguation. The amount of extracted information could be even more, if some types of ambiguity can be considered. However, such existing modern tools for natural language processing as NLTK [2] and Stanford Parser [3] was developed according to the standard model. This model states that the texts should be tagged and disambiguated first. The precision of such tools is pretty high, however it brings some mistakes in the processed data. That is why we decided create a new software tool for finding word sequences according to ambiguous tagging. Such tool should be useful in linguistic investigations of words ambiguity, extraction of tuples of syntactically connected words and for some other purposes.

The goal of our project was to sample a database of syntactically connected words and create a website for L2 learners of the Russian language (for more details see [4]). Unlike such corpora as Sketch Engine [5] and Russian National Corpus [6] offering examples of correctly written text containing given words, this project is aimed to sampling combinations of syntactically connected pairs. Unlike to RNC Sketches [7], we do not need a parser, however we need corpora of bigger size and have less recall. Note that our database is suitable for linguistic research as well (see, e.g., [8]).

This project is supported by RFH grant 15-04-12019.

This paper describes a tool we have created in order to sample the mentioned database. The tool provides query language to untagged corpora written in the Russian language. The query language allows to express ambiguous features of the languages and use them while searching in the corpus.

II. THE QUERY LANGUAGE

As it was claimed by Manning [9], a regular automaton allows extracting the subcategorization frame of English verbs for such phrases, which have clear syntactical structure. However, direct implementation of this method for Russian texts leads us to a high rate of mistakes because of homonymy. Following Manning, we selected regular automata as a theoretical background of our query language. The idea was formulating queries to the sampled text corpora, processing corpora by such queries, and gathering results into a database.

We defined a number of properties for our query language. The first property is morphological homonymy of different types. As it was shown in [1], there is three main types of homonymy, i.e. by initial form, by part of speech, and by grammatical features. Therefore, a user should have possibility to express all of them and their five possible combinations. The second property is the possibility to write rules that are excluding some features. A user should have possibility write a query that excludes some initial forms (e.g. any verb but *be*), parts of speech (e.g. any word but verb), and grammatical features (e.g. any case but nominative). The third property is words coordination and agreement, when a user is able express a query where, for example, an adjective agrees with a noun by its gender, number, and grammatical case. Finally, the fourth possibility is features alternation, when a user is able to write a list of possible values for such features, as initial forms, values of grammatical features, and parts of speech. Note that some combinations are inhibited. One cannot find a word that is ambiguous between a noun and a verb both in nominative case because there is no grammatical case for verbs. Thus, all grammatical features should be expressed accordingly to previously defined part of speech.

Using the defined properties, we have developed a query language. It includes such main properties of regular automata as iteration and alternations. We also had included some extra features, thus the resulting language have a different syntax.

Every token is placed in square brackets. It could have initial form, part of speech and grammatical features; however, any part could be omitted. These three parts of token could be

separated by semicolon. Initial form should be placed in inverted commas. Grammatical features are described as *feature name=feature value*, the order of features is arbitrary. All grammatical features, if more than two, are separated by comma. The following example demonstrates a noun *vyvod* (*conclusion*) in plural form.

[‘вывод’; noun; number=pl]

Note that the list of grammatical features and their values is dependent of the language model. Thus, it can be changed depending on the used vocabulary.

Our format implies usage of such standard features of regular automata as Kleene star, alternations and grouping. The following example demonstrates zero or more occurrences of any adjectives combined with preceding adverbs; the group is followed by a noun.

([;adv ;]? [;adj ;])* [;noun ;]

To use any value but the mentioned one, a user should mark the value by a minus sign. The next example describes a noun in any objective grammatical case (any grammatical case excluding nominative).

[;noun ; case= -nom]

And here is an example defining any verb excluding *byt'* (*be*) in any form excluding infinitive.

[‘-быть’; verb; form= -inf]

Such possibility allows writing rules more easily. Instead of writing all allowed grammatical cases, a user can write just one suppressed case, or, as it will be shown hereinafter, just some of them.

The Russian language is an inflectional one with frequently used words agreement. For example, syntactically connected adjective and noun should be agreed by their gender, number and grammatical case; or more accurately, gender, number and grammatical case of an adjective are governed by a noun that is a syntactical head for this adjective. In our query language, we use a plus sign to express this feature. The following example describes a prepositional phrase according to the fact of words agreement.

[;prep;] ([;adv;]? [;adj; gender +, number +, case +])* [;noun; gender +, number +, case +]

Note that a grammatical feature marked with the plus sign should agree with every other grammatical feature having the same name. It means that we ought not to mark any feature with the given name by the plus sign. The plus sign means that this grammatical feature for the given word should agree with other words, while other words can have any value of the feature with this name. For example, it is obvious that a noun in prepositional phrase could not be in nominative case. Thus, for eliminating mistakes we can rewrite our query.

[;prep;] ([;adv;]? [;adj; gender +, number +, case +])* [;noun; gender +, number +, case= -nom]

The example above express our intuition that a noun should not be in nominative case and adjectives should be governed by the noun. It also shows that grammatical case of adjectives should be exactly the same as the found in the noun.

A user can use minus mark to express the fact that a feature with this name is not agreed with others.

Our query language allows alternations in every part of a token. Alternations are denoted by the | sign. If one need a prepositional phrase in genitive or instrumentative case, then he or she can write the following query.

[;prep;] ([;adv;]? [;adj; gender +, number +, case +])* [;noun; gender +, number +, case= gen | instr]

We can also alternate word’s initial form. The next example extracts verbs of existence combined with adjective.

[‘быть | стать | становиться’;verb;] [;adj;] [;sign;]^

The circumflex sign denotes here the end of the sentence.

Now, have been described all auxiliary properties of our query language, we can start with the main property – queries with grammatical ambiguity.

The morphological analysis of a word token w includes its lemma, its part of speech, and its grammatical features. The list of grammatical features varies depending on the language and the word’s part of speech. Let us define a word form v as a tuple $v=<l, \pi, \mu>$, where l is the lemma of this word form, π is its part of speech (POS) and μ is its set of grammatical features. The result of morphological analysis of a word w is a set of word forms $\varphi(w) = \{v_1, v_2, \ldots v_k\}$, where each v_i is a distinct word form. If the word w is not present in the vocabulary, $\varphi(w) = \emptyset$ (i.e. $k = 0$), otherwise $k > 0$. If $\varphi(w)$ contains more than one word form ($k > 1$), the word w is ambiguous.

By default, a token of regular expression is applicable to a word if it is applicable to at least one word form of this word. We can make this requirement stricter claiming that the token should be applicable to all word forms of this word. For this purposes we use an exclamation mark. We can rewrite the previous example using this new feature.

[! ‘быть | стать | становиться’;verb;] [;adj;] [;sign;]^

This will change obtained results. The Russian verb *stat'* (*to become*) has an ambiguous form *stali* that means the past plural of *to become* and plural nominative of *steel*. Therefore, the last example will consider any form of the verb *stat'* (*to become*) excluding its form *stali*. Usage of non-homonymous forms of words increases the precision rate of query but reduces its recall.

As it was mentioned above, different parts of speech could not share the same grammatical features. That is why we separate part of speech alternations.

The first possibility is to apply the logical OR operator to the list of parts of speech. This fact is defined by the | mark. In this case every possible part of speech is written together with its grammatical features, if any, but separated by | sign from other parts of speech. Let us find prepositional phrases with three or more adjectives where adjectives in the central

position(s) could have ambiguity with noun. It should be written as following.

[;prep;] [! ;adj; gender +, number +, case +] [;adj; gender+, number+, case+ | noun; gender- | noun; number-| noun; case-]+ [! ;adj; gender +, number +, case +] [! ;noun; gender +, number +, case +]

To be sure in our results, we state here that adjectives in the border positions should be unambiguous by the part of speech (they are only adjectives with agreed features). The same we can say about the noun.

Note that the words in the middle can have any possible part of speech but should meet at least one requirement from the list. I.e. it can be just a non-agreed noun but not an agreed adjective. To improve such situation we can use logical AND operator (& sign) that is used at the same position. The following example means that we are looking in the central position for words that are ambiguous between agreed adjectives and any nouns only.

[;prep;] [! ;adj; gender +, number +, case +] [;adj; gender+, number+, case+ & noun;]+ [! ;adj; gender +, number +, case +] [! ;noun; gender +, number +, case +]

However, the noun in the central position can be surprisingly agreed with adjectives and can find a noun phrase preceded by a prepositional phrase. So, the better option is use the next query.

[;prep;] [! ;adj; gender +, number +, case +] ([! ;adj; gender+, number+, case+ & noun; gender-] | [! ;adj; gender+, number+, case+ & noun; number-] | ! [;adj; gender+, number+, case+ & noun; case-])+ [! ;adj; gender +, number +, case +] [! ;noun; gender +, number +, case +]

Unfortunately, it is much longer and harder to read.

Note that our query language includes templates for constructing an output. However, it is not a point of interest for this article; so, it would not be discussed here.

III. Obtained results

In our experiments we used the following untagged Russian corpora: fiction taken from LibRusEc.ru (ca 15 bln word tokens), news wire downloaded from different web sites (about 1 bln word tokens), popular science news wires (about 150 mln word tokens), and scientific texts (PhD thesis and its summaries, conference proceedings, scientific journals in different domains –about 100 mln word tokens in total). Thus, the size of our corpora is comparable with the size of Sketch Engine RuTenTen corpus, which is 14.5 bln of tokens, but not so good balanced.

We had formulated a set of queries that are concerning in words' ambiguity. We selected part-of-speech unambiguous words with clear syntactical structure. Thus, the developed tool helps us to run a fast shallow parsing (about 15 mln words per minute).

The collected results were as big as ca 79 mln verb+prep+noun, ca 134 mln verb+noun, ca 35 mln adj+noun co-occurrences and some other combinations. We failed to gather information for a variety of connections, e.g., on noun+noun because of a very high rate of mistakes in the output (in about 50% of extracted combinations both nouns are connected to the same verb).

The constructed corpus contains 7.5 mln of verb+noun(+preposition) unique combinations and 2.3 mln of noun+adjective. Finally, we gathered information on combinations of 23000 verbs and 57000 nouns, and combinations of 39000 nouns and 31000 adjectives. We have calculated the frequency of occurrence for every word combination: every co-occurrence is connected with a set of examples selected from the text.

All sampled co-occurrences was placed in the Corpus of Syntactical Co-occurrences (CoSyCo). We developed a web-interface to this database; however, it is still under development. For more details, consult [4] or web-site http://cosyco.ru. CoSyCo contains connections between word forms and their initial forms, thus, a user is able to see a word form and its connections. In the future we are planning add two-way links between head and tail words; currently user is able just find a head word and receive a list of dependent words.

Unlike a corpus for researchers, an educational corpus can have less recall but higher precision rate. Such a corpus should be tagged using the information from a vocabulary sampled by linguists but not automatically generated from texts using forecasting technique. Types and initial forms should not be predicted stochastically because this procedure has a relatively higher error rate in comparison with manual tagging. The corpus interface should provide convenient access to lists of words syntactically connected with the selected one, as well as statistical information on co-occurrences' frequencies. It should preferably allow switching between co-occurrences and word forms, change the focus from the head word to the dependent one and vice versa.

Note that our corpus contains rather usage than syntactically correct occurrences. Sometimes both incorrect usage and academic norm are equally frequent. This situation reflects the fact that this phenomenon should be investigated and learned "in vivo" but not "in vitro". Our corpus contains rare syntactically but not semantically correct combinations. In this case, a student can consult frequencies of combinations and make a proper choice.

Even the most developed tool, the Sketch Engine, does not provide all this functionality. Such operations as ordering by frequency of occurrence need extra calculations, and in a big corpus they take a long time. The list of dependent words is previously shortened and cannot be ordered by frequency. Predicted words can be crucial in a research project, but they may mislead even an intermediate-level learner. Because of these features the Sketch Engine should rather be classified as a corpus for researchers than one for a student learning a foreign language and not familiar with the corpus linguistics.

Some of the recent tools, such as Sketch Engine and CoCoCo (http://cococo.cs.helsinki.fi/), are fetching co-occurrences in on-line mode; a user sends his request, and the server calculates set the co-occurrences according to this

request. This approach economizes hard disk space, however processing of a user's request can take a long time. Following such corpora as Russian National Corpus (RNC), we are precalculating any possible output; as result a user is able to navigate through the list of words and receive their connection in a short time. Among other, we do not use word prediction during the analysis of our corpora. Thus, the list of words is shorter than in Sketch Engine or RNC, however our database has less rate of mistakes.

Unlike in such corpora as Sketch Engine, we organized the structure of the corpus as a tree. A user enters a word he or she is interested in, and the site demonstrates the list of words connected with this one by the selected type of connection. In the case of a noun, it could be a list of governed adjectives or governing verbs (and prepositions) depending on the user choice; in the case of an adjective, it could be a list of governing nouns or connected adverbs. When selecting a combination, the user receives a list of sentences containing this co-occurrence. The source of the selected sentence can be found using Yandex search engine by clicking the proper link. In case of noun+adjective combinations, the user is also provided by forms of adjectives.

All results can be filtered by subcorpora in addition to the information on the subcorpora genres. It helps a user detect if that is a professional term or regular word combination.

We also used this tool in a linguistic research project. We considered the word *gordy* (*proud*) and its usage in texts of different genres. For more detail, see [8].

IV. Conclusion

Widespread natural language processing tools do not provide possibility to formulate queries that are concerning in words ambiguity or unambiguity. That is why we had developed a new software tool for processing of untagged texts written in the Russian language. Tis tool demonstrate its speed and practical significance during its usage in practical projects. However, the tool needs some improvement because of it is difficult to write some queries. We are planning include more languages in this project.

We plan to increase the size of the used corpora and attach some new regular expressions. This will allow us to broaden the amount of gathered information. Another problem is the purity of corpus. We have found that LibRusEc corpus contains texts in Ukranian, Belorussian, Bulgarian and Serbian languages. Both news wire and fiction subcorpora are containing text duplicates, thus multiple versions of the same sentence should be eliminated. Such corpora as GICR [10] are processing this information but in case of an educational corpus it hinders the student's perception. Finally, we used the simplest tokenizer and parser in our experiments; thus, we have to completely remaster our corpus and extract new database of co-occurences. The current version can be used as a learner's corpus just after all these changes; however, it can be used as a researcher corpus in the current state.

References

[1] Klyshinsky, E., Logacheva, V., Nivre, J. Ambiguous Words in European Languages [Raspredelenie neodnoznachnyh skov v nekotoryh evropeiskih yazykah]. 2015. In Proc of Corpora Linguistics 2015. (in Russian)

[2] Bird, S., Klein, E., Loper, E. Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit – http://www.nltk.org/book/

[3] Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In LREC 2006.

[4] Klyshinsky, E.S., Ermakov, P.D., et al. The Corpus of Syntactic Co-occurences: the First Glance. 2015. In Proc of AIST 2016.

[5] Kilgarriff, A., Rychly, P., Smrz, P., Tugwell, D. The Sketch Engine, In Proc. of EURALEX 2004.

[6] Lashevskaja, O., Plungian, V. Morphological annotation in Russian National Corpus: a theoretical feedback. In Proc. of 5th International Conference on Formal Description of Slavic Languages (FDSL-5), 2003

[7] RNC Sketches: http://ling.go.mail.ru/synt/

[8] Lukashevich, N.Y., Klyshinsky E.S., Kobozeva I.M. 2016. Lexical Research in Russian: Are Modern Corpora Flexible Enough? In Proc. of Dialogue 2016.

[9] Manning, D. Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In Proc. of the 31st Meeting of ACL, 1993.

[10] Belikov, V., Selegey, V., Sharoff, S. Preliminary considerations towards developing the General Internet Corpus of Russian. [Prolegomeny k proektu General'nogointernet-korpusa russkogo yazyka] In Proc. Int. Conf. on Computational Linguistics "Dialog", 2012. (in Russian)