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

Moscow, June 17–20, 2020

AN ABSTRACT MODEL OF SEARCH INDEX QUERY IN THE RUSSIAN NATIONAL CORPUS

Morozov D. (morozov@ruscorpora.ru)
The Institute for Information Transmission Problems (Kharkevich Institute), Moscow Institute of Physics and Technology (National Research University)

Gladilin S. (gladilin@iitp.ru)
The Institute for Information Transmission Problems (Kharkevich Institute)

The paper discusses the so-called "bag problem," which affects the search accuracy in the Russian National Corpus (RNC). Solving the problem requires a change of the search index data scheme used in RNC, which in its turn requires a significant refactoring of the RNC program code. The basis of such a refactoring is proposed to be an abstract model of the search index query, which allows us to separate the query formation from the query implementation. An experiment was carried out in which one of the RNC system program modules was decomposed, which confirmed sufficient expressiveness of the constructed model. Directions of further work are determined.

Keywords: corpus linguistics, Russian National Corpus, lexical-grammatical search system, abstract search query model

DOI: 10.28995/2075-7182-2020-19-1109-1116

1. Introduction

Language corpora can be distributed in the form of full texts with markup and documentation, allowing the researcher who has certain programming skills, to search over these texts according to arbitrary conditions and to obtain the required statistics. For example, such distribution approach is used for the SynTagRus corpus containing Russian texts with morphosyntactic markup [2]. However, for the Russian National Corpus (RNC), a different approach was chosen. RNC is located on an Internet server, and a lexical grammatical search engine with a web interface is implemented to access it. Such an approach, first, eliminates the need for the researcher to download huge amounts of data (RNC contains more than 280 million words in the main corpus), and secondly, it allows the inclusion of copyrighted texts to the corpus. Such texts cannot be published without the consent of the copyright holder, but organizing a search interface and obtaining statistical data on these texts is permitted.

The lexical grammatical search engine with a web interface, implemented in RNC, is a software environment consisting of:

- web server,
- search index engine,
- RNC software that implements user queries through calls to search index procedures.

Currently, two versions of RNC are operating in parallel: the old one, which is being removed from service, and the new one being introduced. These versions use different software products for the search index engine (the old one is maintained by Yandex.Server, and the new one by “cloud” Yandex.Search), but with the same data scheme. In this study, we analyze the shortcomings of the search, caused by the chosen data scheme and simultaneously present in both the old and the new versions of the RNC, and propose an approach which is expected to minimize them.

2. Problem statement

Search in RNC allows one to select words or phrases in the corpus that satisfy certain conditions. However, the search results do not always exactly match the query conditions. The most common cause is homonymy disambiguation. In disambiguated texts, grammatical and semantic features are not specified in the markup, but are assigned to words automatically, based on the dictionary and rules. In this case, homonyms get several possible *parses* that have different lemmas and/or different attributes. As a rule, only one of these parses is correct, but it is not possible to select it from the whole set automatically (without performing the procedure of reverting disambiguation), therefore, the search engine finds all words, at least one of whose parses satisfies the query conditions. Such inaccuracy of the search cannot be corrected without the complete relapse of disambiguation in all texts.

However, the RNC search engine commits mistakes of a different nature, for the correction of which removal of disambiguation is not required. Consider, for example, the query `печь,S,nom,sg`, referring to all nouns in the nominative case with the lemma “печь”. The result of this query search in a non-disambiguated subcorpus (both in the old and in the new version of RNC) will yield the word “пекло”. This is a mistake, because even though one of the parses of the word “пекло” really matches the lemma “печь”, in this case “пекло” is not a noun, but a verb of the neuter gender in the past tense. Another parse is the noun “пекло” of the singular nominative case, but its lemma is not “печь”.

Consider another example: for `V, t:move` (verbs with the semantics of motion), the word “полет” will be found. This is also a mistake: the word “полет” really has a semantics of movement, but it is not a verb, but the verb “полоть” in the third person singular has no semantics of movement.

Let us now try to search for the word “пирога” in all cases, except for the nominative. Query `пирога, (dat | voc | gen | gen2 | acc | acc2 | ins | loc | loc2 | adnum)` (a word with the lemma “пирога” in one of the listed cases, all but the nominative are listed) in non-disambiguated subcorpus will not give us the desired result: the word “пирога” will also be displayed, because it has the homonym “пирога”, which is in one of cases listed in the query (genitive).

If you use the “minus” (negation) query operator, supported by RNC, and build the query differently: `пирога, -nom` (a word with the lemma “пирога” **not** in the nominative case), then the old and the new versions of RNC system will behave differently. The old version will interpret `-nom` as a synonym for `(dat | voc | gen | gen2 | acc | acc2 | ins | loc | loc2 | adnum)` and will produce the same result. The new one will give the correct result: it will not display the word “пирога” in the nominative case. However, this tool does not always work: to search for the word “пирог” not in the nominative case in a non-disambiguated subcorpus, the query `пирог, -nom` in the new version of RNC will exclude the word “пирога” from the output because it has the homonym “пирога” in the nominative case. In this case, the query `пирог,(dat|voc|gen|gen2|acc|acc2|ins|loc|loc2|adnum)` will work more correctly.

Another example: consider the search for homonyms using the zero-distance function of RNC. The query `S at a distance of 0 from V` will find all the words, one parse of which is a noun, and another a verb (for example, “печь”). However, we consider the case when the desired properties of the two parses coincide. Suppose that we want to find verbs in the active voice that have two different parses (for example, the word form “есть” has a parse with the lemma “есть” and a parse with the lemma “быть”). The corresponding query `V,act at a distance of 0 from V,act` does not give the desired result: it simply finds all the verbs in the active voice, having failed to distinguish the cases when the desired tuple of properties refers to just one parse from cases when this tuple is present in two different parses simultaneously.

In all the examples above, the problem consists in the fact that the properties of each parse of the homonym affect the search by the properties of its other parses. The corpus markup and the dictionaries by which the markup is automatically enriched contain all the information necessary to organize the correct search in all of the above cases. However, the RNC current lexical grammatical search engine (both in the old and the new version) does not use it to a sufficient extent. This problem is known to the RNC developers and computational linguists as the “bag problem”. In the present work, we analyze this problem and search for approaches to its solution.

3. “Bag problem” analysis

The “bag problem” emerged as a result of a discrepancy between the RNC text markup model and the data indexing model of the RNC lexical grammatical search engine.

In accordance with the markup model, four levels of text structure units are distinguished:

- the entire text,
- sentence or another in-text unit (dialogue exchange, stanza, …),
- word,
- parse of the word.

The units of each level can be attributed to the properties which can be searched for. Properties attributed to the entire text (for example, name, year of publication, …) and in-text units (for example, the speaker’s age and gender are dialogue line properties in the oral corpus) are used to select a subcorpus. Grammatical and semantic properties are attributed to individual parses, not to the non-disambiguated word.

At the same time, for technical reasons, currently the search index model (both in the old version of RNC system and in the new one) does not contain the level of word parses and assigns grammatical and semantic attributes not to an individual parse, but to the whole set of possible parses of the word [1]. It causes the “bag problem”. For example, the word “пироги” attributes the properties `pl, nom, m, sg, gen, f, acc` (plural, nominative, masculine, singular, genitive, feminine, accusative) the first three of which relate to parse “пироги” ‘pastries’, and the next three ones – to parse “пироги” ‘pirogues’. The last attribute of the list relates to parse “пироги” as the plural of the feminine accusative, and to parse “пироги” as the plural of the masculine accusative.

A simple search over such a list of attributes would give completely irrelevant results (e.g. the word “пироги” would be found under the condition “singular, nominative case”, because both attributes are present in the list). Therefore, to solve the “bag problem”, an approach using *combined attributes* was proposed [1]. Within this approach, attributes in the search index store not only individual grammatical (semantic, …) properties, but also their various combinations. So, for a parse containing three properties `S, nom, sg`, the following attributes are stored:

- `S`
- `nom`
- `sg`
- `S, nom`
- `S, sg`
- `nom, sg`
- `S, nom, sg`

When searching in the index, it is required that the search query correspond not to the individual properties of the desired word, but to a tuple composed of the combination of properties.

This approach is characterized by an exponential growth of the number of stored attributes from the number of properties attributed to the parse ("combinatorial explosion"): for a parse containing n properties, $2^n - 1$ stored attributes are required. So, to store 10 properties, 1023 stored attributes are required, for 20 properties—more than a million ones.

Thus, this approach is extremely inefficient in terms of computer memory consumption. Therefore, to save memory, all kinds of combinations of grammatical properties and all kinds of combinations of semantic properties are stored separately, but combinations of grammatical and semantic properties are no longer stored. Also, the combination does not include lemmas (they are also stored separately) and additional properties. Therefore, when searching, the lemma of one parse may be matched with the grammar of another and the semantics of the third one, which causes mistakes described in **Section 2**.

4. Suggested approach

To solve the "bag problem", one needs to change the search index data model so that it matches the text markup model. As a result of such a change, it will be possible to fully use the information present in the markup when searching.

The change of the data model is nontrivial because of two factors. First, at present we do not know which software and in what configuration will be able to ensure an efficient search with the required data model in an index containing hundreds of millions of words. The best opportunities for choosing an optimal data scheme are provided by relational databases. Many language corpora (for example, the Bimodal Corpus of Russian-Turkic Bilinguals Speech (RuTuBiC) [6]) use relational databases, since they provide the widest possibilities for choosing a data scheme. However, it is necessary to further investigate whether their use in such a huge corpus as RNC is possible.

Secondly, RNC is a complex software product, the basic assumption in the development of which was the model for representing data in the form of combined attributes, described in **Section 3**. Therefore, changing the model will accordingly require a substantial refactoring of the RNC program code.

Thus, without such refactoring, it is impossible to conduct full-scale experiments and compare the efficiency of various data schemes and their suitability for the RNC. At the same time, the refactoring of the program code requires the selection of a specific model of the search index and search queries.

Our approach is to develop an abstract search query model that allows us to describe the search task with all the information presented in the markup, but not attached to any specific implementation and configuration of the search index. This will allow us:

- first, to rebuild the RNC software so that it is not fixed to specific features of a particular search implementation, but, rather, is determined by an abstract model free of defects in specific implementations;
- secondly, to build, in the future, multiple specific implementations of the search index as part of the RNC system and experimentally compare them for search accuracy and speed.

We have developed an abstract search query model consisting of the following components:

- a reference to the corpus in which the search is being performed;
- search conditions imposed on the text as a whole;
- search conditions imposed on the data concerning the authors of the text;
- search conditions imposed on an individual sentence;
- search conditions imposed on the distance between words in a collocation;
- search conditions imposed on the parse that must be present in the word;
- an indication of the required grouping of search results;
- an indication of the required sorting of search or aggregation results;
- an indication of the required selection of the display window (page in a multi-page output).

Search conditions and indications of the required grouping and sorting are specified in terms of key-value attributes and a fixed set of operations performed on them. Depending on which component the attribute is used in, it can be search, sorting and/or grouping.

The following types of search attributes are introduced:

- date interval (example: date of creation of the text);
- text string (example: word form);
- string list: text allowing to be searched for individual words (example: text title);
- subset of a predetermined set (example: a text genre is one or more values from a predefined list of genres).

For each attribute type in the model, we determine the possible *matchings* of the value S specified in the search query and the value A of the attribute in the search index (**table 1**).

Table 1: Mappings allowed for various attribute types

Data type of attribute	Valid mappings
String	S is equal to A
	S is the prefix of A
	S is a suffix of A
Date interval	non-empty intersection of S and A
	exact occurrence of A in S
Subset	non-empty intersection of S and A
	exact occurrence of S in A
String list	non-empty intersection of S and A
	exact occurrence of S in A

In addition to the *core* attributes presented in the RNC markup and described in [4], [5], [7], [8], additional *calculated* attributes are introduced into the model (for example, for sorting that takes account of several factors).

The model currently used in RNC does not provide search accuracy due to the “bag problem”, therefore we consider it as an approximate implementation of the abstract model. This allows us to propose the following RNC software conversion algorithm to solve the “bag problem”:

- the search algorithms currently present in the system are formalized as the initial implementation of the abstract model;
- system modules that do not directly interact with the search index are refactored to use an abstract model instead of a concrete one;
- new concrete implementations of the abstract model are developed, which are embedded in the system and compared for accuracy and speed with the initial implementation.

We have tested the proposed approach on a module for processing complex queries in lexical and grammatical search. This module is a small part of the RNC software environment. The module was refactored and divided into two isolated parts: the first builds an abstract query, and the second provides its concrete implementation using the Yandex.Search engine.

Extensive testing has confirmed that queries to the Yandex.Search engine generated by our modification of the module are equivalent to the queries generated by the previous one, except for specific cases of correction “bugs” presented in the old version of the module. Therefore, the results and speed of search queries execution remained unchanged.

The developed implementation of the module has been successfully tested and included in the new revision of RNC system as of May, 2020. As a result, it became possible to replace the specific implementation of the search index with another module without rebuilding the rest of the system.

5. Conclusion

We proposed an approach to RNC software refactoring, aimed at solving the “bag problem”, which affects the search accuracy, and performed a first test of this approach.

The idea underlying our approach is to decompose the program code into two isolated parts, the first of which builds an abstract search query, and the second implements the query using a specific search engine.

A model of the abstract query was proposed and an experiment was conducted, which consisted in the decomposition of one of the program modules of the RNC system.

The experiment confirmed that the developed abstract model is indeed expressive enough to treat all the queries that may arise in the lexical/grammatical RNC search system, and to separate query formation from query implementation.

Further work will be aimed at a similar decomposition of other parts of the system and the looking for a software configuration that can ensure the implementation of abstract queries with the necessary accuracy and speed.

An approach similar to that described is also planned to be applied to improve the RNC user interface. The modern JavaScript single-page approach to building web interfaces is used, for example, in the Multichannel corpus [3]. It seems that the refactoring of the RNC system in accordance with this approach should also be carried out through the development of an abstract model (in this case—an abstract model of a user query).

References

1. *Abroskin, A. A.:* Poisk po korpusu: Problemy i metody ikh resheniya. Nacional'nyj korpus russkogo yazyka: 2006–2008. Novye rezul'taty i perspektivy. 277–282 (2009).
2. *Inshakova, E. S. et al.:* SynTagRus segodnya. Proceedings of the V.V. Vinogradov Russian Language Institute. vol.21, 14–41 (2019).
3. *Korotaev, N. A. et al.:* SEARCH in a multichannel corpus: DEVELOPING an online version. Computational Linguistics and Intellectual Technologies Papers from the Annual International Conference "Dialogue" (2019). Issue 18. Supplementary volume, 72–80 (2019).
4. *Kretov, A. A.:* Analiz semanticheskih pomet v nkrya. Nacional'nyj korpus russkogo yazyka: 2006–2008. Novye rezul'taty i perspektivy. 240–257 (2009).
5. *Rakhilina, E. V. et al.:* Zadachi i principy semanticheskoj razmetki leksiki v nkrya. Nacional'nyj korpus russkogo yazyka: 2006–2008. Novye rezul'taty i perspektivy. 215–239 (2009).
6. *Rezanova, Z. I. et al.:* THE bimodal corpus of russianturkic bilinguals' speech (rutubic). Computational Linguistics and Intellectual Technologies Papers from the Annual International Conference "Dialogue" (2019). Issue 18. Supplementary volume, 200–210 (2019).
7. *Savchuk, S. O.:* Metatekstovaya razmetka v nacional'nom korpuse russkogo yazyka: Bazovye principy i osnovnye funkcii. Nacional'nyj korpus russkogo yazyka: 2003–2005. Rezul'taty i perspektivy. 62–88 (2005).
8. *Sitchinava, D. V.:* Obrabotka tekstov s grammaticheskoj razmetkoj: Instrukciya razmetchika. Nacional'nyj korpus russkogo yazyka: 2003–2005. Rezul'taty i perspektivy. 136–154 (2005).