Using FLOSS for Storing, Processing and Linking Corpus Data

Damir Mukhamedshin, Olga Nevzorova(✉), and Alexander Kirillovich

Kazan Federal University, Kazan, Russia
damirmuh@gmail.com, onevzoro@gmail.com,
alik.kirillovich@gmail.com

Abstract. Corpus data is widely used to solve different linguistic, educational and applied problems. The Tatar corpus management system (http://tugantel.tatar) is specifically developed for Turkic languages. The functionality of our corpus management system includes a search of lexical units, morphological and lexical search, a search of syntactic units, a search of N-grams and others. The search is performed using open source tools (database management system MariaDB, Redis data store). This article describes the process of choosing FLOSS for the main components of our system and also processing a search query and building a linked open dataset based on corpus data.

Keywords: Corpus linguistics · Corpus manager · Linked open data

1 Introduction

In this paper, we discuss the development of the corpus management system for the Tatar National Corpus "Tugan Tel" [1]. The corpus is organized as a collection of texts covering different genres, such as fiction, news, science, official, etc. A text consists in sentences (called contexts). The words from a sentence are provided with morphological annotation. The annotation of a word includes the lemma, POS and the sequence of grammatical features expressed by the affixes.

Access to the corpus is provided by the corpus management system. The system allows users to search for word tokens with a specified full form or lemma and grammatical features. The search results are represented as a list of the sentences containing the found word tokens. Figure 1 shows the search result for the word *kuman* 'book' in the plural number and the genitive or the directive case. The tooltip under one of the found tokens contains its morphological annotation.

The general architecture of the corpus management system is represented at Fig. 2. This model includes three main components: a web interface, a search engine, and a database. The system imports the annotated texts from the corpus annotation tool, and exports them to a triplestore of the LLOD publishing platform.

Development of the corpus management system cannot begin without a clear understanding of which tools will be used to store and process corpus data. Therefore, one of the important factors when choosing and using tools for data storing and processing is

Published by Springer Nature Switzerland AG 2020
V. Ivanov et al. (Eds.): OSS 2020, IFIP AICT 582, pp. 177–182, 2020.
https://doi.org/10.1007/978-3-030-47240-5_17

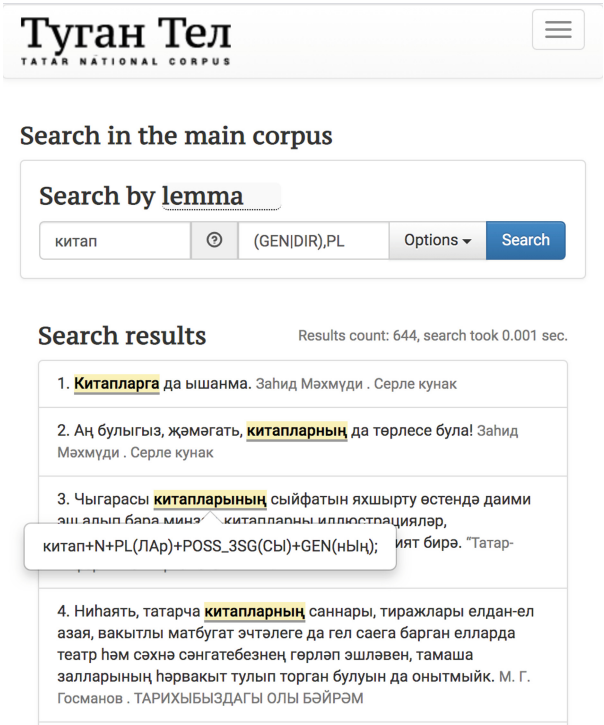

Fig. 1. The corpus manager GUI

for us their open source code, which is fundamental to ensure transparency and flexibility of the tool.

When choosing tools for processing and storing corpus data, we were faced the task of finding a set of FLOSS to ensure high speed of performing search queries (no more than 0.1 seconds for direct search queries, no more than 1 second for reverse search queries), wide search capabilities (at least direct and reverse search, search in parts of word forms and lemmas, mixed search, phrase search), and the possibility of further growth of system performance and functionality.

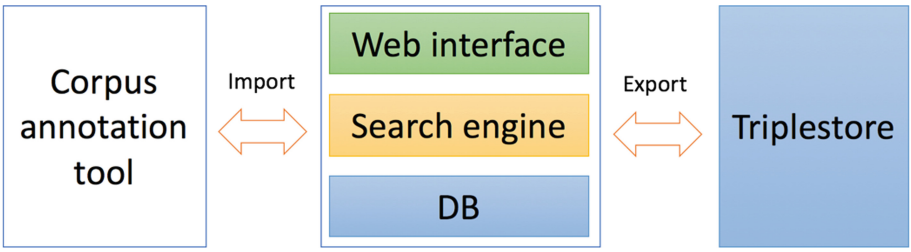

Fig. 2. Abstract structure of the corpus data management system

In Sect. 2 we describe using of FLOSS in the corpus management system, and in Sect. 3 we describe using of FLOSS the LLOD publishing platform.

2 Using FLOSS in Corpus Management System

To choose FLOSS for storing and processing of corpus data, we formed a list of the most important selection criteria:

1. Performance. The solution should produce an average of at least 10 search operations per second.
2. Functionality. The solution should provide a possibility of direct and reverse search by wordforms and lemmas, search by parts of wordforms and lemmas, and phrase search.
3. Compatibility with other software. FLOSS for data storage should work with FLOSS for data processing and vice versa.
4. Completeness of documentation and presence of a large community. Everyone should be able to continue its development, focusing on the documentation and the accumulated experience of the community.
5. Development prospects. The solution should develop and maintain modern technologies.

We analyzed information in public sources [2] and chose FLOSS according to criteria 2–5. The resulting FLOSS set is presented in Table 1.

Table 1. Chosen FLOSS for storing and processing corpus data.

Name	Type and main features
memcached[a]	Storage system of keys and values in memory
memcacheDB[b]	Distributed key-value storage system. It supports transactions and replication
Redis[c]	Non-relational distributed data storage system. It allows storing strings and arrays and making selections from them
FoundationDB[d]	NoSQL database with a shared nothing architecture. It provides an ordered key-value store with transactions
Sphinx[e]	Search system consisting of an indexer and a full-text search module
Elasticsearch[f]	Distributed RESTful search engine. It allows making full-text search for structured data and performing complex search queries
MySQL[g]	Relational DBMS

[a] https://github.com/memcached/memcached
[b] https://github.com/LMDB/memcachedb
[c] https://github.com/antirez/redis
[d] https://github.com/apple/foundationdb
[e] https://github.com/sphinxsearch/sphinx
[f] https://github.com/elastic/elasticsearch
[g] https://github.com/mysql/mysql-server

To verify compliance with the first criterion, we conducted a series of experiments on writing and searching based on the generated data [3]. The best results were shown by the

MySQL + Redis suite, which was chosen by us to store data in the corpus management system.

To build the system based on the selected components, additional elements must be included. So, to bind the PHP interpreter and the Redis data warehouse, we use the PhpRedis extension. In order for the PHP interpreter and MySQL DBMS to work in tandem, we use the php-mysql package, namely the mysqlnd driver, which allows working with the DBMS using its own low-level protocol.

Thus, the scripts executed by the PHP interpreter allow operations with data both from the Redis data storage and from the MySQL DBMS. Performing operations in the order necessary to solve the tasks assigned to the corpus data management system, PHP scripts are the link between indexes and cached data stored in Redis storage, index tables and text data stored in MySQL DBMS.

Let us consider how the processing of a simple direct search query *alma* $_{\text{tat}}$/apple $_{\text{en}}$ is executed. The process execution of such query is shown in Fig. 3.

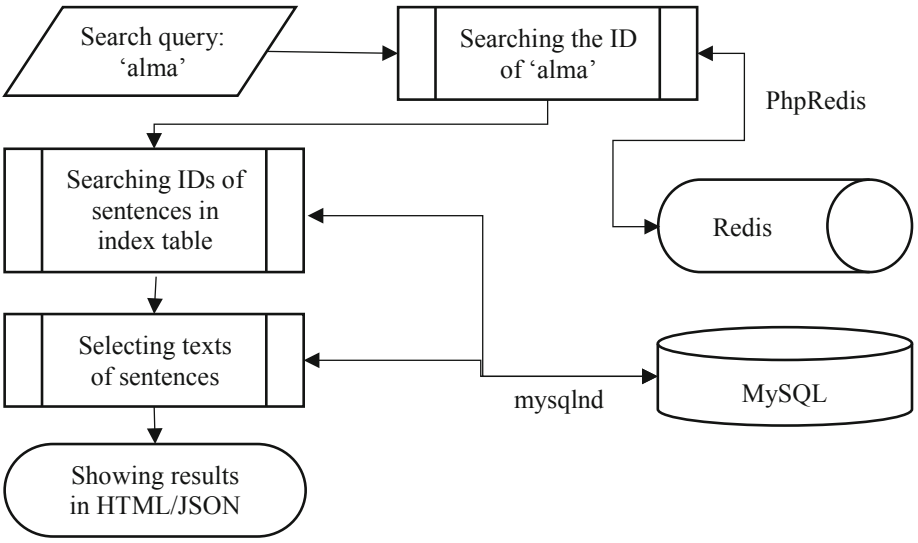

Fig. 3. Process execution of search query *алма*

First of all, the script that performs this task searches the identifier of the word-form *alma* in the Redis database, by making a request using the PhpRedis extension. Already at this point, the script may report an error in the query, if the identifier of the desired wordform is not found in the corpus data.

In the second step, the script makes a request to the MySQL database using mysqlnd. In this case, the script needs to get a list of sentences containing the desired wordform. This data is stored in the index table of corpus data. In response, the script receives a list of context (sentence) identifiers, according to which in the third step the required number of sentence texts is requested. The received data is displayed to an user in the form of an HTML document or a JSON structure.

3 Building a Linked Open Dataset Based on Corpus Data

The corpus has been published on the Linguistic Linked Open Data cloud [4]. The dataset is represented in terms of NIF [5], OntoLex-Lemon [6], LexInfo, OLiA [7] and MMoOn [8] ontologies.

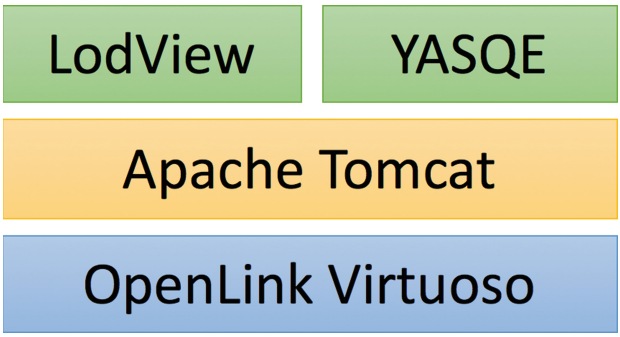

Fig. 4. LLOD publishing platform

LLOD publishing platform consists in the following open-source components (Fig. 4). The RDF is stored in the OpenLink Virtuoso triplestore. The dataset is available via deferrable URI's and SPARQL endpoint. Deferrable URI's are accessible throws the LodView RDF browser, based on the Apache Tomacat webserver. The query interface is powered by YASQE [9].

Publication of the corpus on the LLOD cloud makes possible its interlinking with the external linguistic resources for Tatar, including Russian-Tatar Socio-Political Thesaurus [10], TatWordNet and TatVerbBank.

4 Conclusion

The solutions presented in this article are applied in the developed corpus management system. Using FLOSS significantly reduced the development time and ensured the transparency of processes, flexibility, and possibility of in-depth analysis, as well as opportunities for further development.

The corpus data management system is used to work with Tatar texts. The total volume of the collection of Tatar texts is about 200 million wordforms. The average execution time of the direct search query does not exceed 0.05 s in 98% of cases, and the reverse search is performed by the system within 0.1 s in 82% of cases, which exceeded the expected system performance. In many ways, such performance is provided by FLOSS, used for data storage and processing. Also, thanks to FLOSS, search capabilities were expanded in the system. The search using category lists, the complex search using logical expressions, the search for named entities and other functions were added to the advanced version of the system.

Currently, the corpus management system is in open beta testing and available online at http://tugantel.tatar. After that, we are going to release it under an open license.

Acknowledgements. The work was funded by Russian Science Foundation according to the research project no. 19-71-10056.

References

1. Suleymanov, D., Nevzorova, O., Gatiatullin, A., Gilmullin, R., Khakimov, B.: National corpus of the Tatar language "Tugan Tel": grammatical annotation and implementation. In: Vargas-Sierra, C., (ed.). Selected Papers from the 5th International Conference on Corpus Linguistics, (CILC2013) [Special issue]. Proc. Soc. Behav. Sci. **95**, 68–74 (2013). https://doi.org/10.1016/j.sbspro.2013.10.623
2. Katkar, M., Kutchhii, S., Kutchhii, A.: Performance analysis for NoSQL and SQL. Int. J. Innov. Emerg. Res. Eng. **2**(3), 12–17 (2015)
3. Mukhamedshin, D., Suleymanov, D., Nevzorova, O.: Choosing the right storage solution for the corpus management system (analytical overview and experiments). In: Bouhlel, M.S., Rovetta, S. (eds.) SETIT 2018. SIST, vol. 146, pp. 105–114. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-21005-2_10
4. Cimiano, P., Chiarcos, C., McCrae, J.P., Gracia, J.: Linguistic linked open data cloud. In: Cimiano, P., et al. (eds.) Linguistic Linked Data: Representation, Generation and Applications, pp. 29–41. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-30225-2_3
5. Hellmann, S., Lehmann, J., Auer, S., Brümmer, M.: Integrating NLP using linked data. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 98–113. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_7
6. McCrae, J.P., Bosque-Gil, J., Gracia, J., Buitelaar, P., and Cimiano, P.: The OntoLex-lemon model: development and applications. In: Kosem I., et al. (eds.) Proceedings of the 5th biennial conference on Electronic Lexicography (eLex 2017), pp. 587–597. Lexical Computing CZ (2017)
7. Chiarcos, C.: OLiA – ontologies of linguistic annotation. Seman. Web **6**(4), 379–386 (2015). https://doi.org/10.3233/SW-140167
8. Klimek, B., Arndt, N., Krause, S., Arndt, T.: Creating Linked data morphological language resources with MMoOn - the hebrew morpheme inventory. In: Calzolari N., et al. (eds.) Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), pp. 892–899. ELRA (2016)
9. Rietveld, L., Hoekstra, R.: The YASGUI Family of SPARQL Clients. Seman. Web **8**(3), 373–383 (2017). https://doi.org/10.3233/SW-150197
10. Galieva, A., Kirillovich, A., Khakimov, B., Loukachevitch, N., Nevzorova, O., Suleymanov, D.: Toward domain-specific Russian-Tatar thesaurus construction. In: Proceedings of the International Conference IMS-2017, pp. 120–124. ACM (2017). https://doi.org/10.1145/3143699.3143716