Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 7003–7008
Marseille, 11–16 May 2020

KonText: Advanced and Flexible Corpus Query Interface

Tomáš Machálek
Institute of the Czech National Corpus, Faculty of Arts - Charles University
nám. Jana Palacha 1/2, 116 38 Praha 1, Czech Republic
tomas.machalek@ff.cuni.cz

Abstract

We present an advanced, highly customizable corpus query interface KonText built on top of core libraries of the open-source corpus search engine NoSketch Engine (NoSkE). The aim is to overcome some limitations of the original NoSkE user interface and provide integration capabilities allowing connection of the basic search service with other language resources (LRs). The introduced features are based on long-term feedback given by the users and researchers of the Czech National Corpus (CNC) along with other LRs providers running KonText as a part of their services. KonText is a fully operational and mature software deployed at the CNC since 2014 that currently handles thousands user queries per day.

Keywords: corpus query interface, open-source, web application, distributed application, corpus linguistics

1. General overview and motivation

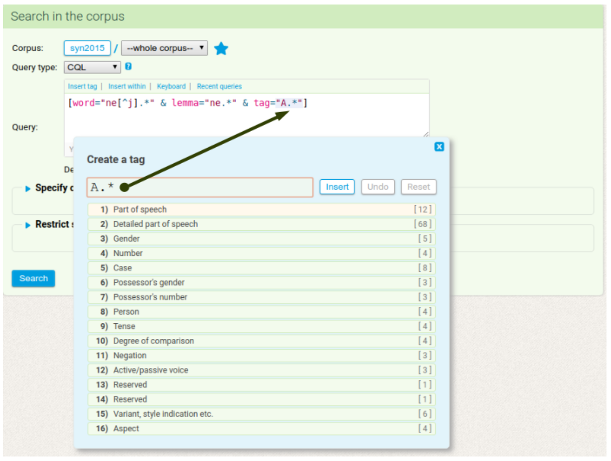
Creating a fully featured corpus search engine with a powerful query language, analytical functions, user-friendly interface, performance scalable to billion tokens corpora and tens or more concurrent users is a tremendous task which takes many years to accomplish. Only a limited number of projects have been able to reach production-ready status and general awareness in linguistic community. Among them, Corpus Workbench (Hardie, 2012), SketchEngine (Kilgarriff et al., 2014), BlackLab (Does et al., 2017) or KorAP (Diewald and Margaretha, 2017) should be mentioned. Looking for a suitable corpus search system, our experience shows that there is simply not a clear winner especially when considering both supported *features* and *scalability* (in terms of corpus size and number of users). A good example of this performance-feature dilemma can be seen on two systems we have experience with - Annis (Krause and Zeldes, 2016) and NoSketch Engine (NoSkE) (Rychlý, 2007). While the former system provides rich, multi-layered annotation capabilities at the cost of limited usability for large corpora (Krause, 2019), the latter system is able to handle billion token corpora reasonably fast but only for simple, single-layer token and span annotations.
Instead of trying to build another search engine fully compliant with our needs, we took a pragmatic approach by choosing a simple, scalable search engine and building a new interface and auxiliary services integrating the search engine with other data resources and providing many features missing in the original software.
KonText builds on our long-term experience with NoSkE which is an open-source version of a commercial corpus search software Sketch Engine (SkE) by Lexical Computing Ltd. During our initial attempts to enhance the original user interface (Machálek and Křen, 2013), we were faced by increasing number of requested features. As we also had a different approach to the application architecture, we have gradually rewritten the original interface under the new name of KonText. The only remaining parts now are core data retrieval and indexing libraries of NoSkE including its core library Manatee-open, which also means that KonText supports the same query language as NoSkE and SkE - the CQL [1].
Another strong motivation to develop KonText came from the need for a truly open-source project, in contrast with NoSkE which is controlled solely by its authors and tied to the more feature-rich and commercial SkE. This means that KonText is able not only to keep pace with the current state-of-the-art in corpus indexing and searching software, but also to integrate the code from other developers and to reflect user feedback in short update cycles.
KonText is endowed with the following key features:

- support for *spoken* (audio playback, visual representation of dialogues) and *parallel* corpora,
- rendering of dependency *syntax trees*,
- advanced creation of *subcorpora*
  - based on *user-defined ratios* of different text types,
  - based on the corpora *alignment*,
- helper tools for *query creation* (Figures 1 and 2) and exploring of corpora structure,
- *integration with other services* that can provide additional information about searched terms.

2. End user features

In addition to the features inherited from NoSkE (concordance sorting and filtering, frequency distribution, collocation analysis), KonText provides a rich set of new, original features, some of which (e.g. 2.2.2., 2.2.3., 2.3.3.) can be rarely seen in other corpus search interfaces. But even with the more common features on mind, it is their presence and

[1]CQL is a variant of the query language developed at the Corpora and Lexicons group, IMS, University of Stuttgart. See `https://www.sketchengine.eu/documentation/corpus-querying/` for more information.



```
[word="ne[^j].*" & lemma="ne.*" & tag="A.*"]
```

```
A.*
```

1) Part of speech	[12]
2) Detailed part of speech	[68]
3) Gender	[5]
4) Number	[4]
5) Case	[8]
6) Possessor's gender	[3]
7) Possessor's number	[3]
8) Person	[4]
9) Tense	[4]
10) Degree of comparison	[4]
11) Negation	[3]
12) Active/passive voice	[3]
13) Reserved	[1]
14) Reserved	[1]
15) Variant, style indication etc.	[6]
16) Aspect	[4]

Figure 1: CQL query editor with syntax highlighting and interactive widget for entering PoS tags

mutual complementation within a single application which gives KonText its value.
In this section, we will discuss in more detail features that have been implemented beyond those inherited from NoSkE. They can be divided into the following categories:

- query construction,
- data selection,
- result presentation and manipulation.

2.1. Query Construction

Modern corpora are accompanied by rich metadata sets and Manatee-open supports a powerful query language CQL. This poses a challenge not only for users who want to learn CQL, especially when their background is in the humanities, but also for experienced users. For this reason, KonText helps users in CQL query construction by introducing extensions of the basic query form that allow for interactive selection of parameters as well as for easy re-use of the individual items stored in a query history database.

2.1.1. Query syntax highlighting

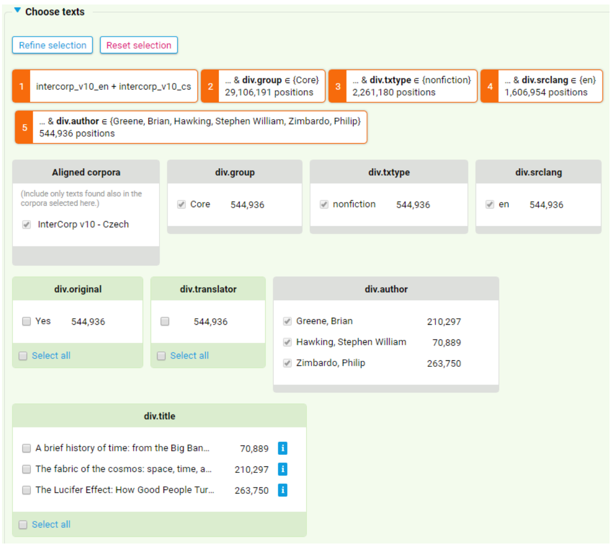
KonText includes its own custom CQL parser which provides syntax highlighting and basic validation of corpus-specific values of both positional and structural attributes (Figure 2).

2.1.2. Tag builder

As a part of the CQL editor, KonText offers an interactive tag builder widget for user-friendly selection of individual categories within a given tagset. The tag builder can be configured to support any *positional* and *attribute-value* tagset, including Universal Dependencies (UD).

2.1.3. Query history and persistence

Query history is stored by KonText on the server on a per-user basis that makes it available regardless of the device a user may use. The query history includes also additional information, e.g. date and time, queried corpus, query type and parameters. Furthermore, KonText includes an overview screen for easy searching and filtering the user



1 intercorp_v10_en + intercorp_v10_cs
2 ... & **div.group** ∈ {Core} 29,106,191 positions
3 ... & **div.txtype** ∈ {nonfiction} 2,261,180 positions
4 ... & **div.srclang** ∈ {en} 1,606,954 positions
5 ... & **div.author** ∈ {Greene, Brian, Hawking, Stephen William, Zimbardo, Philip} 544,936 positions

Aligned corpora: (Include only texts found also in the corpora selected here.) ✓ InterCorp v10 - Czech

div.group: ✓ Core 544,936

div.txtype: ✓ nonfiction 544,936

div.srclang: ✓ en 544,936

div.original: ☐ Yes 544,936

div.translator: ☐ 544,936

div.author:

✓ Greene, Brian	210,297
✓ Hawking, Stephen William	70,889
✓ Zimbardo, Philip	263,750

div.title:

☐ A brief history of time: from the Big Ban...	70,889
☐ The fabric of the cosmos: space, time, a...	210,297
☐ The Lucifer Effect: How Good People Tur...	263,750

Figure 2: A widget for interactive text type selection by gradual application of data filtering

query history. There is also an option to archive individual items under a custom name for later reuse. To support reproducibility, the archiving of a query includes also selected subcorpora and all other options.

2.2. Data Selection

We believe that corpus query engines should provide users with the possibility to easily examine the actual contents of a corpus, as well as to define subcorpora based on various user-selected criteria.

2.2.1. Finding a corpus based on different criteria

Out of the box, KonText provides two principally different modules for organizing and searching for corpora. One is based on a tree hierarchy, the other uses multiple keywords (tags) attachment to individual corpora for more flexible categorization. It is also possible to use different data backends for the modules. The core KonText installation offers either a variant based on an XML file (best suited for smaller installations with only a few corpora) or a backend operating on top of a relational database.

2.2.2. Interactive text selection

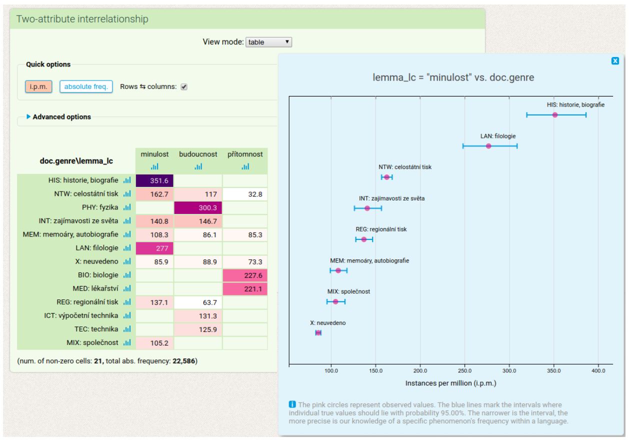
A module for interactive text selection (based on a combination of criteria) facilitates user creation of tailor-made subcorpora by ”zooming into” the selected parts of a corpus (Figure 2). The selection can be made also on the level of individual documents. For parallel corpora, there is a possibility to define also alignment-based subcorpora.

2.2.3. Proportion-based text selection

KonText also supports automatic text selection based on custom proportions of individual attributes (typically text registers). As a result, the created subcorpus can keep the user-specified proportion of the main registers without the need to specify individual texts.

2.2.4. Public subcorpora

All the subcorpora created by an individual user (using any of the methods above) can be named and shared with other


Two-attribute interrelationship

View mode: table

Quick options

i.p.m. | absolute freq. | Rows ⇆ columns: ✓

Advanced options

doc.genre\lemma_lc	minulost	budoucnost	přítomnost
HIS: historie, biografie	351.6		
NTW: celostátní tisk	162.7	117	32.8
PHY: fyzika		300.3	
INT: zajímavosti ze světa	140.8	146.7	
MEM: memoáry, autobiografie	108.3	86.1	85.3
LAN: filologie	277		
X: neuvedeno	85.9	88.9	73.3
BIO: biologie			227.6
MED: lékařství			221.1
REG: regionální tisk	137.1	63.7	
ICT: výpočetní technika		131.3	
TEC: technika		125.9	
MIX: společnost	105.2		

(num. of non-zero cells: **21**, total abs. frequency: **22,586**)

lemma_lc = "minulost" vs. doc.genre

The pink circles represent observed values. The blue lines mark the intervals where individual true values should lie with probability 95.00%. The narrower is the interval, the more precise is our knowledge of a specific phenomenon's frequency within a language.

Figure 3: 2-dimensional frequency distribution

users, which helps cooperation and research reproducibility.

2.3. Result presentation and manipulation

As for the concordances and results of analytic functions, KonText builds above the output provided by the core Manatee-open library and extends it by additional data or rearranges it for better readability and interpretability. As examples of such additional data we can mention external dictionary entries for individual concordance tokens or syntax information for sentences loaded from an external database (more on this can be found in 3.1.).

2.3.1. Reproducibility and control

After a query is executed, there are numerous operations that can be performed on a resulting concordance: filtering, sorting, random sampling, frequency distribution, collocation analysis etc. As these operations can be chained, the KonText breadcrumb navigation keeps track of all the consecutive steps, so that it is easy to control them. Furthermore, *the individual processing steps are editable*, which means it is possible only to change parameters of a single operation and then to re-run the whole operation chain while keeping the other operations intact.
In addition, each operation sequence gets its unique ID which is stored in the URL, so that sharing the URL (in a paper, with a colleague etc.) leads to reproducible results.

2.3.2. Adequate frequency characteristics

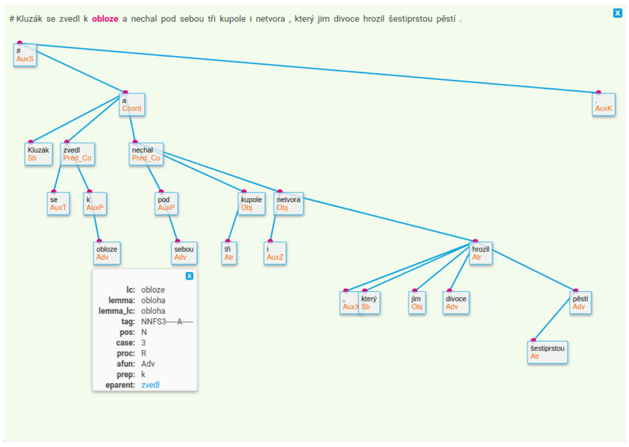
KonText supports more advanced frequency characteristics next to regular frequency: i.p.m. & dispersion-based ARF (Savický and Hlaváčová, 2002), including their use only in well-defined situations.

2.3.3. 2-dimensional frequency distribution

As shown in Figure 3, KonText allows for analysis of frequency interrelationship between two attributes (both positional and structural ones).

2.3.4. Manual categorization of concordance lines

It is possible to manually select concordance lines and attach custom numerical labels to them. This is useful for any manual categorization, e.g. for classification of the individual word senses of a searched term. Such a selection can be


Kluzák se zvedl k **obloze** a nechal pod sebou tři kupole i netvora , který jim divoce hrozil šestiprstou pěstí .

lc:	obloze
lemma:	obloha
lemma_lc:	obloha
tag:	NNFS3-----A----
pos:	N
case:	3
proc:	R
afun:	Adv
prep:	k
eparent:	zvedl

Figure 4: A dependency syntax tree example

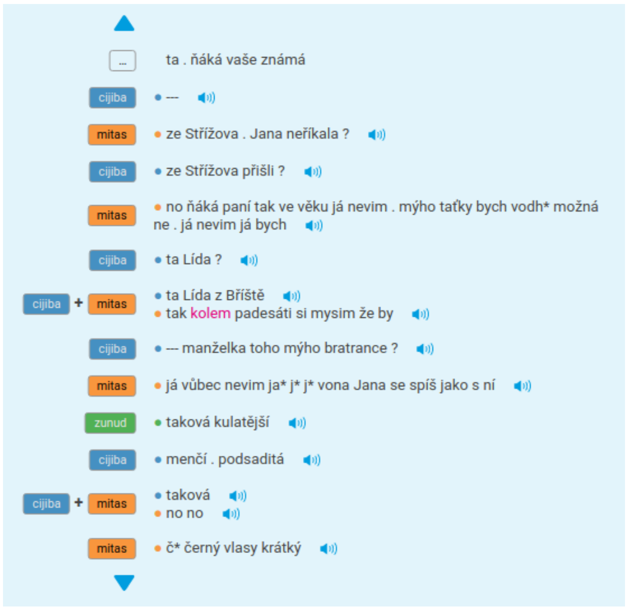

...	ta . ňáká vaše známá
cijiba	• ---
mitas	• ze Střížova . Jana neříkala ?
cijiba	• ze Střížova přišli ?
mitas	• no ňáká paní tak ve věku já nevim . mýho taťky bych vodh* možná ne . já nevim já bych
cijiba	• ta Lída ?
cijiba + mitas	• ta Lída z Bříště • tak kolem padesáti si mysim že by
cijiba	• --- manželka toho mýho bratrance ?
mitas	• já vůbec nevim ja* j* j* vona Jana se spíš jako s ní
zunud	• taková kulatější
cijiba	• menčí . podsaditá
cijiba + mitas	• taková • no no
mitas	• č* černý vlasy krátký

Figure 5: A dialog visualization example

further filtered, exported or passed to other users via a URL address.

2.3.5. Rendering of dependency syntax trees

KonText is currently able to render PDT-like dependency syntax trees (Figure 4), support for UD (Nivre et al., 2016) is in progress. This functionality is managed by plug-ins (see 3.1.), so that other research groups can make KonText support syntax in their own formalism.

2.3.6. Support for spoken corpora

In spite of the underlying Manatee-open data model that is scalable, but rather simple, KonText features a support for querying spoken conversational corpora:

- text regions can be accompanied by audio chunks for direct playback,
- concordance detail views can be rendered as dialogues with a clear indication of speaker turns and overlaps (Figure 5).

3. Provider-oriented features

KonText focuses not only on user features but also on issues regarding its adaptability by providers besides CNC and on performance improvements when facing large number of users requesting computationally demanding tasks.

3.1. System Integrability and extensibility

One of the key goals right from the start was to prevent KonText from being locked to the environment and specific CNC infrastructure. Seeing other individuals and institutions facing similar problems to ours, we believed that KonText could provide a solution for them too, but only as long as we were able to develop it in a way general and flexible enough to avoid a need for rewriting the same code each time someone needs a specific feature or customization. This is addressed via the concept of *plug-ins*.
The goal of the plug-in architecture is not just to extend the functionality of KonText in different ways but also to make core parts of the application replaceable with alternative implementations. This allows other providers to integrate KonText into their existing information systems without the need to interfere with the KonText core source code. For example it is very common that each organization has its own authentication method along with a custom user database or that the organization uses one of the widely adopted authentication methods/services (e.g. Shibboleth in academia). For KonText, this means just a custom implementation of its *auth* plug-in.
The plug-ins are developed in *Python* on the server-side and *JavaScript/TypeScript* on the client-side (if needed).

Listing 1: An example of a plug-in set-up in the main configuration file. Here we specify a plug-in used to connect individual tokens to external services.

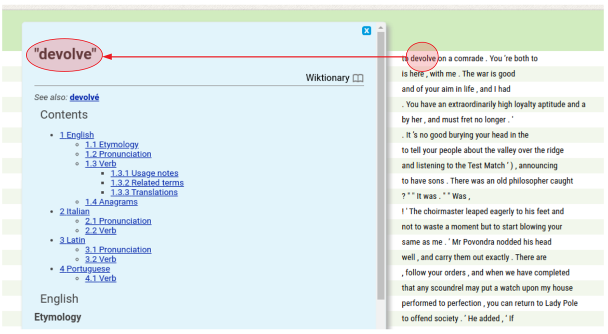
```
<plugins>
   ...
  <token_connect>
    <module>default_token_connect </module>
    <js_module>defaultTokenConnect </js_module>
    <providers_conf  extension-by="default">
       /opt/kontext/token-detail-providers.json
     </providers_conf>
    <cache_db_path  extension-by="default">
       /var/opt/kontext/td_cache.db
     </cache_db_path>
    <cache_ttl_days  extension-by="default">
       7
     </cache_ttl_days>
    <cache_rows_limit  extension-by="default">
       100000
     </cache_rows_limit>
   </token_connect>
   ...
</plugins>
```

In KonText, plug-ins can be divided into two groups:

- *required plug-ins* which are necessary for KonText to run (e.g. plug-ins for authentication and database access),
- *optional plug-ins* which extend core functionality in different ways (e.g. by adding an interactive tag builder widget or a module for syntax dependency tree rendering).

For plug-in developers, KonText defines a set of abstract classes prescribing behavior a custom plug-in implementation must implement. KonText configuration file then specifies which concrete implementations are used for all the enabled/required functions (Listing 1). The required plug-ins used by KonText include:

- *db* plugin for unified access to a key-value storage used across the application (Figure 8); currently available implementations are for Redis and SQLite3 databases,
- *sessions* for handling HTTP user sessions,
- *auth* for custom authentication methods (including remote authentication),
- *conc_cache* for storing calculated concordance results for later reuse.


"devolve"

Wiktionary

See also: devolvé

Contents

- 1 English
  - 1.1 Etymology
  - 1.2 Pronunciation
  - 1.3 Verb
    - 1.3.1 Usage notes
    - 1.3.2 Related terms
    - 1.3.3 Translations
  - 1.4 Anagrams
- 2 Italian
  - 2.1 Pronunciation
  - 2.2 Verb
- 3 Latin
  - 3.1 Pronunciation
  - 3.2 Verb
- 4 Portuguese
  - 4.1 Verb

English

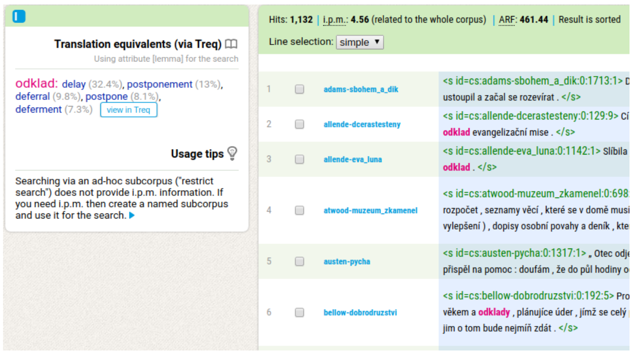
Etymology

```
to devolve on a comrade . You 're both to
is here , with me . The war is good
and of your aim in life , and I had
. You have an extraordinarily high loyalty aptitude and a
by her , and must fret no longer . '
. It 's no good burying your head in the
to tell your people about the valley over the ridge
and listening to the Test Match ' ) , announcing
to have sons . There was an old philosopher caught
? " " It was . " " Was ,
! ' The choirmaster leaped eagerly to his feet and
not to waste a moment but to start blowing your
same as me . ' Mr Povondra nodded his head
well , and carry them out exactly . There are
, follow your orders , and when we have completed
that any scoundrel may put a watch upon my house
performed to perfection , you can return to Lady Pole
to offend society . ' He added , ' If
```

Figure 6: A data integration example - Wiktionary.org

3.2. Integrability with other LRs

One of the ways how to extend the functionality provided by KonText lies in integrating it with other LRs. Most progress has been reached in connecting individual tokens or aggregated results from concordances. The results are now available as *token_connect* and *kwic_connect* plug-ins (Figures 6 and 7).
The *token_connect* plug-in allows connecting any position in the text (along with available meta-data) with an external service. Such a solution allows e.g. an integration with dictionaries or providing an alternative view on a respective part of the concordance (e.g. a different annotation/tokenization level).
The *kwic_connect* plug-in obtains the most frequent KWIC matches from a concordance and searches for each of them individually using an external service.
Among examples of LR integration, we can mention the CNC translation service Treq (Škrabal and Vavřín, 2017) (Figure 7), integration with Vallex (Urešová et al., 2014) provided by LINDAT/CLARIAH-CZ and also an integration of KonText and TEITOK (Janssen, 2015) which is currently in development by the author of TEITOK.
We have been also experimenting with using whole concordances as sources of data processed by a 3rd party collocation analysis library (Belica, 1995), yet this will require further analysis and testing as the performance penalty is much higher due to the need to encode and transfer large amounts of data proportional to a respective concordance size (which can be many gigabytes in size).



Translation equivalents (via Treq)
Using attribute [lemma] for the search

odklad: delay (32.4%), postponement (13%), deferral (9.8%), postpone (8.1%), deferment (7.3%) view in Treq

Usage tips

Searching via an ad-hoc subcorpus ("restrict search") does not provide i.p.m. information. If you need i.p.m. then create a named subcorpus and use it for the search.

Hits: **1,132** | i.p.m.: **4.56** (related to the whole corpus) | ARF: **461.44** | Result is sorted

Line selection: simple

1	☐	adams-sbohem_a_dik	<s id=cs:adams-sbohem_a_dik:0:1713:1> D ustoupil a začal se rozevírat . </s>
2	☐	allende-dcerastesteny	<s id=cs:allende-dcerastesteny:0:129:9> Cí **odklad** evangelizační mise . </s>
3	☐	allende-eva_luna	<s id=cs:allende-eva_luna:0:1142:1> Slíbila **odklad** . </s>
4	☐	atwood-muzeum_zkamenel	<s id=cs:atwood-muzeum_zkamenel:0:698 rozpočet , seznamy věcí , které se v domě musi vylepšení) , dopisy osobní povahy a deník , kte
5	☐	austen-pycha	<s id=cs:austen-pycha:0:1317:1> „ Otec odj přispěl na pomoc : doufám , že do půl hodiny o
6	☐	bellow-dobrodruzstvi	<s id=cs:bellow-dobrodruzstvi:0:192:5> Pro věkem a **odklady** , plánujíce úder , jímž se celý jim o tom bude nejmíň zdát . </s>

Figure 7: A data integration example - translation database Treq

3.3. Performance optimizations

As for the performance optimizations, the possibilities are rather limited as most of the search and data aggregation performance is determined by the Manatee-open search engine. However even in this case we have been able to reach some progress by involving a distributed asynchronous task queue for computationally demanding jobs.

Even though NoSkE supports asynchronous calculation of concordances, it is only able to run the calculation on the same computer via spawning of operating system processes and only for certain types of queries. In contrast, KonText *delegates such computations to one or more worker servers* allowing horizontal scaling of the application performance by engaging more physical computers. The *web* and *worker* server *decoupling* has its limits in a way that the servers must share a file system allowing the web server to access calculated results. But such a requirement can be met via standard GNU/Linux server set-up (e.g. by involving a network or a distributed file systems).

The aforementioned optimization is suitable to tackle increasing number of concurrent users but it cannot be easily used to distribute/optimize a single concordance search as this would require a low-level support on the search engine side to get a reasonable operational efficiency. In terms of single query parallelization in (No)SkE, there was a promissing progress presented in (Rábara and Rychlý, 2015).

4. Architecture

KonText is a client-server web application developed to run on a regular GNU/Linux server (tested on Ubuntu and CentOS) along with a few common dependencies (HTTP proxy server, Python runtime, Node.JS). The essential components of a running KonText installation are:

- *Manatee-open* library as a search and analytics engine,
- *WSGI application* written in Python (a part of KonText),
- dynamic HTML user interface generated by a client-side application written in TypeScript language (a part of KonText),
- *key-value database* Redis for storing auxiliary data,

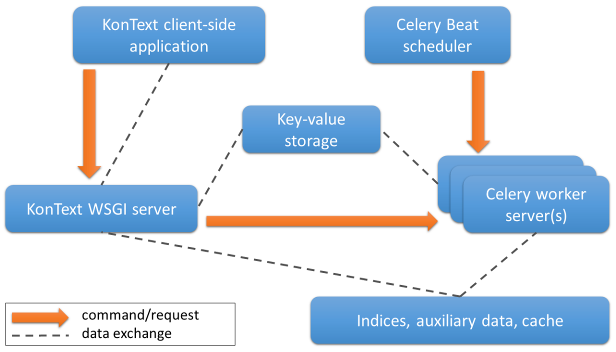


Figure 8: a communication and data exchange structure within KonText

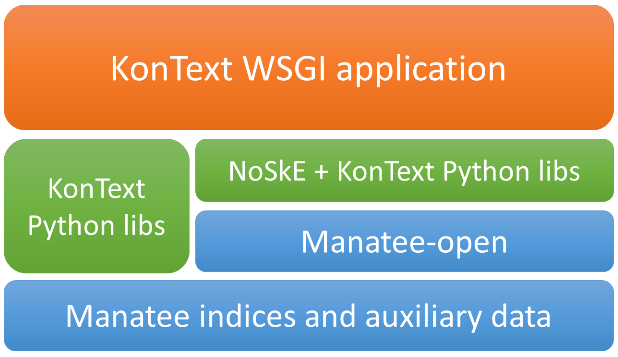

KonText WSGI application
KonText Python libs
NoSkE + KonText Python libs
Manatee-open
Manatee indices and auxiliary data

Figure 9: A layer view of the server-side application

- *asynchronous task queue* server Celery,
- *task scheduler* Celery Beat for regular automatic maintenance.

A raw scheme depicting how these components pass commands and exchange data can be seen in Figure 8.

4.1. Server-side part

Server-side code comprises of two parts - a web server and a worker server for asynchronous tasks.

4.1.1. WSGI web server

WSGI [2] application is implemented on top of the *Werkzeug WSGI web application library*. Rather than a strict framework, Werkzeug provides a set of libraries for flexible web application development. Its design allowed for gradual replacement of the original NoSkE server-side components with Werkzeug-based ones. A layered view of KonText server-side application structure is depicted in Figure 9.

The WSGI application can be run in any compatible environment, though we can recommend *Gunicorn* or *uWSGI* HTTP servers for production installations.

4.1.2. Asynchronous task queue

Some KonText operations (e.g. creating large concordances, creating subcorpora with respect to a translation alignment or with custom text type proportions) are computationally/data intensive with typical waiting times from

[2]Web Server Gateway Interface - a widely used interface for forwarding HTTP requests to Python applications

tens of seconds to minutes. To avoid blocking user interface for the time of the calculation, KonText processes such actions in an asynchronous way. When a user initiates an action, KonText puts the action to a task queue allowing the user to interact with the interface in a normal way. Once the task is finished, the user is notified that the result is available. To implement the task queue, KonText uses a Python application Celery which provides a complete environment for asynchronous distributed task processing.

4.2. Client-side part

Client-side part of KonText is run in a web browser as an interactive web application. KonText relies on modern, yet still proven libraries based on the state-of-the-art paradigms for developing user interfaces:

- *React* user interface framework,
- custom state management library inspired by *Flux* pattern and *Redux* library built on top of reactive programming library *RxJS*,
- immutable data structures,
- *TypeScript* language.

5. Summary and Outlook

KonText has been actively developed since 2013 with more than 6700 commits in its Git repository and hundreds of resolved issues and topics discussed on CNC's public support website.

A long-term cooperation on development of KonText exists with the Institute of Formal and Applied Linguistics (Faculty of Mathematics and Physics at Charles University) and KonText has been adopted as the query interface at the LINDAT/CLARIAH-CZ repository.

KonText is deployed or actively tested by CLARIN centres in the Czech Republic, Poland and Slovenia, as well as by other institutions (mova.institute, Serbski institut).

The development takes place on GitHub (`https://github.com/czcorpus/kontext`) where developers and users are welcome to contribute in different ways – fixing/improving code, reporting bugs or discussing new features. The production version is available at `https://kontext.korpus.cz`.

As for the features planned for the near future we have put focus on the problem of recognizing statistical significance of results, improved support for UD tokenized and tagged texts and also on improved support for presenting of time-based data.

6. Acknowledgements

This paper resulted from the implementation of the Czech National Corpus project (LM2018137) funded by the Ministry of Education, Youth and Sports of the Czech Republic within the framework of Large Research, Development and Innovation Infrastructures.

The author would also like to thank Michal Křen for valuable comments on this article.

7. Bibliographical References

Belica, C. (1995). Collocation analysis and clustering. *Corpus Analysis Module*.

Diewald, N. and Margaretha, E. (2017). Krill: Korap search and analysis engine. *Journal for Language Technology and Computational Linguistics (JLCL), 31 (1)*, pages 63–80.

Does, J. D., Niestadt, J., and Depuydt, K. (2017). Creating research environments with blacklab. In J. Odijk, editor, *CLARIN in the Low Countries*, pages 245–257. Ubiquity Press, Utrecht.

Hardie, A. (2012). Cqpweb - combining power, flexibility and usability in a corpus analysis tool. *International Journal of Corpus Linguistics*, 17(3):380–409.

Janssen, M. (2015). Multi-level manuscript transcription: Teitok. *Congresso de Humanidades Digitais em Portugal, Lisboa*.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlý, P., and Suchomel, V. (2014). The sketch engine: Ten years on. *Lexicography*.

Krause, T. and Zeldes, A. (2016). Annis3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*.

Krause, T. (2019). *ANNIS: A graph-based query system for deeply annotated text corpora*. Ph.D. thesis.

Machálek, T. and Křen, M. (2013). Query interface for diverse corpus types. *Natural language processing, corpus linguistics, e-learning*.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Rychlý, P. (2007). Manatee/bonito - a modular corpus manager. *1st Workshop on Recent Advances in Slavonic Natural Language Processing.*, pages 65–70.

Rábara, J. and Rychlý, P. (2015). Concurrent processing of text corpus queries. *Ninth Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 49–58.

Savický, P. and Hlaváčová, J. (2002). Measures of word commonness. *Journal of Quantitative Linguistics*, 9(3):215–231.

Urešová, Z., Štěpánek, J., Hajič, J., Panevová, J., and Mikulová, M. (2014). Pdt-vallex: Czech valency lexicon linked to treebanks. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Škrabal, M. and Vavřín, M. (2017). The translation equivalents database (treq) as a lexicographer's aid. In Iztok Kosem, editor, *Electronic lexicography in the 21st century. Proceedings of eLex 2017 conference*, pages 124–137. Lexical Computing.