Tēzaurs.lv – the experience of building a multifunctional lexical resource

Mikus Grasmanis, Pēteris Paikens, Lauma Pretkalniņa, Laura Rituma, Laine Strankale, Artūrs Znotiņš, and Normunds Grūzītis

Institute of Mathematics and Computer Science, University of Latvia, 29 Raina Blvd., Riga, LV-1459, Latvia
E-mail: peteris@ailab.lv

Abstract

In this paper, we describe our findings from developing the lexicographic platform Tēzaurs.lv, extending it from a traditional explanatory dictionary into a multifunctional resource for structured lexical data. Tēzaurs.lv is the largest Latvian dictionary with more than 390,000 entries, which emerged as a compilation from nearly 300 prior dictionaries and other sources. Recently, it has been extended with Latvian WordNet data, effectively making it also a synonym dictionary and a translation dictionary. Each entry can contain multiple lexemes with their grammatical information and inflection tables, enabling search on inflection forms and spelling variants.

For the new requirements, we have developed a lexical database system and a collaborative online editor toolkit, which are also used for two other major Latvian dictionaries. While previously the data model and tools were based on what the end user would see in a dictionary entry, the current infrastructure is designed with a highly structured lexical data model. This avoids duplication and helps to ensure consistency if entries or word senses are edited or merged, and it supports the usage of this data in computational linguistics.

Keywords: lexicography; platform; data model; Latvian

Introduction

Tēzaurs.lv is the largest Latvian electronic dictionary with more than 390,000 entries, which was initiated as a consolidated compilation from approximately 300 dictionaries (Spektors et al., 2016) and other sources, and has recently been extended and developed with the addition of Latvian WordNet (Paikens et al., 2022) data (6,610 synonym sets) and 75,400 manually curated corpus examples for specific senses. All entries contain at least one lexeme and one sense defined. 118,000 lexemes contain appropriate inflectional paradigms to provide inflectional tables and the ability to search by inflectional forms.

Tēzaurs.lv emerged around 2009 as a side result of a larger research project in computational linguistics. The dictionary was encoded in an ad-hoc text format which annotated the beginning of each element with a two-letter code. As the dictionary grew in size, a set of consistency verification scripts was developed (Danovskis, 2014), but there was no multi-user editing support, and everything had to be done by a single editor in the single authoritative copy of the dictionary. New releases were published four times per year by pre-processing the dictionary data from the in-house format to static HTML and loading the pre-rendered entries into a simple database with a thin front-end application on top. It provided basic search and display functionality, however, it lacked in-depth search

functionality, e.g., it was possible to search only by the headword of the entry, and not by derivatives, multi-word expressions (MWEs), glosses, etc.

Until 2020, Tēzaurs.lv had a flat structure, resembling a printed dictionary with lots of duplicated information (for example, each MWE was usually described multiple times – in each constituent word entry – but with potentially different definitions), which also turned out to be a major obstacle for further enhancement of the dictionary.

Since the last report on Tēzaurs.lv (Paikens et al., 2019), it has seen a significant shift in its focus and features, transforming from a traditional explanatory dictionary towards a "3-in-1" lexical resource that augments the senses and their explanations with WordNet style (Fellbaum, 1998) links, effectively making it also a synonym dictionary and a translation dictionary where translation equivalents are aligned at the sense level. Each entry can contain multiple lexemes, including spelling variants and derivations, and also inflectional and grammatical information for them. Senses are organised in two levels – top level senses and subsenses, and each can have corpora examples attached. Both lexemes and senses can have additional data about language style, usage, domain, etc. Entries can also contain unstructured information about etymology, and normative commentary.

The dictionary editing tools and the whole infrastructure have also undergone major changes over the course of Tēzaurs.lv development.

The goal for this undertaking was a web-based multi-user multi-dictionary application with a centralized database as a single source of truth, which supports dictionary creation and editing, as well as dictionary publishing. For these needs, we have developed a lexical database system and an editor toolkit which, besides the Tēzaurs.lv dictionary itself, is used also for two other major Latvian dictionaries: Dictionary of Standard Latvian (LLVV, retro-digitised)[1] and Dictionary of Contemporary Latvian (MLVV, continuously updated)[2]. We also considered using TLex[3] or Lexonomy[4] (Rambousek et al., 2021), but we were worried about that the large amount and rather complex structure of already existing Tēzaurs.lv data might make these solutions slow and hard to maintain. From newer development it is worth mentioning Lexmart[5] (Simões et al., 2019), however it was not available yet when work on Tēzaurs.lv platform started, and it works on top of an XML database, which does not fit our plans for using a fine-granular data model.

In Section 1, we describe the Tēzaurs.lv online platform and the features relevant for its end-users. Section 2 describes the data model used for the multifaceted lexicographic data, and Section 3 describes the tools supporting the lexicographic workflow.

1. Tēzaurs.lv online platform

The Tēzaurs.lv lexicographic platform is developed as a web application which supports collaborative dictionary editing as well as dictionary publishing.

In the editor mode, the application works directly on the atomic data stored in the database. Data consistency is ensured via backend validations, database constraints and

[1] https://llvv.tezaurs.lv
[2] https://mlvv.tezaurs.lv
[3] https://tshwanedje.com/tshwanelex/
[4] https://www.lexonomy.eu/
[5] http://lexmart.eu/

transactions. In the publishing mode, it works in read-only mode on pre-generated data (complete entries for fast response time and reverse indices for search support), which are created in the quarterly release preparation process.

From the data point of view, the published version is an enriched read-only copy (snapshot) of the dictionary database state in the moment of publishing. From the application point of view, the published version utilizes a subset of the same data procedures and view templates as the editor's view, thus ensuring consistency between both views.

1.1 User Interface

1.1.1 Entry View

The central element of the interface is the view of an Entry (see Figure 1). It consists of the Heading, one or more Lexeme blocks and one or more Sense blocks, and ends with a list of the lexical sources for this entry. Lexeme blocks may have inflection information, Sense blocks may have several sub-blocks: usage examples, related senses, translations, MWEs. To make the presentation mode compact, the inflection tables in lexeme blocks and all sub-blocks in the sense blocks are expandable but initially collapsed. All blocks may have verbalization of grammatical and usage information.

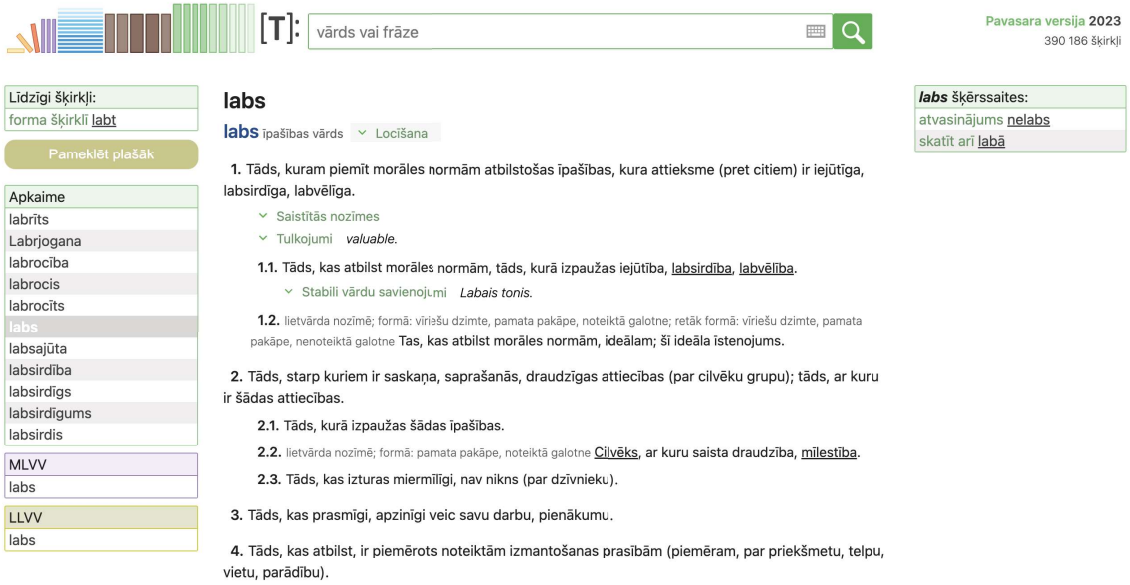
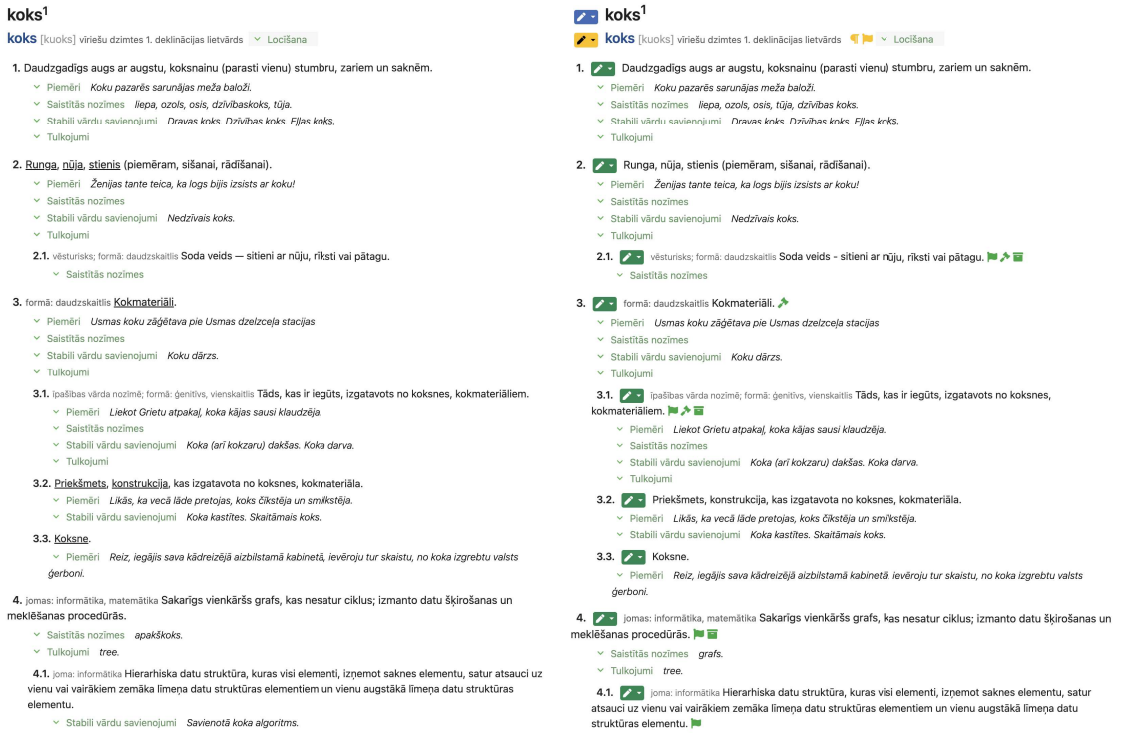


Figure 1: Public view of a Tēzaurs.lv entry. The upper panel contains a search bar, the left side panel contains a box of related entries, a box of neighbours, and the boxes of results of the search in two other dictionaries. The right side panel contains a box of other links to related entries, an entry text in the middle.

At the top of the entry area is the header with the search box. On both sides of the entry area there are side bars with navigational items. The left side bar is devoted to the neighbourhood navigation, and the right side bar to the larger distance navigation.

The application user interface is constructed from a set of templates, which are rendered on the appropriate data fetched from the database.



(a) Public view of an entry *koks:1*.

(b) Editor's view of the same entry *koks:1*.

Figure 2: Public and editor's view of the same entry, with all blocks collapsed.

The editor's view uses the same templates as the reader's view to ensure WYSIWYG[6], augmenting it with some additional icons for extra information and for initiating editing actions, thus ensuring that editors see as close as possible the look of the final entry (see Figure 2).

Where possible, editor tools allow to choose attribute values from pre-filled drop-down lists to ensure data consistency, as shown in Figure 3. The platform also provides means to link to multiple external data sources. Currently it is possible to add usage evidence from the Latvian National Corpora Collection (Saulite et al., 2022) and links to the Princeton WordNet (Paikens et al., 2023).

To ease the adding of corpora links, the editor's view can be switched to a view where in addition to the meanings also any examples found in corpora are also visible, as shown in Figure 4. A separate editing window has been created for marking Wordnet links, where you can create links to both Latvian language wordnet synsets and Princeton wordnet synsets (Figure 5).

The platform hosts multiple (currently three) dictionaries with slightly different entry standards and requirements. The differences are mostly covered via conditional fragments

[6] What You See Is What You Get

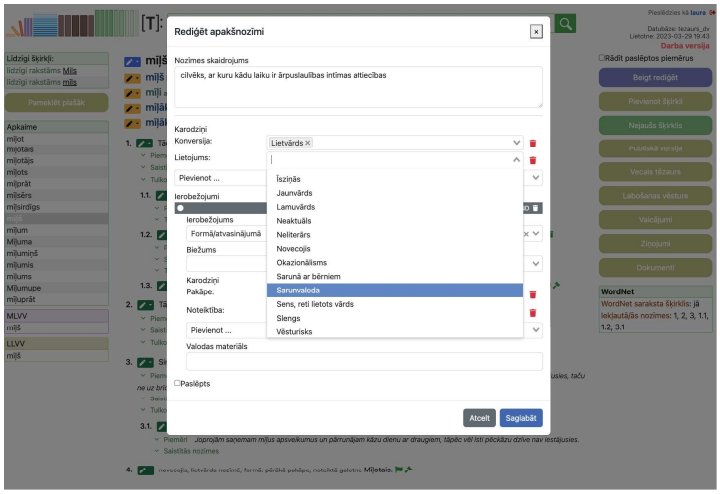


Figure 3: Editing dialog for a Sense, with a flag value selection dropdown opened.

in the templates or conditionally selected sub-templates. More dictionaries could be easily added.

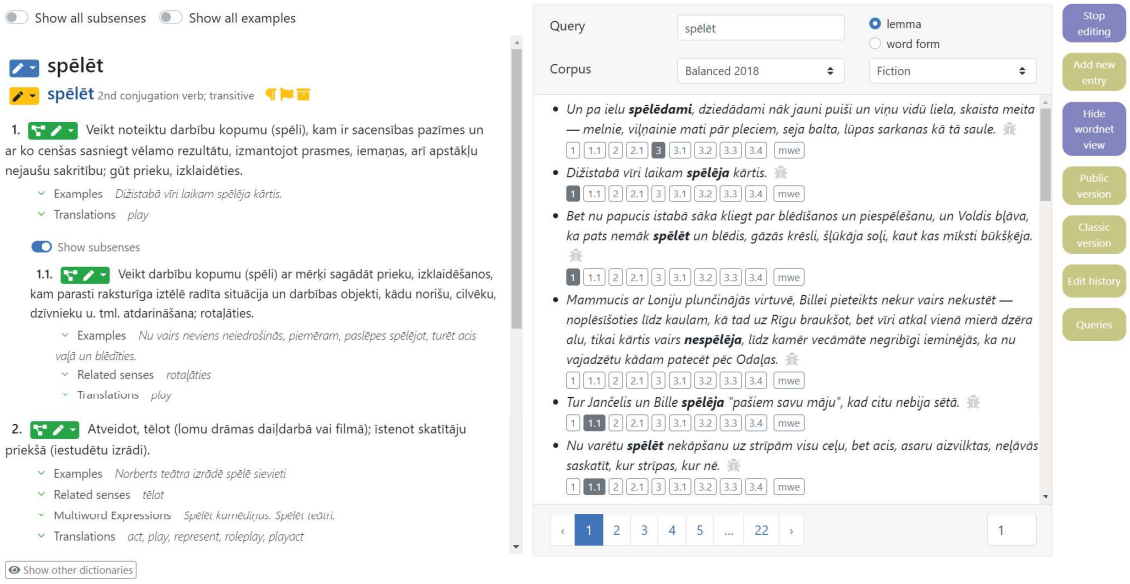


Figure 4: Editor's view of a Tēzaurs.lv entry in mode which allows adding examples. The upper left side shows the list of word senses. The newly added example browser is on the right.

1.1.2 Search

The goal is to provide a simple and unified interface for different types of users through a single input field that can handle all their needs, much like a search bar on a modern web browser.

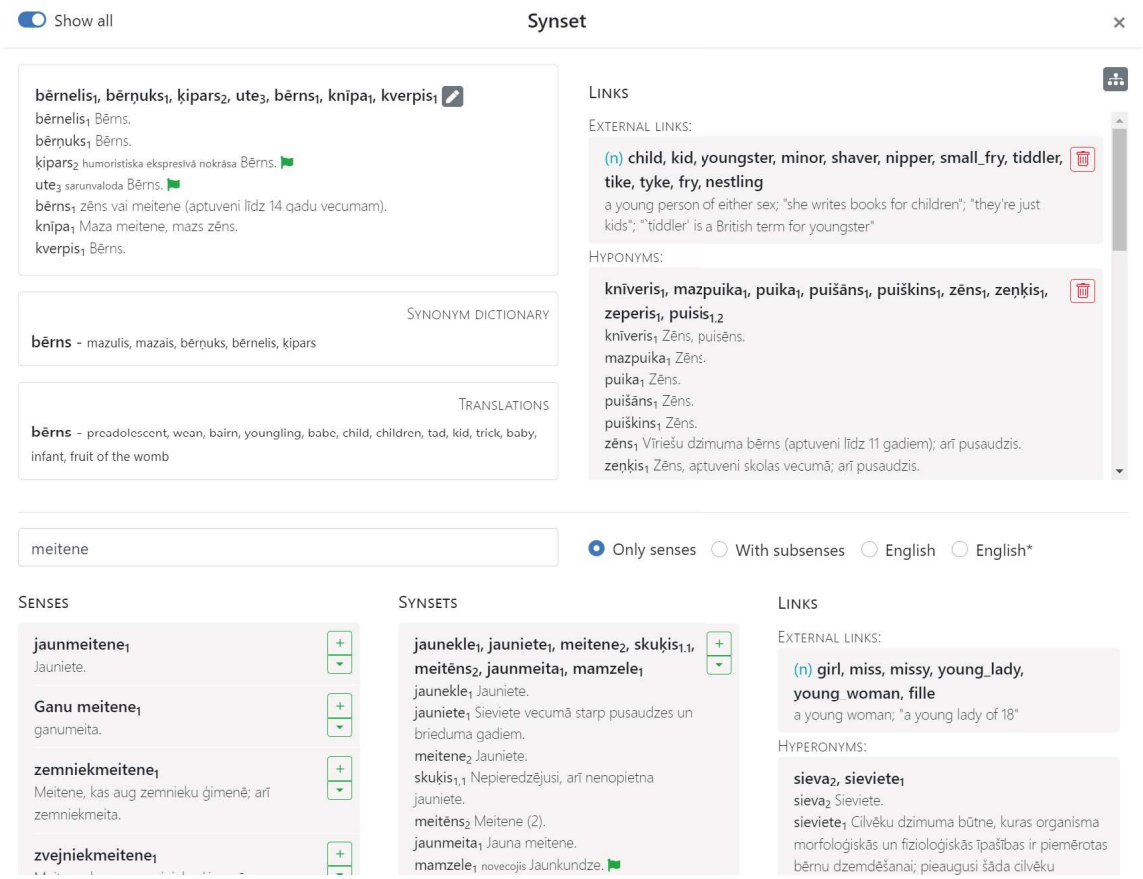


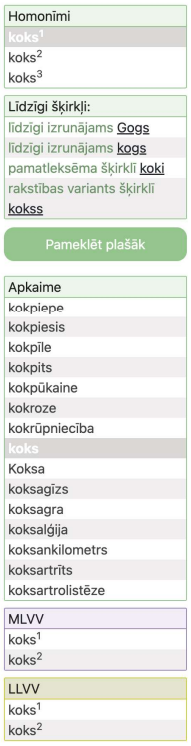
Figure 5: Synset edit view for a sense of the word *bērns* 'child'. The upper section contains synset information: included senses, synonyms and word translations on the left, a list of linked synsets by type on the right. The lower section allows to add new links by searching within Tēzaurs.lv or Princeton WordNet. The image shows Tēzaurs.lv results for a search query *meitene* 'girl'. The first result column contains a list of senses that are not yet in Latvian WordNet. The second column contains a list of Latvian WordNet synsets. By clicking on any of these synsets, a list of all its links is displayed in the third column.

If a match to the search prompt is found, the corresponding entry is opened in the main area.

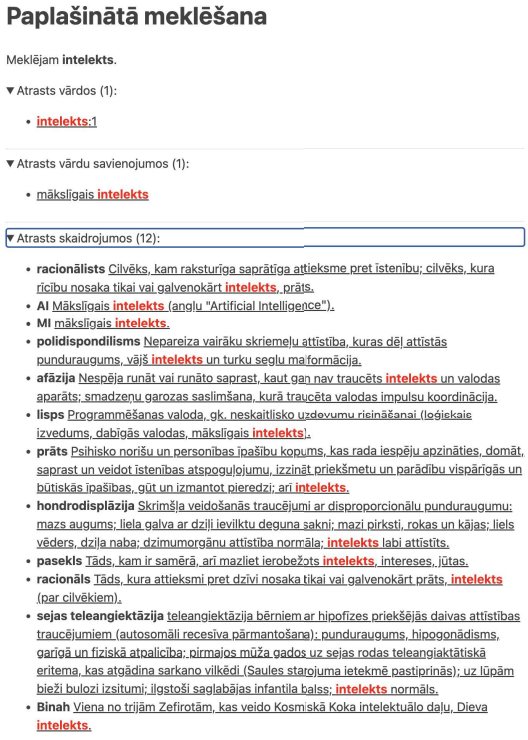
On the left side, several boxes may be shown each containing links to neighbour entries of various kinds of neighbourhood (see Figure 6a): homonym entries, homoforms in other entries, entries having inflectional forms similar to the search prompt, similarly spelt words, alphabetical neighborhood with adjacent entries of the same type (words, MWEs, word parts), and search word in other dictionaries.

The right side is reserved for graph relations between entries.

If in the first pass no satisfying entry (i.e., matching the search prompt) has been found, a deeper search is performed, looking also into the glosses etc., and the search results are presented grouped by match place and type, as shown in Figure 6b.



(a) Results of basic search for the prompt *koks*.



(b) Results of extended search for the prompt *intelekts*.

Figure 6: Basic and extended search.

Additionally, the word is looked up in other dictionaries hosted on the Platform, and in case of success the links are provided for opening the corresponding entry in sibling dictionaries.

1.2 Inflections and morphology

We generate and display inflection tables for lexemes that have their morphological paradigm specified, as illustrated in Figure 7. The inflections are generated by an external morphology engine (Pretkalniņa & Paikens, 2018) and fetched via an API call. The engine returns wordforms and certain lexical flags which are then used for both generation of inflection tables and searching for inflected word forms.

2. Data model of the dictionary

While previously the data model and tools were based on what the end user would see in a dictionary entry (document based model where the documents were dictionary entries), the current infrastructure is designed with a focus on a maintainable structured model – a graph which consists of lexical entities and links between them, thus avoiding duplication and enabling persistent links that stay consistent even if word senses are edited or moved. For example, multi-word entities used to be listed separately in the entry of words referring to it, duplicating the data with some accidental variation, but now both entries include the same entity.

Figure 7: Expanded block with inflection information for the verb *darīt*

This design permitted us to start with a more classical, entry oriented data structure, coming from the paper age, and incrementally move towards a graph oriented data model.

This highly structured approach simplifies exporting data for various purposes. Currently, we have TEI[7] for most dictionary data and LMF[8] for WordNet related data. Additionally, an export to the PINI tool (Barzdins et al., 2020) has been developed for marking word senses in literary texts.

2.1 Core structure

The new data model of the dictionary (see Figure 8) consists of entities (Entry, Lexeme, Sense, Example, Synset) and links between them. The main root elements in the data model are Entry and Synset.

The nature of the new data model is a hybrid between a document-based and a graph-based model: some of the relations are represented as graph edges between entities (many-to-many, either symmetric or asymmetric), some others are based on one-to-many relations (Entry ← Sense, Entry ← Lexeme, Synset ← Sense, Sense ← Subsense, etc.). Symmetric links are used in WordNet between synsets to represent relations "anthonymy" and "similar". Asymmetric links depict also the direction of a relation, e.g., "A derivativeOf B" tells that A is derived from B.

2.1.1 Entry

An Entry roughly corresponds to the entry in a traditional dictionary. However, most of the lexicographic information is delegated to other entities, and the entry itself serves just as the point joining these entities together. Lexemes, senses, examples, and sources of lexicographic information all are attached to the Entry. An entry can have a link to another entry if it is a derivative of a word with its own senses (`derivativeOf`) or an entry of a MWE that contains this word (`hasMWE`). In rare cases, a `seeAlso` link is used between an entry and another entry to indicate some kind of relationship between the words that

[7] Text Encoding Initiative, P5 Guidelines, Chapter 9: Dictionaries, available: https://tei-c.org/release/doc/tei-p5-doc/en/html/DI.html

[8] Global WordNet Association, guidelines for formats: https://globalwordnet.github.io/schemas/#xml

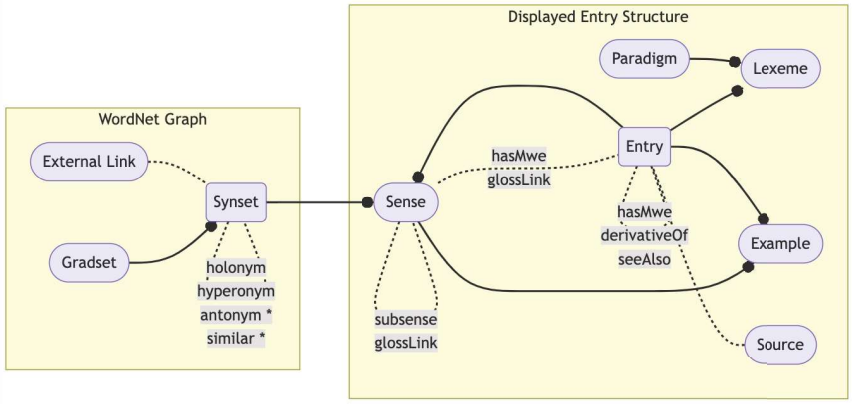


Figure 8: Conceptual data model. Continuous line: one-to-many relation (bullet denotes the singular end of the relation); dotted line: a many-to-many relation between entities, which can be symmetric (with *) or asymmetric.

are not in the derivation relations, nor in the WordNet-defined relations between word meanings.

2.1.2 Lexeme

Each Lexeme is built around one lemma, together with all information related to it. An entry has one or more lexemes attached to it. A lexeme may have a paradigm assigned to it. Paradigm has its own table in the database. Each lexeme has a type: main lexeme, spelling variant or derivation lexeme. Even though there is no direct database relation linking senses with lexemes, the lexeme type provides us information about how the senses in the entry are related to the lexemes in the same entry. By default, a sense in the entry provide definition for all main lexemes and spelling variants, but not derivation lexemes. However, with the use of Restriction (see 2.2.3) any given Sense can be targeted to a specific derivation lexeme.

2.1.3 Sense

An entry has one or more Senses and subsenses attached to it. The main content of a sense is its textual description (gloss). In order to fine-tune the meaning described, a sense may have one or more examples attached to it and one or more MWE entries linked to it, in which the word is used in this specific sense. Senses can be organized in a two level hierarchy – senses and subsenses. A gloss of a sense can have “anchor” links that create a link from a word used in a gloss to another entry or to a particular sense of another entry. “Anchor link” is asymmetrical. Examples of word usage where the word is used in a specific sense may be attached to the meaning. Sense is also an element in creating the Latvian WordNet, so a link to the WordNet synset can be made from it.

2.1.4 Synset

A WordNet core element is a Synset, which can be composed of either one or more Senses, usually coming from different dictionary entries. WordNet links are drawn between synsets, not directly between senses. Two or more synsets can be involved in a larger set "Gradset". A synset can have one or more External Link attached, showing the relation between Latvian synset and a related entity in other, external lexicographical resources. Currently, synsets are linked to the corresponding Princeton WordNet synset, but links to other resources may be added later.

2.1.5 Example

An Example consists of a text fragment together with information about its origin. Normally, examples are attached to a sense, but the structure supports also examples on the entry level.

2.2 Supplementary non-tabular data

In addition to the core tabular data structure, each data item (entity or link) can be enriched with supplementary non-tabular information in the form of structured JSON data. This approach enables the inclusion of more detailed, complexly structured information, while also allowing for uncomplicated data model extension without altering the database structure. Certain elements of the JSON data (`Flags`, `StructuralRestrictions`) are predefined, while the data can be easily expanded with new components (e.g., `Pronunciations`, `Etymology`, `Normative`, `ImportNotices`, `sketchEngineTokenNum`, etc.). This approach also keeps open the option to move some parts of the JSON data over to relational database tables if such optimization needs should arise.

2.2.1 Paradigms

In this model, Paradigm is a named category that defines a set of inflection rules and flags which can be assigned one for each lexeme. Lexeme inherits certain properties, such as part of speech or grammatical gender, from the assigned paradigm, but each of these inherited properties can be overridden by flags defined at the lexeme level. Currently around one third of all single-word lexemes has a paradigm assigned, however, the desirable future state would be to have the paradigms for most if not all single word lexemes.

Providing paradigm for lexeme ensures that the morphological analyzer (Paikens et al., 2013) can be used to generate all inflectional forms for given word. These forms are further used to improve dictionary search and to generate inflection tables shown to the user.

2.2.2 Flags

A part of the payload information in data items is structured as Flags. A flag is a key-value pair, where the key is a descriptive name, and the value can be a string or a list of strings. The definition of a flag type usually contains a set of permitted values, however, free entry values are also supported; additionally, a flag type definition prescribes cardinality (single

value, or list of values) and scope (for which entity types the flag can be used). Flag type definitions are stored in the database, thus enabling to provide convenient UI components for entering/editing the flag assignments.

Currently, there are 64 predefined flag types, with 470 predefined values in total. Both flag types and flag values can be marked as deprecated, thus supporting evolution of the flag-set. Some examples of flag types from different aspects are: *POS*, *Conjugation*, *Transitivity*, *Tense*, *Pronoun type*, *Domain*, *Style*, *Language*, *Dialect features*.

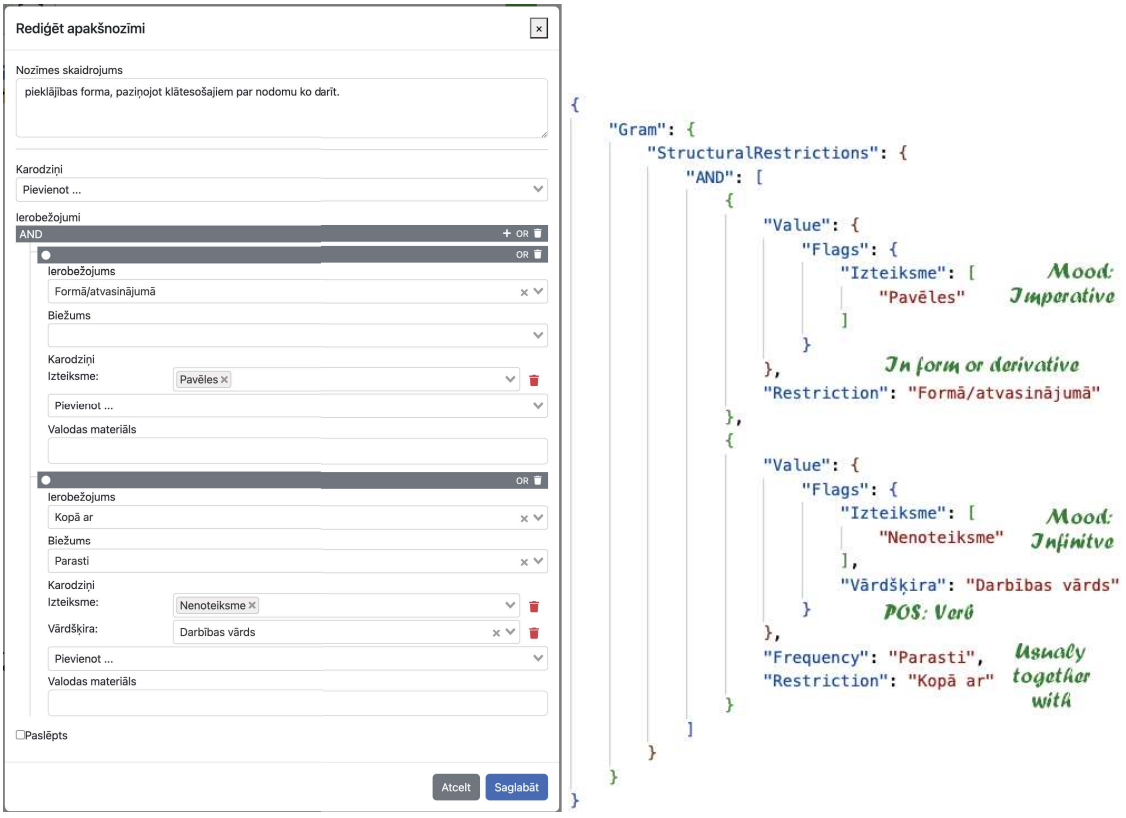
2.2.3 Restrictions

In order to represent additional contextual grammatical or usage restrictions on some entities, the Restriction data structure has been created. These restrictions can be attached to any entity, besides the set of flags. These restrictions describe things like the fact that certain sense in the entry is used only for certain wordforms of the lexeme, or when lexeme is used in certain grammatical structure (see example in Figure 9) or that a certain lexeme might be used only in some of the forms its inflectional paradigm formally prescribes.

Each such restriction consists of 3 parts: restriction type, restriction frequency and restriction's "value" – a set of attribute-value flags. Restriction type broadly classifies all restrictions in several groups by their functioning (see below). Restriction frequency loosely describes how often this restriction is applicable, and currently their values are, e.g., `always`, `often`, `rare` and unspecified as inherited from retro-digitised paper dictionaries. However, we envision possible improvement here by switching to data-backed frequencies. The third part of the restriction structure is set of flags. Flags here are the same as described above (see Section 2.2.2) and they describe actual properties we want to restrict by, e.g., `Case=Nominative`. This part can also contain a free-text string that describes some kind of language material – either some phrase or certain word form.

Currently, the platform supports following 6 Restriction types:

- `togetherWith` – denotes usage together with certain parts-of-speech, lexemes or forms
- `inStruct` – denotes usage in certain structures, e.g., exclamation sentences
- `inForm` – denotes selection of certain inflectional forms or derivative of the lexeme in question
- `wordbuildingPart` – one of the restriction types meant specifically for entries describing parts of the words: this restriction is used for describing the other parts of the compound to be made
- `wordbuildingResult` – another one of the restriction types meant specifically for entries describing parts of the words: this restriction is used for describing resulting compound
- `overallFrequency` – for cases when certain sense is described as *rare* or *often* we decided to use restriction with this type and appropriate restriction frequency, but without any flags, not to duplicate restriction frequencies as flag values



(a) The editing dialog of a structural restriction.

(b) The corresponding JSON data.

Figure 9: The restriction for sense 1.2 of the word *atļaut*: this sense is expressed with imperative form and usually in a phrase with an infinitive verb.

2.3 The Technical Stack

The application uses *node.js*[9] as the application host, with *express.js*[10] as the HTTP server, and *pug.js*[11] as the template engine for server side rendering. Mixins are extensively utilized for rendering repetitive components in the interface.

vue.js[12] forms are used as self-contained independent components providing property editing dialogues for each entity type. These forms communicate directly with the backend, which is responsible for data validation and persistence.

PostgreSQL[13] is used as the database engine. Each entity type and link type has its own table in the database. Currently, the data model consists of approximately 40 database tables, including administrative and supporting tables. Operations of larger scale, such as the merging of two entries, are implemented as database procedures. Change logging is also implemented as a database trigger function.

[9] https://nodejs.org/en
[10] https://expressjs.com/
[11] https://pugjs.org
[12] https://vuejs.org/
[13] https://www.postgresql.org/

The dictionary application is being deployed on *Ubuntu Linux*[14] as a *Docker*[15] container, with *nginx*[16] as a reverse proxy engine in front of it, and *PostgreSQL* installed directly on the server.

In publishing mode the application is currently serving a moderate workload of up to 200K requests per day. If future increases in workload would cause performance issues, there are several easy yet unexplored optimization possibilities. Additionally, if necessary, load balancing can be implemented across multiple servers. In editing mode, the app supports multiple named editors who can work concurrently.

3. Tools for lexicographic workflow

3.1 Verbalization of structured data

The grammatical and usage information for the dictionary entities (entries, lexemes, senses, and examples) is stored as structured data in the form of paradigms, flags and structural restrictions. The verbalization module generates a human-readable textual representation of this information. Verbalization builds upon atomic rules for simple flags, enhanced with aggregation rules for logical expressions, overriding rules for specific over general, prioritization rules for placing the most important facts at the beginning, etc. Sample results of verbalization are shown in Figure 10.

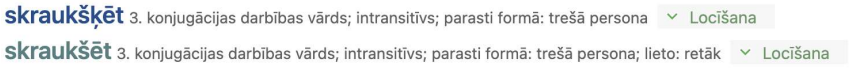


Figure 10: Verbalization results (in gray) for two lexemes.

3.2 Queries / Reports

To support the lexicographers, a query subsystem has been created, which allows to define reusable queries for finding entities satisfying a specific lexicographic criteria as well as data validation queries. This module enables users to define and reuse queries and presents the results as navigable tables or lists of dictionary items (an example see in Figure 11). The system supports both pure SQL queries and SQL+code queries. Currently, the system comprises around 100 queries of varying complexity.

3.3 Interface for bulk-editing

To support bulk-editing of some aspects in the lexicographer's work which cannot be fully automated, a special module has been created which presents a list of micro-tasks to the editor, who can select one of the quick choices, or open the entry for regular editing in the unclear cases (see Figure 12).

[14] https://ubuntu.com/
[15] https://www.docker.com/
[16] https://www.nginx.com/

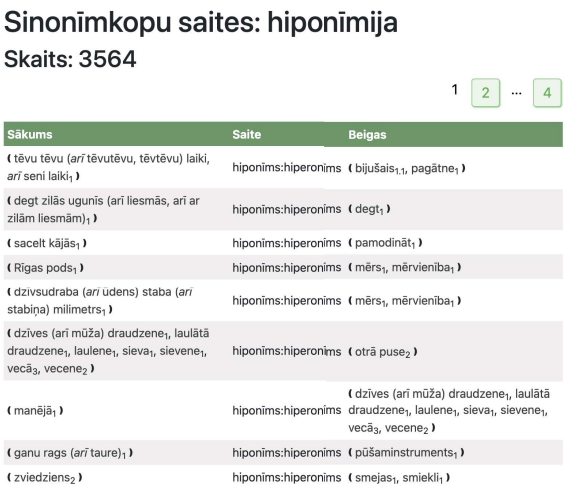


Figure 11: Result fragment of a query for hyponomy links – first and last columns contain link endpoints, middle column displays link type (direction).

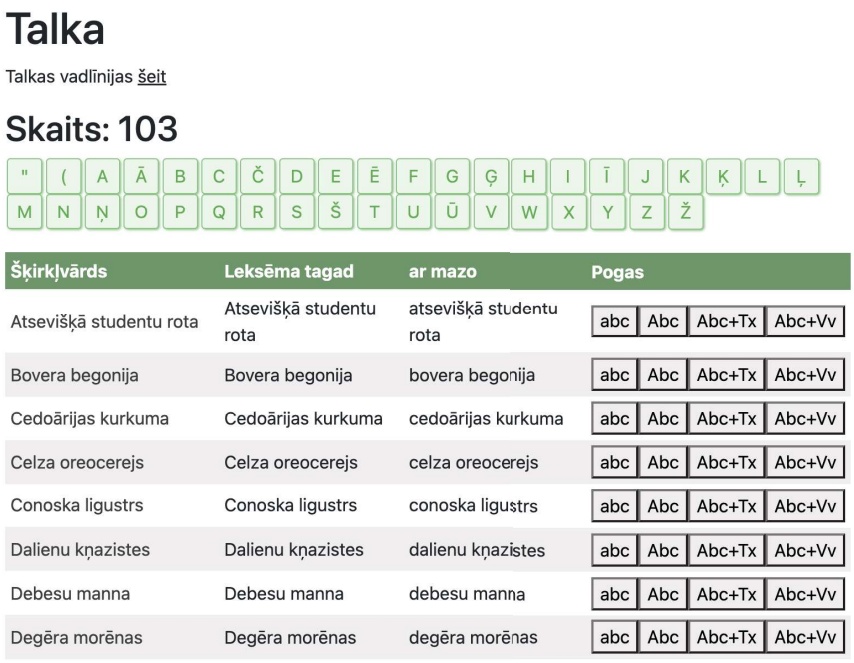


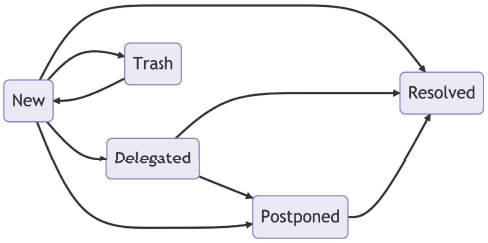
Figure 12: Bulk-editing interface. Task: capitalization of multi-word expressions; table columns contain entryword, current lexeme, task specific automatically provided potential correction and buttons for answering.

3.4 Collecting statistics

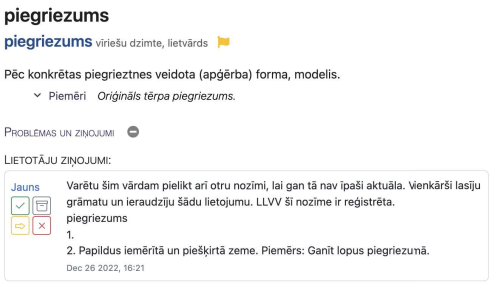
In the publishing mode, the application reports statistics on successful entry requests, as well as on failed (entry not found) entry requests. The log of not found requests is utilized to further enhance the dictionary content.

3.5 Collecting user feedback and suggestions

The platform provides a system that enables end-users to provide feedback and suggestions for any dictionary entry. All suggestions entered are stored in the database, and a simple workflow is provided to facilitate the feedback processing (see Figure 13a).



(a) Workflow for user feedback processing.



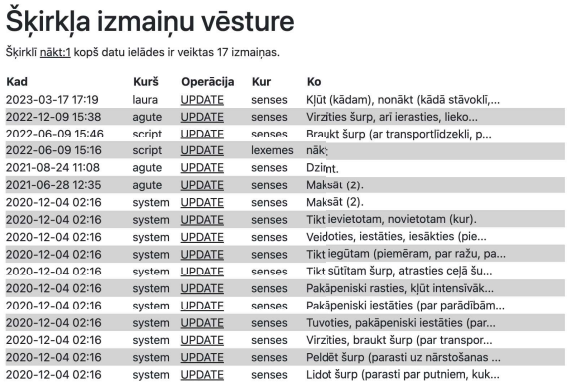
(b) A new feedback message embedded in the editor's view of an entry.

Figure 13: User feedback processing.

All feedback can be viewed either in a list by workflow state, or in the related entry as an embedded block with message and action buttons (see Figure 13b).

3.6 Keeping change history

In the editing mode, the application records all changes made to the dictionary at the entity level. This includes information such as who made the change, when it was made, the type of operation performed, and the data before and after the change. These logs (see Figure 14) enable editors to trace the change history and resolve any errors or misunderstandings that may arise during the editing process.



Šķirkļa izmaiņu vēsture

Šķirklī nākt:1 kopš datu ielādes ir veiktas 17 izmaiņas.

Kad	Kurš	Operācija	Kur	Ko
2023-03-17 17:19	laura	UPDATE	senses	Kļūt (kādam), nonākt (kādā stāvoklī,...
2022-12-09 15:38	agute	UPDATE	senses	Virzīties šurp, arī ierasties, lieko...
2022-06-09 15:46	script	UPDATE	senses	Braukt šurp (ar transportlīdzekli, p...
2022-06-09 15:16	script	UPDATE	lexemes	nākt
2021-08-24 11:08	agute	UPDATE	senses	Dzimt.
2021-06-28 12:35	agute	UPDATE	senses	Maksāt (2).
2020-12-04 02:16	system	UPDATE	senses	Maksāt (2).
2020-12-04 02:16	system	UPDATE	senses	Tikt ievietotam, novietotam (kur).
2020-12-04 02:16	system	UPDATE	senses	Veidoties, iestāties, iesākties (pie...
2020-12-04 02:16	system	UPDATE	senses	Tikt iegūtam (piemēram, par ražu, pa...
2020-12-04 02:16	system	UPDATE	senses	Tikt sūtītam šurp, atrasties ceļā šu...
2020-12-04 02:16	system	UPDATE	senses	Pakāpeniski rasties, kļūt intensīvāk...
2020-12-04 02:16	system	UPDATE	senses	Pakāpeniski iestāties (par parādībām...
2020-12-04 02:16	system	UPDATE	senses	Tuvoties, pakāpeniski iestāties (par...
2020-12-04 02:16	system	UPDATE	senses	Virzīties, braukt šurp (par transpor...
2020-12-04 02:16	system	UPDATE	senses	Peldēt šurp (parasti uz nārstošanas ...
2020-12-04 02:16	system	UPDATE	senses	Lidot šurp (parasti par putniem, kuk...

Figure 14: List of change history for the verb *nākt*

4. Use cases

The primary role of Tēzaurs.lv is as a multi-functional online dictionary designed to meet the needs of a diverse range of users. We provide search results based on inflectional forms and spelling variants, as well as links to phonetically similar, alphabetically adjacent, and semantically linked words. For the entries of Latvian WordNet we also provide translations allocated to specific senses, which helps language learners and translators. To manage the extensive information, we use openable/closable blocks to display the data, rather than hiding it. We have also redesigned the interface with the understanding that the amount of data may continue to expand in the future. This allows each user to explore the data as deeply as they desire (see Figure 15). This approach makes Tēzaurs.lv a valuable resource for a wide range of end-users, including language learners, students, translators, and the general population.

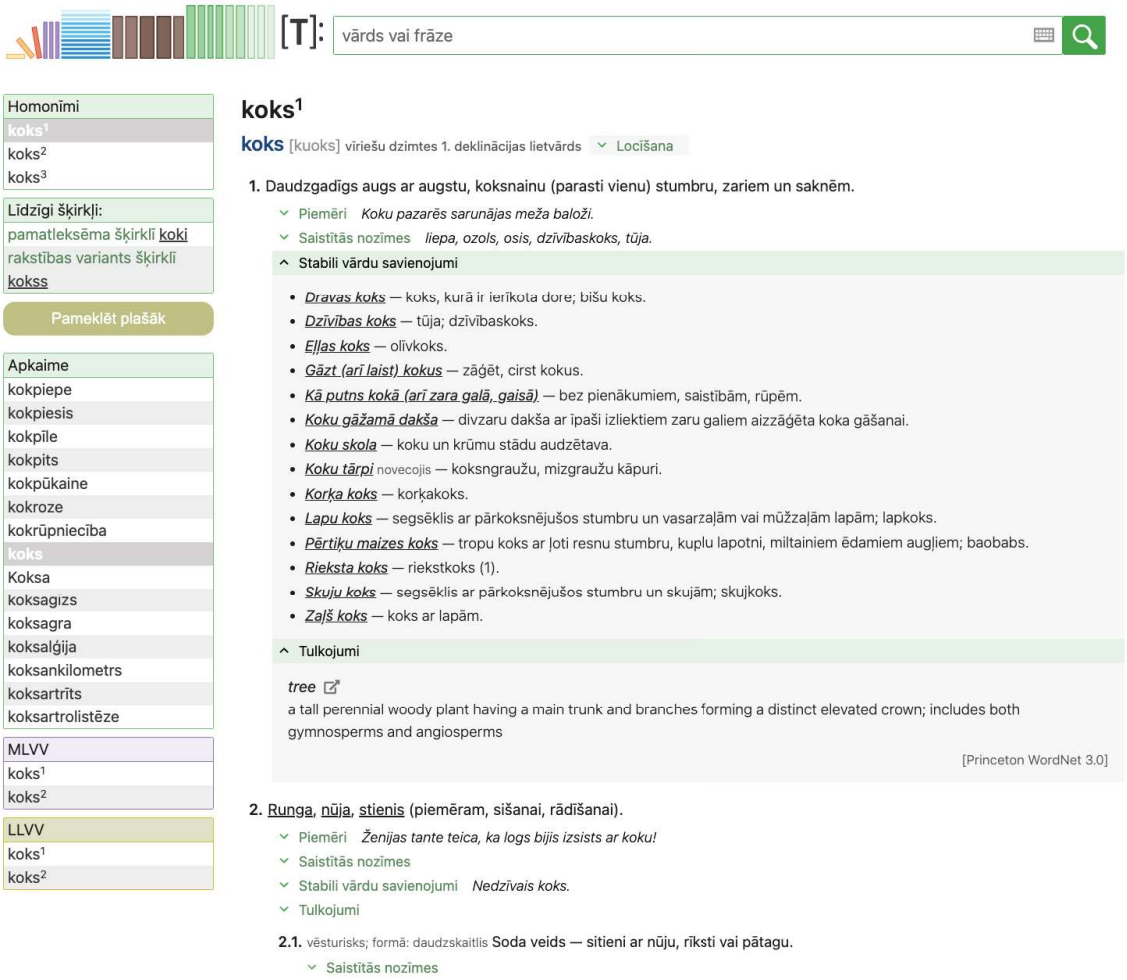


Figure 15: The public view of the entry *koks* ('tree') with opened interface panels for MWEs and translations.

Secondly, the created platform provides wide functionality for dictionary editors. Despite the fact that the information to be included in the dictionary can be quite extensive and

structurally complex (as in cases with restrictions), according to Tēzaurs.lv editorial team, the dictionary editor is quite convenient, intuitive and user-friendly. Furthermore, it can be adapted for the creation and development of other dictionaries with minimal modifications, as demonstrated by its successful use with two other dictionaries. Key advantages and benefits:

- The system efficiently handles even relatively large dictionaries
- Editor authentication/authorization, change history tracking, and parallel work capabilities (with a limitation that multiple editors cannot edit the same entity (lexeme, sense, etc.) at the same time)
- Queries and reports enable data review from diverse angles, with future plans to support more advanced searches (e.g., using regular expressions)
- Where feasible, data entry utilizes predefined lists to minimize errors by editors, and incorporates routinely updated automatic error checks to accommodate emerging requirements
- User feedback storage and processing, including content error reporting as well as suggestions for new entries or clarifications, is fully integrated into the system, eliminating the need for e-mail communication
- Offers a visually appealing web-based interface, reducing compatibility issues across different operating systems
- Streamlines the creation of interfaces for bulk data processing and facilitates regular cleanups to address more uniform issues (e.g., sorting MWEs by types such as toponyms, taxons, other proper names, etc.)

4.1 Support for multiple dictionaries

Currently, the platform is used for 3 different dictionaries already published online: for Tēzaurs.lv, for the actively developed MLVV (Dictionary of Modern Latvian) (Zuicena, 2012), and for a retro-digitised version of the earlier authoritative dictionary of Latvian – LLVV (Dictionary of Standard Latvian)[17]. All 3 dictionaries are served from the same application code, pointing to different databases. The only differences between them are the used style sheets and some conditionally excluded features that are not required for a specific dictionary.

In general, the platform is designed to be extensible and adaptable for a multitude of future uses, too. We are starting to develop a Latgalian dictionary on this platform, and hope in future to add several retro-digitised dictionaries. While we focus more on Latvian, the core platform itself is language independent as long as someone translates the user interface, updates the flag sets, the morphological analyser integration, the verbalization, and if target language uses similar dictionary structure.

5. Conclusions and future work

The choice of a hybrid data model has permitted to evolutionary move from an entry-oriented view towards more graph-oriented data structures, as well as to support dictionaries of various level of formalization and improve the formalization of Tēzaurs.lv itself.

[17] https://llvv.tezaurs.lv

When building such a diverse and multi-functional lexical resource, precise and task specific tool support turned out quite crucial even if our team is quite small. The searching and error-checking abilities of the developed system majorly improved both the speed of content creation and the quality of the result.

Another advantage of this approach is its flexibility to enable supplementary micro-tools which build upon the shared core data model. Currently the synthesizer of morphological forms is integrated in this manner, but in future we would like to add other micro tools as well, such as generation of pronunciation samples. We have also recently started a project on extending Tēzaurs.lv with additional lexicographic data (namely, etymology and derivation links) and this platform enables us to include them as extra information in a shared lexical resource, instead of creating a separate resource like Derinet (Vidra et al., 2019) which afterwards could diverge from the continuously maintained dictionary.

Future platform improvements include extending the graph related features of the data model both on the data model level and on the visualization level. We plan to extend the reach of this work by publishing the source code of the platform under the GPL licence. We also plan to use Tēzaurs.lv platform to host even more dictionaries, both including other retro-digitised dictionaries and providing a platform for Latvian researchers to create new digital dictionaries.

We hope that this experience will be useful for other researchers building lexical resources and tools for maintaining them.

6. Acknowledgements

This work was supported by the Latvian Council of Science, project "Advancing Latvian computational lexical resources for natural language understanding and generation" (LZP-2022/1-0443) and State Research Programme project "Research on Modern Latvian Language and Development of Language Technology" (VPP-LETONIKA-2021/1-0006). We also thank the anonymous reviewers for their input in improving this paper.

7. References

Barzdins, G., Gosko, D., Cerans, K., Barzdins, O.F., Znotins, A., Barzdins, P.F., Gruzitis, N., Grasmanis, M., Barzdins, J., Lavrinovics, U., Mayer, S.K., Students, I., Celms, E., Sprogis, A., Nespore-Berzkalne, G. & Paikens, P. (2020). Pini Language and PiniTree Ontology Editor: Annotation and Verbalisation for Atomised Journalism. In *Proceedings of the 17th Extended Semantic Web Conference (ESWC): Posters and Demos*. URL https://preprints.2020.eswc-conferences.org/posters_demos/paper_281.pdf.

Danovskis, G. (2014). *Leksikogrāfa rīks kļūdu meklēšanā*. Kvalifikācijas darbs, Latvijas Universitāte.

Fellbaum, C. (ed.) (1998). *WordNet: An electronic lexical database*. MIT Press.

Paikens, P., Grasmanis, M., Klints, A., Lokmane, I., Pretkalnina, L., Rituma, L., Stade, M. & Strankale, L. (2022). Towards Latvian WordNet. In *Proceedings of the 13th Language Resources and Evaluation Conference (LREC)*. pp. 2808–2815. URL http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.300.pdf.

Paikens, P., Gruzitis, N., Rituma, L., Nespore, G., Lipskis, V., Pretkalnina, L. & Spektors, A. (2019). Enriching an Explanatory Dictionary with FrameNet and PropBank Corpus

Examples. In *Proceedings of the 6th Biennial Conference on Electronic Lexicography (eLex)*. pp. 922–933. URL https://elex.link/elex2019/wp-content/uploads/2019/09/eLex_2019_52.pdf.

Paikens, P., Klints, A., Lokmane, I., Pretkalnina, L., Rituma, L., Stade, M. & Strankale, L. (2023). Latvian WordNet. In *Proceedings of the 12th Global Wordnet Conference (GWC2023)*. To be published.

Paikens, P., Rituma, L. & Pretkalnina, L. (2013). Morphological analysis with limited resources: Latvian example. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA)*. URL http://stp.lingfil.uu.se/nodalida/2013/pdf/NODALIDA24.pdf.

Pretkalniņa, L. & Paikens, P. (2018). Extending Tēzaurs. lv Online Dictionary into a Morphological Lexicon. In *Human Language Technologies–The Baltic Perspective*. IOS Press, pp. 120–125.

Rambousek, A., Jakubíček, M. & Kosem, I. (2021). New developments in Lexonomy. In *Electronic lexicography in the 21st century. Proceedings of the eLex 2021 conference*. pp. 455–462. URL https://elex.link/elex2021/wp-content/uploads/2021/08/eLex_2021_28_pp455-462.pdf.

Saulite, B., Dargis, R., Gruzitis, N., Auzina, I., Levane-Petrova, K., Pretkalnina, L., Rituma, L., Paikens, P., Znotins, A., Strankale, L., Pokratniece, K., Poikans, I., Barzdins, G., Skadina, I., Baklane, A., Saulespurens, V. & Ziedins, J. (2022). Latvian National Corpora Collection – Korpuss.lv. In *Proceedings of the 13th Language Resources and Evaluation Conference (LREC)*. pp. 5123–5129. URL http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.548.pdf.

Simões, A., Salgado, A., Costa, R. & Almeida, J. (2019). LeXmart: A Smart Tool for Lexicographers. In *Proceedings of eLex 2019 conference*. pp. 453–466. URL https://elex.link/elex2019/wp-content/uploads/2019/09/eLex_2019_25.pdf.

Spektors, A., Auzina, I., Dargis, R., Gruzitis, N., Paikens, P., Pretkalnina, L., Rituma, L. & Saulite, B. (2016). Tezaurs.lv: the largest open lexical database for Latvian. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*. Portoroz, Slovenia, pp. 2568–2571.

Vidra, J., Žabokrtský, Z., Ševčíková, M. & Kyjánek, L. (2019). DeriNet 2.0: Towards an All-in-One Word-Formation Resource. In *Proceedings of the Second International Workshop on Resources and Tools for Derivational Morphology*. Prague, Czechia: Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, pp. 81–89. URL https://aclanthology.org/W19-8510.

Zuicena, I. (2012). Ilustratīvā materiāla atveide "Mūsdienu latviešu valodas vārdnīcā". *Baltistica*, (8), pp. 181–188. URL http://www.baltistica.lt/index.php/baltistica/article/view/2120/2026.