Introducing Lexonomy: an open-source dictionary writing and publishing system

Michal Měchura

Natural Language Processing Centre, Masaryk University, Brno, Czech Republic
E-mail: michmech@mail.muni.cz

Abstract

This demo introduces Lexonomy `www.lexonomy.eu`, a free, open-source, web-based dictionary writing and publishing system. In Lexonomy, users can take a dictionary project from initial set-up to final online publication in a completely self-service fashion, with no technical skills required and no financial cost.
Keywords: dictionary writing systems; online dictionaries; XML editing

1. Introduction

Lexonomy is a web-based platform for writing and publishing dictionaries. Its mission is to be an easy-to-use tool for small to medium-sized dictionary projects. In Lexonomy, individuals and teams can create a dictionary, design an arbitrary XML structure for the entries, edit entries, and eventually make the dictionary publicly available as a 'microsite' within the Lexonomy website. Lexonomy exists in order to lower the barriers of entry into modern born-digital lexicography.[1] Compared to other dictionary writing systems[2] it requires no installation or set-up, expects no knowledge of coding or programming,[3] and is free from financial cost. It is simply a website where lexicographers can sign up and start working.

Each Lexonomy user logs in with a user name and password. Users are allowed to create an unlimited number of dictionaries. The process of creating a dictionary consists of deciding what it should be called (this can be changed later) and what its URL should be, for example `www.lexonomy.eu/mydictionary`. This is the address at which the dictionary will eventually be publicly viewable, if and when its creators decide to make it public. By default, newly created dictionaries are not publicly viewable.

Once a dictionary has been created, the user who created it may add additional users and these can all start adding and editing entries. The rest of this introduction to Lexonomy will unfold in a logical sequence. Fist we will introduce features related to **dictionary planning**: specifying the structure of entries etc. Second, we will look at **dictionary editing** with Lexonomy's built-in XML editor. Third, we will show how Lexonomy can be used as a platform for online **dictionary publishing**.

2. Entry structure

Dictionary entries in Lexonomy are stored as XML documents and their structure is defined by a schema which is unique to each dictionary. Users can choose a predefined

[1] One implication of this is that Lexonomy is not a good match for retro-digitized dictionaries.
[2] The reader is kindly asked to read what follows as a mission statement rather than an empirically verified fact. A thorough comparison of existing dictionary writing systems is beyond the scope of this paper.
[3] Familiarity with XML is a plus but Lexonomy users are not expected to be able to hand-code XML.

schema while creating a new dictionary (the options are *monolingual dictionary*, *bilingual dictionary* and so on) and customize it later or, if they prefer, they can start from a completely blank schema.

A Lexonomy schema is similar to a DTD (Document Type Definition): it lists the XML elements which are allowed to appear in the entries and specifies how they may be nested, how many of them must or may be there, which attributes they may or must have, what their values may be and so on. In a conventional dictionary writing system the schema would typically be hand-coded by an IT specialist. Lexonomy, on the other hand, offers a visual schema editor where users can define the structure of their entries without having to hand-code anything.

The left-hand side of the screen (Fig. 1) contains a list of XML elements and attributes. The tree structure indicates how they may be nested, such that the top-most element will be the root element of every entry. It is up to the user to decide what the elements and attributes are called and how they are nested. The right-hand side of the screen then contains detailed settings for the selected element or attribute: this is where the user specifies what child elements or attributes the element may contain, in what order, how many of each, and what content they are allowed to hold.

2.1 Element content

The content of each element can be constrained by making a choice from these options:

- *Child elements*: elements of this name will contain other elements.
- *Text*: elements of this name will contain plain text.
- *Text with markup*: elements of this name will have mixed content (= plain text interlaced with other XML elements).
- *Value from list*: elements of this name will contain a value from a predefined list.
- *Empty*: elements of this name will have no content.

Depending on the type of content chosen, the schema editor will offer different additional options. If the content is *Child elements* or *Text with markup*, we can specify the child elements as in Fig. 1. The *min* and *max* numbers control how many instances of the child element must be present inside the parent element: *min* = 1 and no *max* means 'one or more', *max* = 1 and no *min* means 'none or one', no *min* and no *max* means 'none, one or more', and so on. We will see in a later section how Lexonomy imposes these constraints while the lexicographer is editing an entry.

If the element's content is set to *Value from list*, we can specify the values on that list, along with optional captions (Fig. 2). We will see later how Lexonomy's XML editor makes use of this setting by giving the lexicographer a menu to choose from when inputting an attribute value. The captions are used instead of values for visualization to end-users.

2.2 Attributes

Besides child elements, XML elements in Lexonomy can have XML attributes. When specifying that an element can have an attribute, we can declare the attribute optional or obligatory, as in Fig. 1. Further settings for attributes are a subset of those for elements: an attribute's content can be either *Text* or *Value from list* (Fig. 3).

According to the XML standard[4], the attributes of an element are considered unordered: the order in which they appear in the XML document is insignificant. But, as a convenience to human users, Lexonomy makes sure that attributes always appear in the order in which they are listed in the schema.

2.3 Element nesting

It is possible in Lexonomy for elements of a certain name to appear as children under parent elements of more than one type. For example, if your dictionary has separate elements for senses and subsenses, say `<sense>` and `<subsense>`, they can both have child elements called `<definition>`, `<example>` etc. (Fig. 4). Element nesting can be recursive, too: it is possible to allow `<sense>` elements to appear inside `<sense>` elements (Fig. 5).

2.4 Expressivity of the schema formalism

The schema formalism used internally by Lexonomy and exposed through its schema editor is approximately as expressive as a DTD (Document Type Definition). The only major point of difference is how child elements are ordered. In Lexonomy, child elements (under parents whose content is *Child elements*) must appear in exactly the same order in which they are given in the schema, while a DTD allows more flexibility in this regard.

3. Entry editing

Once the structure has been finalized lexicographers can start working on the actual entries. Lexonomy's entry browser and editor offers a familiar interface with an entry list on the left-hand side and entry details on the right-hand side (Fig. 6). Clicking the *Edit* button opens the entry in Lexonomy's built-in XML editor (Fig. 7).

The XML editor in Lexonomy[5] emulates the look and feel of a text editor with syntax highlighting, code folding and autocompletion. It is, however, not a text editor: lexicographers edit XML by clicking on things, selecting options from context menus, selecting attribute values from picklists, dragging and dropping elements around and so on. This serves the dual purpose of making the lexicographer aware that he or she is manipulating XML while simultaneously making it impossible for them to corrupt the entries by entering non-well-formed XML. In fact, no knowledge of XML syntax is needed for working with Lexonomy: the angle brackets and other formalities of XML syntax are merely a kind of 'decoration'. Users who are not comfortable with the XML notation can turn it off completely and switch Lexonomy into *laic mode* (Fig. 8).

3.1 Knowing where to click

Almost everything in the XML editor is clickable:

[4] `https://www.w3.org/TR/REC-xml/#attdecls`

[5] The XML editor is actually a separate software product called Xonomy: `www.lexiconista.com/xonomy`

- Click the name of an element (it its opening or closing tag) to get a menu with options for adding child elements, for adding optional attributes, and also for removing the element itself. The options offered are in accordance with the schema.
- Click the name of an attribute to get a menu with an option to remove the attribute.
- Click the value of an attribute to get a pop-up box for editing the value. This will be either a text box or a menu to choose from a list, as per the schema.
- Click a text node (= a stretch of text between tags) to get a pop-up box for editing the text. Again, this will be either a text box or a menu to choose from a list, as per the schema.

To change the order of elements (for example to re-order senses) or to move an element to a different location inside the entry (for example to move an example sentence from one sense to another) you can use the 'drag handle' (six grey dots) beside the opening tag of each element. As you drag this with the mouse, Lexonomy will show you 'drop targets' (grey spots) in different places in the entry: these are locations where you can legally drop the element you are dragging ('legally' here means 'the schema allows it').

3.2 Keyboard navigation

A frequent complaint by users of web-based editing interfaces[6] is that the work is slow because there is 'too much clicking' involved. For increased productivity and ergonomics, Lexonomy makes it possible for lexicographers to perform the most repetitive tasks with the keyboard as well as the mouse. While editing an entry in the XML editor, the following keyboard shortcuts are available:

- The cursor keys `up` and `down`, `left` and `right` to navigate around the hierarchical structure of the entry, from tag to tag, from attribute to attribute, and so on.
- When an element has the plus sign next to its opening tag, `Ctrl + right` can be used to expand it and `Ctrl + left` to collapse it again.
- Press `Enter` to open the menu or pop-up editor associated with the currently highlighted element, attribute, attribute value or text node. Then press `Esc` to close it again.
- When a pop-up menu is open, use the cursor keys `up` and `down` to move up and down the menu, and `Enter` to select an item from the menu.
- If the entry is very long and has a scrollbar next to it, you can use `Ctrl + up` and `Ctrl + down` to scroll the entry up and down.

These keyboard shortcuts work when the entry editor is focused. If it is not focused (you will know because the keyboard shortcuts are not working) you can press `Alt + right` at any time to focus it. Similarly, you can press `Alt + left` at any time to focus the entry list on the left hand side of the screen. When the entry list is focused, the following keyboard shortcuts can be used:

- The cursor keys `up` and `down` to move up and down the list.
- `Enter` to the currently highlighted entry.

[6] Based on the author's long career in building, and dealing with users of, such interfaces.

- `Ctrl + up` and `Ctrl + down` to scroll the entry list up and down.

Last but not least, the following keyboard shortcuts are available at any time, regardless of which side of the screen is focused:

- When an entry is being displayed on the right-side of the screen, you can press `Ctrl + Shift + E` to open it for editing: this is the same as pressing the *Edit* button. Then press `Ctrl + Shift + E` again to cancel editing and switch back to viewing: this is the same as clicking the *Cancel* button.
- `Ctrl + Shift + S` to save the entry being edited: this is the same as clicking the *Save* button.
- `Ctrl + Shift + N` to start creating a new entry: this is the same as clicking the *New* button.
- `Ctrl + Shift + T` to move move the cursor into the search box in the top left-hand corner of the screen.

In all keyboard shortcuts mentioned here, Mac users can (but do not have to) substitute the `Cmd` key for the `Ctrl` key.

3.3 Editing inline markup

One area which tends to be particularly troublesome for XML editors is 'mixed content': situations in which an XML element contains a mixture of text and other XML elements. Here is how Lexonomy handles it. If the schema says that the content of an element is *Text with markup*, Lexonomy lets the lexicographer edit its text as if it were normal plain text: clicking it opens a pop-up text box. Additionally, a thin grey line appears underneath the text and the lexicographer can click on this to select stretches of text and annotate them with inline XML markup. When a stretch of text is selected, a menu will appear with options for 'wrapping' that selected text with XML elements (see Fig. 9). The options on that menu come from the schema. Once markup has been inserted, it is again possible to click the inline element and a menu will appear with an option to remove ('unwrap') the element.

3.4 Entry validation

While working with an entry, the options that appear in menus and dialogs conform to the dictionary's schema: users are only allowed to add child elements to parents that may have them, and so on. When adding a new element into the entry, Lexonomy will automatically pre-populate the element with everything it needs to have, as per the schema: obligatory attributes, the correct number of child elements and so on. The same happens when creating a new entry: Lexonomy will automatically start you off with a 'prefabricated' blank entry which conforms to the schema as much as possible: for example, if your schema says that every `<entry>` must have at least one `<headword>`, then every new `<entry>` will come with one (empty) `<headword>` already inserted.

As you make changes to the entry, Lexonomy is constantly validating it against the schema. If you make an edit which is not allowed by the schema, such as insert more child entries than the schema allows, Lexonomy will notify you with a small warning triangle next to the offending element or attribute (Fig. 10). As a general rule, however, Lexonomy's approach to entry validation is permissive: it gives warnings but it will not prevent you from saving an invalid entry (= an entry that does not conform to the schema).

4. Advanced settings

Each dictionary hosted in Lexonomy comes with an extensive configuration screen (Fig. 11). Many settings on this screen are of an advanced nature and we will explore some of those in this section.

4.1 Where is the headword?

In Lexonomy, dictionary authors themselves decide what names the XML elements and attributes in their entries will have. There is no requirement to use a standard vocabulary of names such as `<entry>` or `<headword>`, these can have any names at all, including names in other languages than English.[7] But, at the same time, Lexonomy needs to understand what (at least some of) those element names mean. For example, it needs to know where to find the headword in each entry.

The *Headwords* area on the configuration screen is where the dictionary administrator can make such information explicit (Fig. 12). Lexonomy uses this information for various things, including listing the entries by headword in the entry list on the left-hand side of the editing screen. If you make no selection here, Lexonomy will try to guess where the headword is by simply taking the first non-empty text node it finds in each entry.

In many dictionaries, headwords are 'annotated' with additional elements such as homograph numbers and part-of-speech labels. These can be made to appear in the entry list by selecting them in the *Headword annotations* section. Headwords are displayed in bold fond and are searchable (more about searching later), while annotations are displayed in non-bold font and are not searchable, but are taken into consideration for alphabetical sorting.

4.2 Alphabetical order

When listing entries by headword, the question of alphabetical order unavoidably comes up. Alphabetical order depends not only on the alphabet used (Latin, Cyrillic etc.) but also on the language (e.g. *ä* is sorted right after *a* in German but at the end of the alphabet after *z* in Swedish) and, in extreme cases, even on personal preference. Lexonomy takes an agnostic view and allows dictionary authors to set up their own alphabetical order by simply inputting a linebreak-delimited sequence of characters into the *Headwords* area of the configuration screen (see Fig. 12; characters that appear on the same line with a space between them are sorted as if they were the same). There is a default sort order which dictionary administrators can customize, for example by moving characters around

[7] The names of XML elements and attributes in Lexonomy can even contain non-ASCII characters, such as extended Latin characters and characters from other alphabets.

or by adding characters for their language. Alphabetical sorting in Lexonomy is always case-insensitive.

The sorting algorithm supports digraphs, that is, sequences of characters which are sorted as if they were a single character, such as the Czech *ch* which sorts between *h* and *i* or the Welsh *ng* which sorts between *g* and *h*. All the dictionary administrator needs to do is include the digraph in the correct place in the alphabetical order, e.g. `ch` on a separate line between `h` and `i`.

4.3 Search

Another thing which is under the dictionary author's control is the extent to which the dictionary is searchable by typing some text into the search box (in the top left corner of the editing screen, and also on the dictionary's public home page if the dictionary is publicly viewable). By default, searching means searching for headwords, and typing anything into the search box will return a list of entries whose headwords contain that sequence of characters. But dictionary administrators can search-enable other XML elements too, and this is done in the *Search* area of the configuration screen (Fig. 13). For example, if yours is a bilingual dictionary and if you would like reverse searches to be possible, you can search-enable the elements containing the translations. Then, when you search for a sequence of characters, Lexonomy will return a list of entries where either the headword or one of the translations match (Fig. 14).

Search in Lexonomy is always based on simple substring matching: when you search for *go* you will get entries where this sequence of characters occurs in one of the search-enabled elements, regardless of where in the element it is: *gorge*, *mango*, *mongoose* as well as *go* itself. In other words, search in Lexonomy is not linguistically 'clever': it is aware of neither word boundaries nor word inflection (e.g. a search for *bring* does not match *brought*), as these features are language-dependent. One implication of this is that Lexonomy's search functionality is really only suitable for short strings of text (such as headwords and their translations) but will not perform as well as full-text search (e.g. for example sentences or for definitions).

5. Entry formatting

Beside the schema designer and the entry editor, a third crucial feature of Lexonomy is its formatting designer. This is where users can design the visual appearance of their entries. In a conventional dictionary writing system this task would typically be achieved by hand-coding an XSL and/or CSS stylesheet, and an IT specialist would be required for the job. In Lexonomy, users can design the look of their entries themselves, without knowledge of any stylesheet language.

Similarly to the schema editor, the user sees a hierarchical list of elements and attributes on the left-hand side of the screen, while the right-hand side is where he or she sets the formatting properties of the selected element or attribute (Fig. 15). A randomly selected entry is shown on the right on which all formatting changes are previewed in real time to help lexicographers understand the visual impact of their choices.

Under *Visibility* you select whether the element or attribute is shown at all (the default is *Shown* for elements and *Hidden* for attributes), and under *Layout* you select whether the element is separated from other elements by line breaks or not. The rest of the screen is for setting individual formatting properties of the element or attribute:

- *Separation from other content*: the options are *whitespace* or *none*. For inline elements *whitespace* means that there is a space character between it and any elements that precede or follow it. For line-breaked elements *whitespace* means there is an additional amount of vertical space (approximately half the height of a line of text) above and below.
- *Indentation and bulleting*: the options include various kinds of bullets (round, square-shaped etc) and various sense numbering patterns. It goes without saying[8] that senses in Lexonomy are numbered automatically at display-time and that sense numbers should not be included in entries explicitly.
- *Box border*: the options are *dotted*, *thin* and *thick* for putting a visual border around the element.
- *Background colour*: the options are *none*, *yellow*, *blue* and *grey*.
- *Outer punctuation*: these indicate how the element should be separated from other elements by punctuation such as commas, semicolons or brackets.
- *Text colour*: the options are *none*, *red*, *blue*, *green* and *grey*.
- *Text slant*: the options are *none* and *italic*.
- *Text weight*: the options are *none* and *bold*.
- *Inner punctuation*: the options are the same as *outer punctuation* above. The difference is that inner punctuation is inserted in the same colour, slant and weight as the content while outer punctuation is not: it is 'outside' the scope of font formatting.

5.1 Expressivity of the formatting formalism

Depending on your perspective, the formatting properties available in Lexonomy may seem either carefully curated or inconveniently constrained. The truth is a bit of both. Lexonomy's formatting mechanism is certainly not nearly as expressive as stylesheet languages such as XSL and CSS. On the other hand, the full gamut of XSL and CSS would probably be too confusing for the average lexicographer and would likely lead to amateurish misuse. Lexonomy wants all dictionaries to look good in it, but also, it wants lexicographers themselves to be in control of the formatting of their dictionaries – this calls for simplification. Time will tell whether this level of simplification is the right one.

6. Online publishing

Finally, a dictionary can be made available to the public as a 'micro-site' within Lexonomy, e.g. `www.lexonomy.eu/mydictionary`. This does not require any complicated work, the user merely needs to change a few settings in the dictionary's configuration section (Fig. 16).

When a dictionary is made public, Lexonomy gives it a simple user interface which allows the dictionary to be searched and browsed (Fig. 17). The home page offers a random selection of headwords and a search box. Search here works exactly like it does in the edit screen. Each individual entry has its own page with its own URL, and the headword's

[8] But let us say it anyway.

alphabetical neighbourhood is displayed on the side (Fig. 18). The interface is responsive (therefore mobile screen-friendly) and optimized for indexing by search engines.

When a dictionary has been made public, the public interface is of course viewable by anybody, regardless of whether they are currently logged into Lexonomy or not. If logged in, and if the user has editing access to the dictionary, an *Edit* link is shown beside the dictionary title. When a dictionary has not been made public yet, the dictionary's home page is essentially the same but has only the dictionary's name and optional description (both supplied by the dictionary author) and nothing else.

7. Conclusion

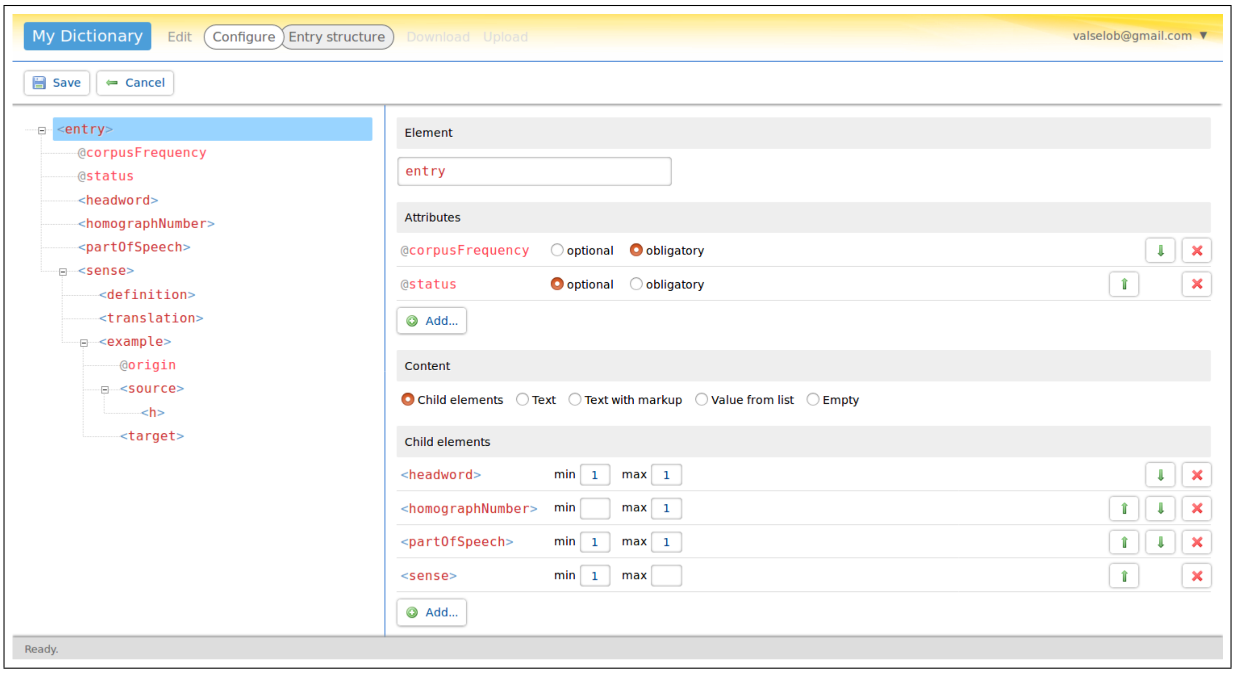
This concludes our brief introduction to Lexonomy. We have seen how Lexonomy can be used to develop a dictionary from initial set-up to final online publication. Hopefully the reader is now convinced that Lexonomy is a good home for small-to-medium sized dictionary projects. What remains is to mention a few administrative and house-keeping matters.

Lexonomy was originally created as a training tool for a lexicographic training event organized by the European Network of e-Lexicography[9] in May 2016 in Ljubljana, Slovenia. The version of Lexonomy presented here has been completely rewritten since then, contains several new or improved features, and the author believes it is fit for real-world applications.

Lexonomy is and will continue to be open-source software, licensed under the MIT Licence.[10] The source code is hosted in Lexonomy's GitHub repository [11]. Teams who do not want to use Lexonomy's 'home' installation at `www.lexonomy.eu` can download the source code, set up a local installation on their own server and customize it to their requirements. Lexonomy is written in Node.js,[12] a technology which makes it capable of running on both Linux and Windows servers.

Lexonomy will continue to be actively developed over the next number of years, thanks partly to financial support from Lexical Computing, the makers of Sketch Engine,[13] a popular corpus query system.

[9] `http://www.elexicography.eu/`
[10] `https://opensource.org/licenses/MIT`
[11] `https://github.com/michmech/lexonomy`
[12] `https://nodejs.org/`
[13] `https://www.sketchengine.co.uk/`



```
<entry>
  @corpusFrequency
  @status
  <headword>
  <homographNumber>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <example>
      @origin
      <source>
        <h>
      <target>
```

Element

entry

Attributes

@corpusFrequency	optional	[✓ obligatory]
@status	[✓ optional]	obligatory

Add...

Content

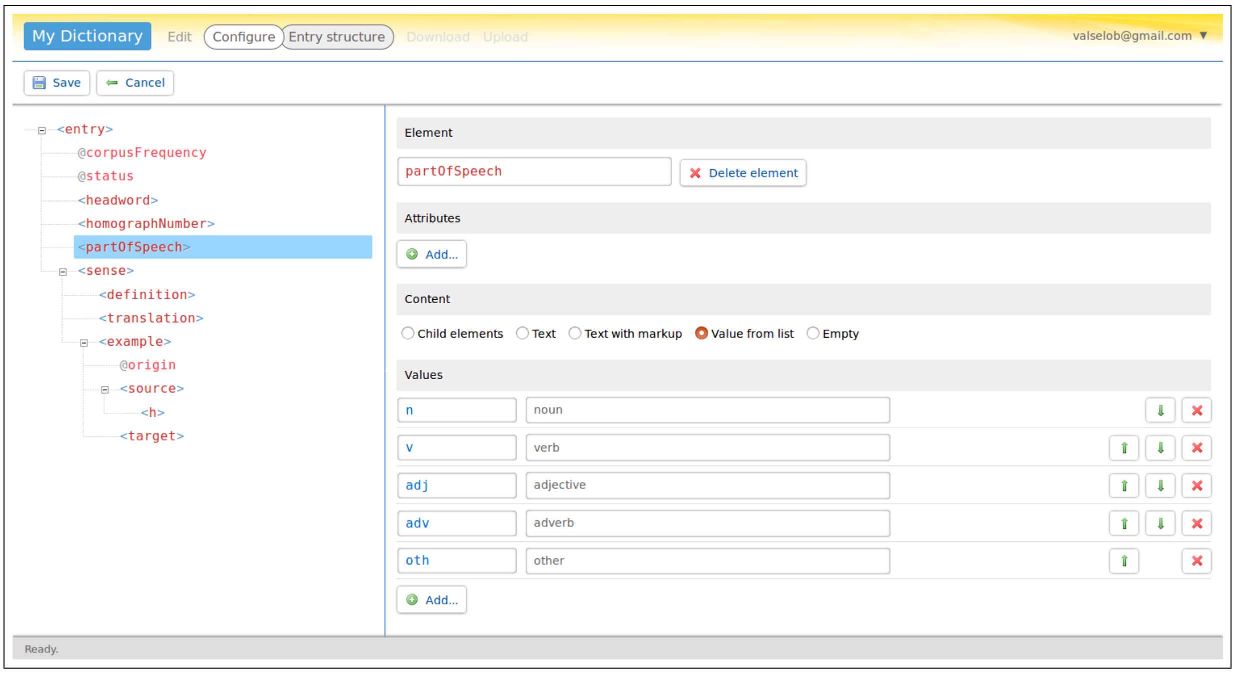
[✓ Child elements] Text Text with markup Value from list Empty

Child elements

	min	max
<headword>	1	1
<homographNumber>		1
<partOfSpeech>	1	1
<sense>	1	

Add...

Fig. 1: Entry schema editor



```
<entry>
  @corpusFrequency
  @status
  <headword>
  <homographNumber>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <example>
      @origin
      <source>
        <h>
      <target>
```

Element

partOfSpeech   Delete element

Attributes

Add...

Content

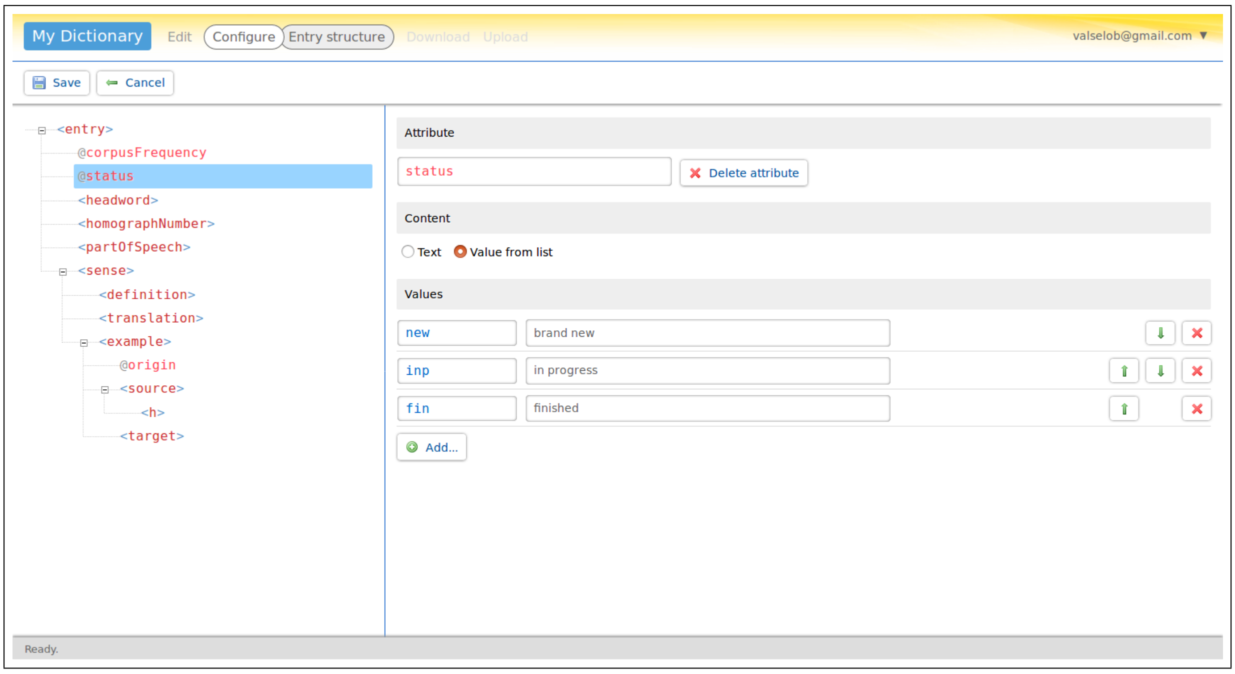
Child elements Text Text with markup [✓ Value from list] Empty

Values

n	noun
v	verb
adj	adjective
adv	adverb
oth	other

Add...

Fig. 2: Specifying the values that can appear in an element



```
<entry>
  @corpusFrequency
  @status
  <headword>
  <homographNumber>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <example>
      @origin
      <source>
        <h>
      <target>
```

Attribute

status

Content

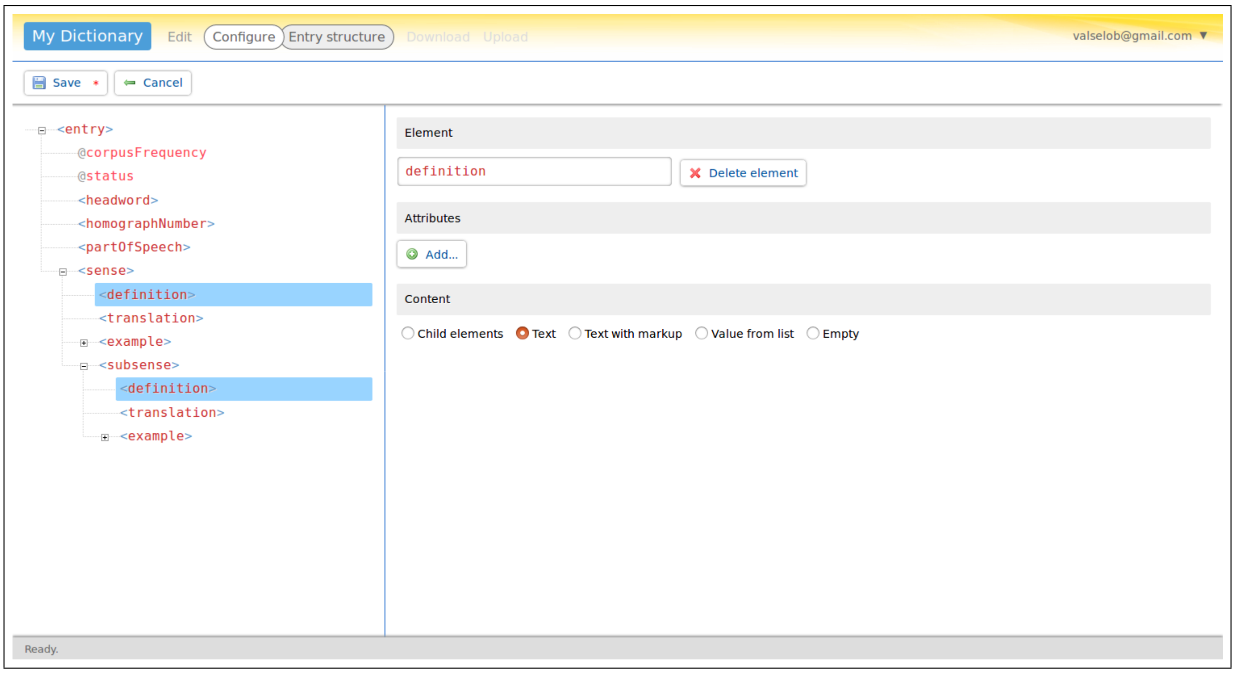
Text [✓ Value from list]

Values

new	brand new
inp	in progress
fin	finished

Ready.

Fig. 3: Specifying the content of an attribute



```
<entry>
  @corpusFrequency
  @status
  <headword>
  <homographNumber>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <example>
    <subsense>
      <definition>
      <translation>
      <example>
```

Element

definition

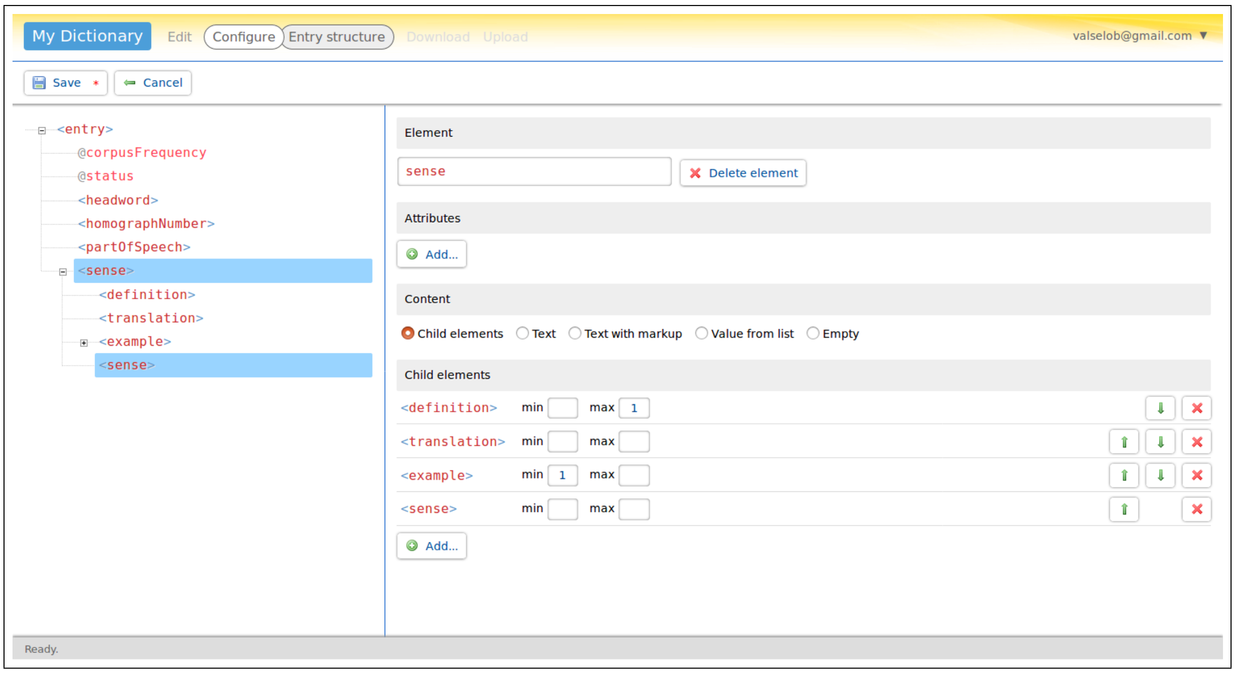
Attributes

Content

Child elements [✓ Text] Text with markup Value from list Empty

Ready.

Fig. 4: Allowing elements of a given name to appear under more than one type of parent



```
<entry>
    @corpusFrequency
    @status
    <headword>
    <homographNumber>
    <partOfSpeech>
    <sense>
        <definition>
        <translation>
        <example>
        <sense>
```

Element: sense

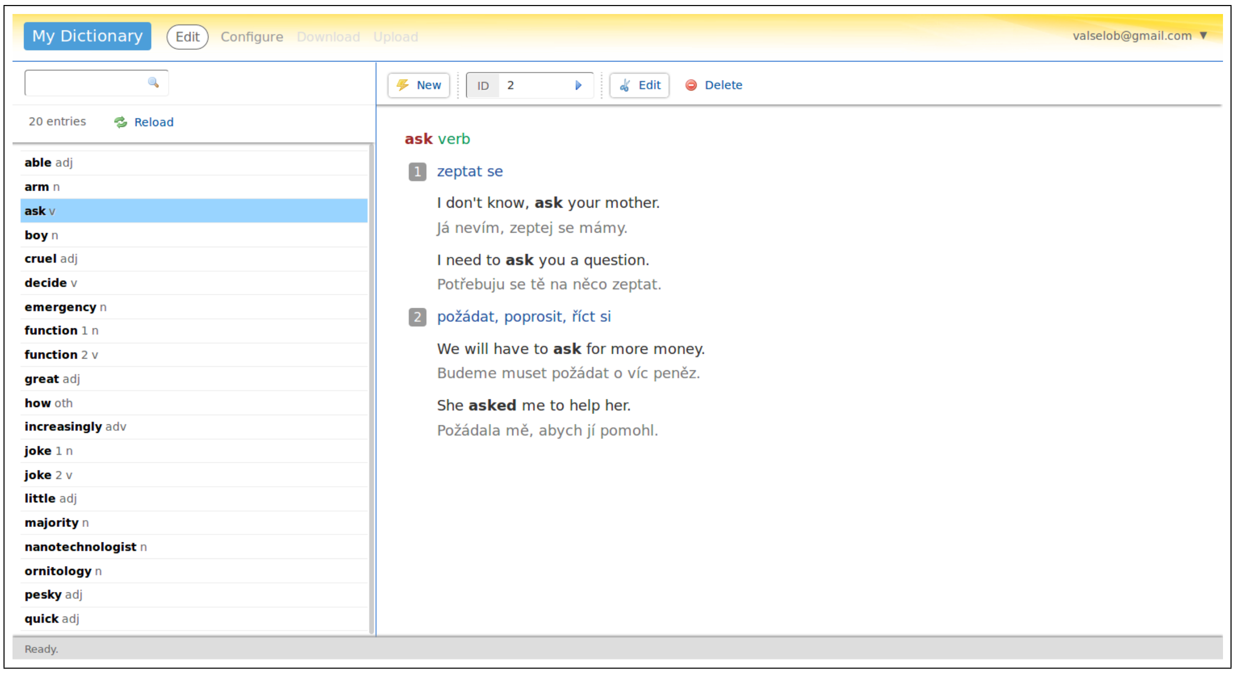
Attributes

Content: [✓ Child elements] Text Text with markup Value from list Empty

Child elements

`<definition>`	min		max	1
`<translation>`	min		max	
`<example>`	min	1	max	
`<sense>`	min		max	

Fig. 5: Recursive element nesting



20 entries

able	adj
arm	n
ask	v
boy	n
cruel	adj
decide	v
emergency	n
function 1	n
function 2	v
great	adj
how	oth
increasingly	adv
joke 1	n
joke 2	v
little	adj
majority	n
nanotechnologist	n
ornitology	n
pesky	adj
quick	adj

ask verb

1 zeptat se

I don't know, **ask** your mother.
Já nevím, zeptej se mámy.

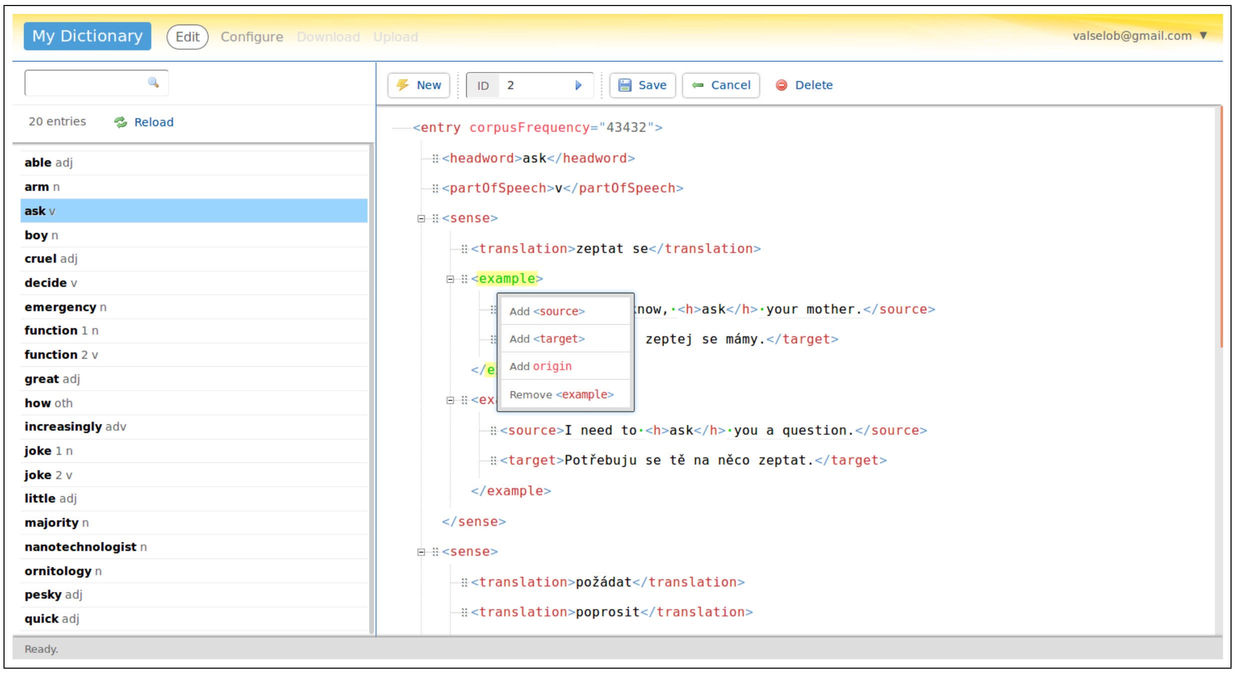
I need to **ask** you a question.
Potřebuju se tě na něco zeptat.

2 požádat, poprosit, říct si

We will have to **ask** for more money.
Budeme muset požádat o víc peněz.

She **asked** me to help her.
Požádala mě, abych jí pomohl.

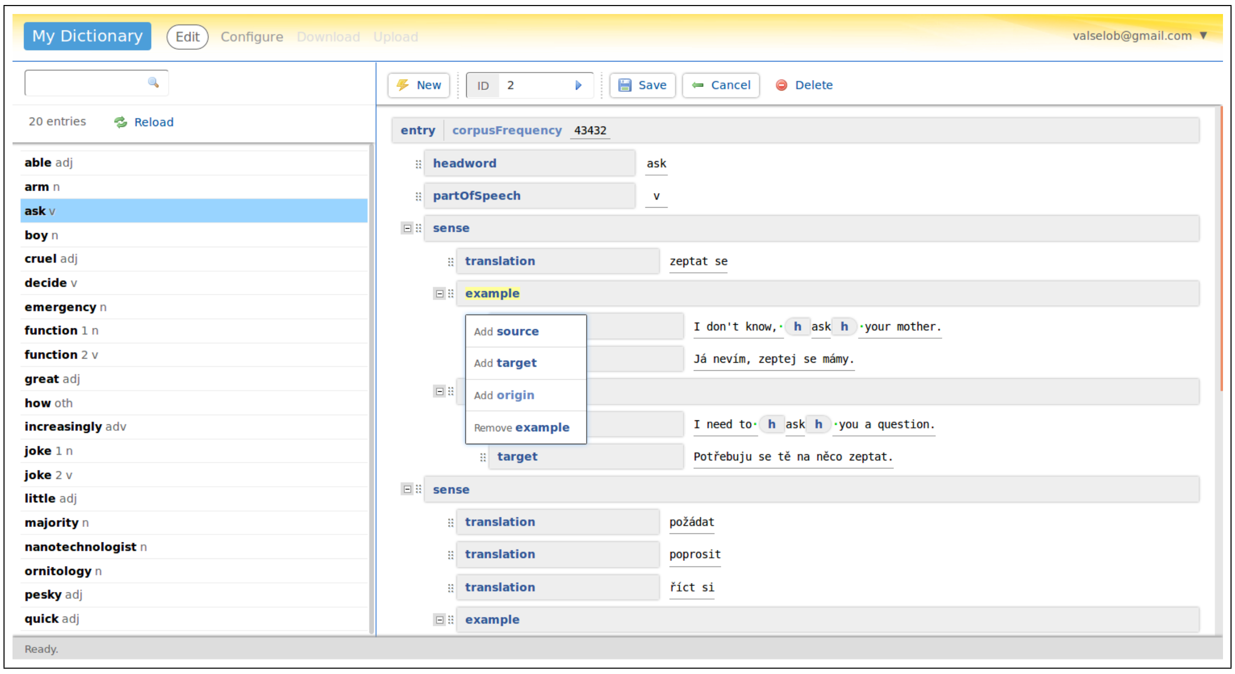
Fig. 6: Browsing and viewing a dictionary



able adj
arm n
ask v
boy n
cruel adj
decide v
emergency n
function 1 n
function 2 v
great adj
how oth
increasingly adv
joke 1 n
joke 2 v
little adj
majority n
nanotechnologist n
ornitology n
pesky adj
quick adj

```
<entry corpusFrequency="43432">
  <headword>ask</headword>
  <partOfSpeech>v</partOfSpeech>
  <sense>
    <translation>zeptat se</translation>
    <example>
                  now,·<h>ask</h>·your mother.</source>
                   zeptej se mámy.</target>
    </e
    <ex
      <source>I need to·<h>ask</h>·you a question.</source>
      <target>Potřebuju se tě na něco zeptat.</target>
    </example>
  </sense>
  <sense>
    <translation>požádat</translation>
    <translation>poprosit</translation>
```

Fig. 7: Entry editor

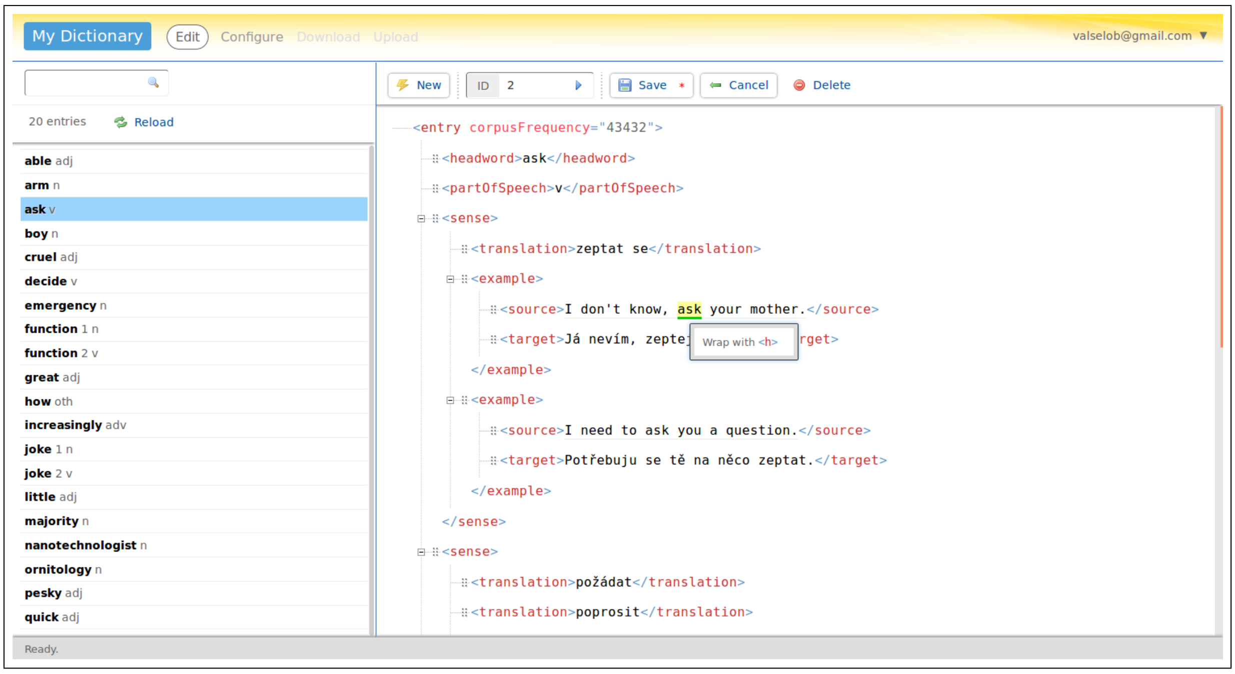


able adj
arm n
ask v
boy n
cruel adj
decide v
emergency n
function 1 n
function 2 v
great adj
how oth
increasingly adv
joke 1 n
joke 2 v
little adj
majority n
nanotechnologist n
ornitology n
pesky adj
quick adj

```
entry  corpusFrequency  43432
  headword          ask
  partOfSpeech      v
  sense
    translation       zeptat se
    example
                        I don't know,· h ask h ·your mother.
                        Já nevím, zeptej se mámy.

                        I need to· h ask h ·you a question.
      target            Potřebuju se tě na něco zeptat.
  sense
    translation       požádat
    translation       poprosit
    translation       říct si
    example
```

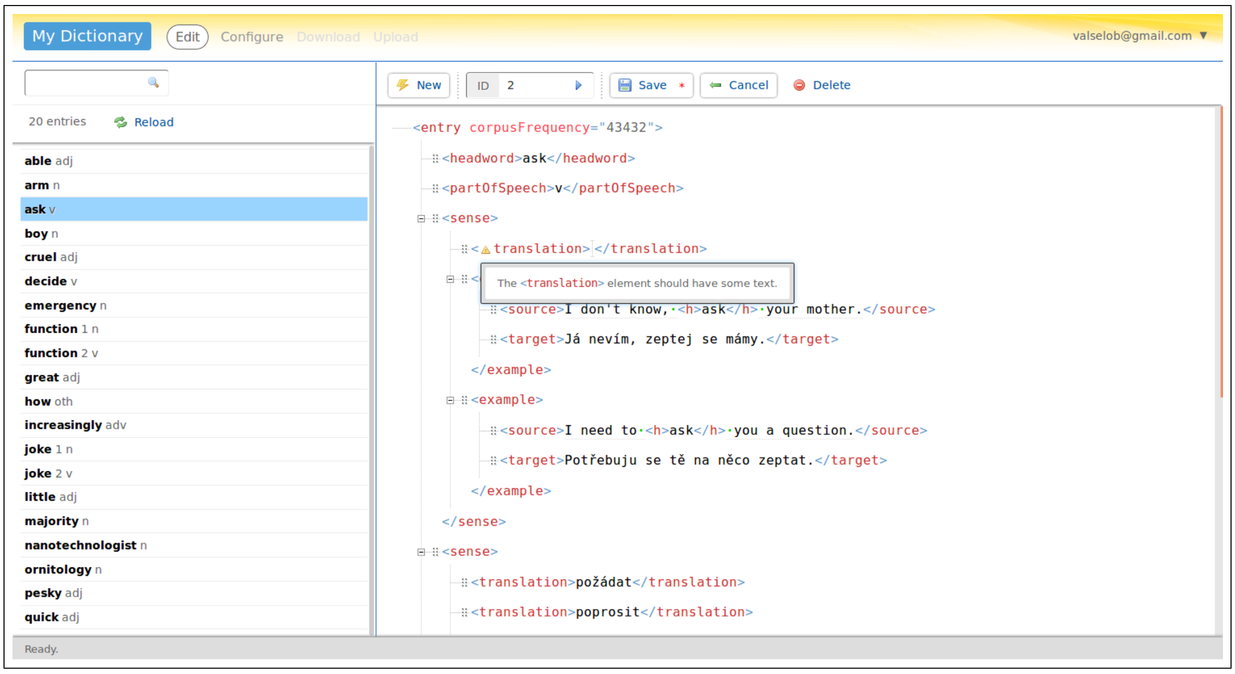
Fig. 8: Entry editor in laic mode



able adj
arm n
ask v
boy n
cruel adj
decide v
emergency n
function 1 n
function 2 v
great adj
how oth
increasingly adv
joke 1 n
joke 2 v
little adj
majority n
nanotechnologist n
ornitology n
pesky adj
quick adj

```
<entry corpusFrequency="43432">
  <headword>ask</headword>
  <partOfSpeech>v</partOfSpeech>
  <sense>
    <translation>zeptat se</translation>
    <example>
      <source>I don't know, ask your mother.</source>
      <target>Já nevím, zeptej      rget>
    </example>
    <example>
      <source>I need to ask you a question.</source>
      <target>Potřebuju se tě na něco zeptat.</target>
    </example>
  </sense>
  <sense>
    <translation>požádat</translation>
    <translation>poprosit</translation>
```

Fig. 9: Inserting inline markup



able adj
arm n
ask v
boy n
cruel adj
decide v
emergency n
function 1 n
function 2 v
great adj
how oth
increasingly adv
joke 1 n
joke 2 v
little adj
majority n
nanotechnologist n
ornitology n
pesky adj
quick adj

```
<entry corpusFrequency="43432">
  <headword>ask</headword>
  <partOfSpeech>v</partOfSpeech>
  <sense>
    <⚠translation></translation>
    <
      <source>I don't know, <h>ask</h> your mother.</source>
      <target>Já nevím, zeptej se mámy.</target>
    </example>
    <example>
      <source>I need to <h>ask</h> you a question.</source>
      <target>Potřebuju se tě na něco zeptat.</target>
    </example>
  </sense>
  <sense>
    <translation>požádat</translation>
    <translation>poprosit</translation>
```

The <translation> element should have some text.

Fig. 10: XML validation

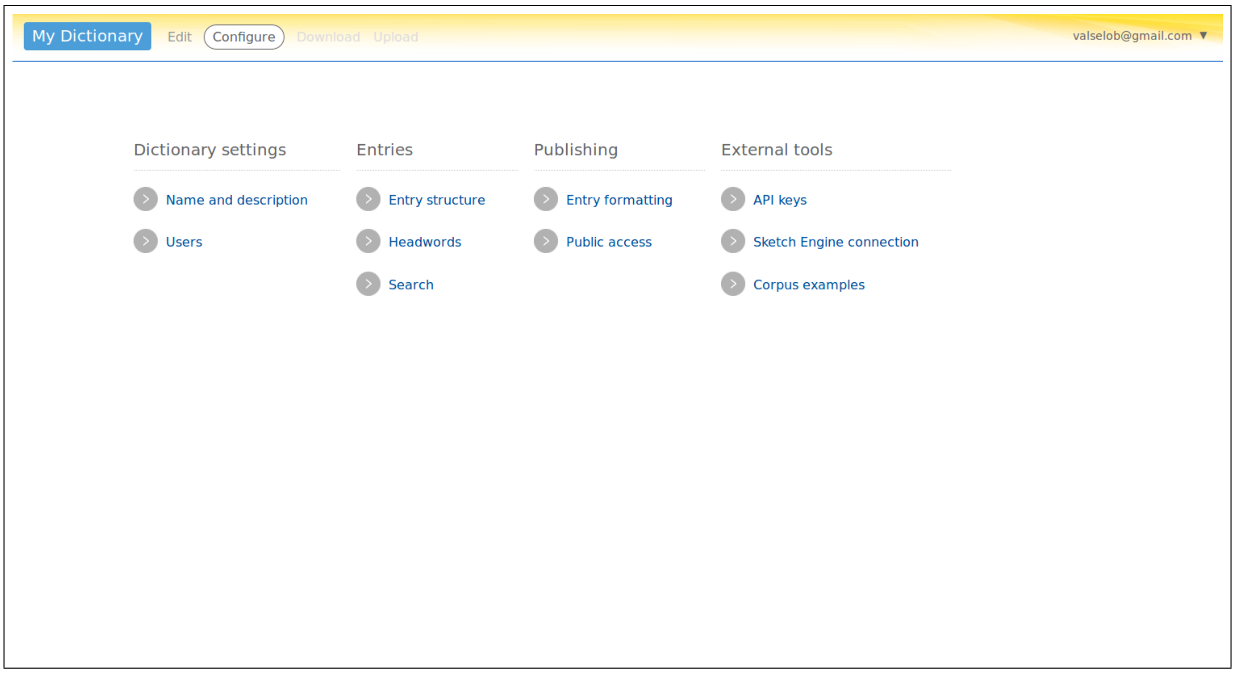
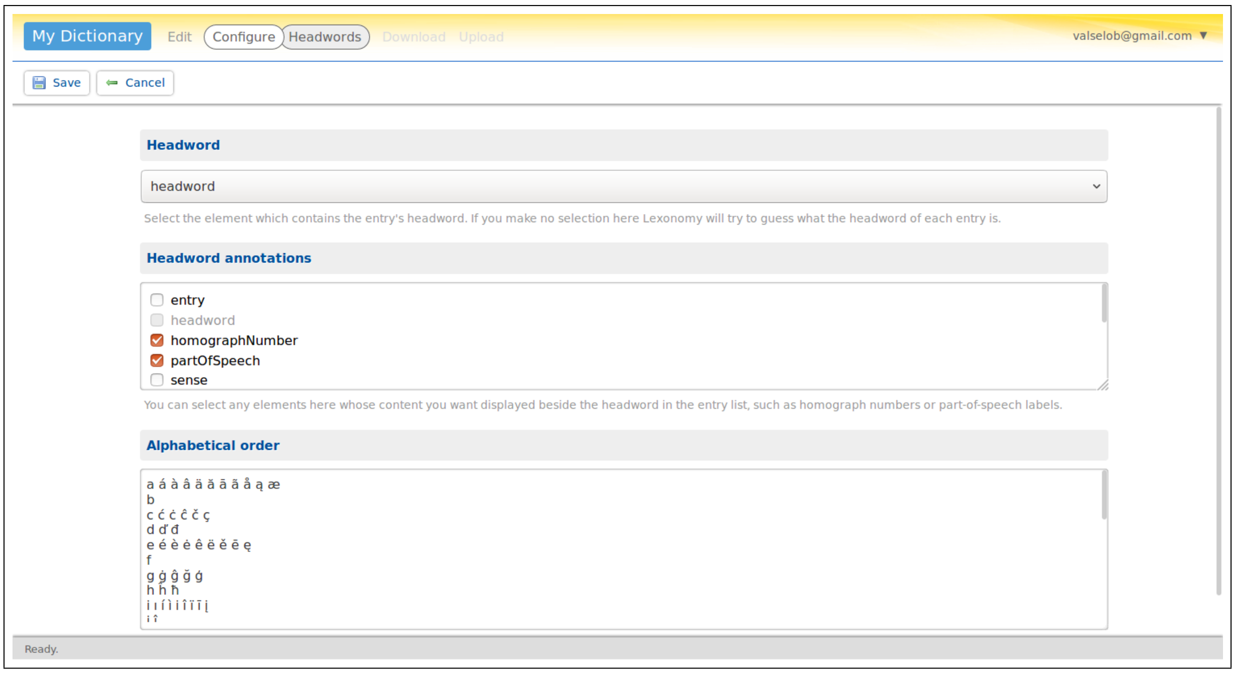


Fig. 11: Configuration screen



Headword

headword

Select the element which contains the entry's headword. If you make no selection here Lexonomy will try to guess what the headword of each entry is.

Headword annotations

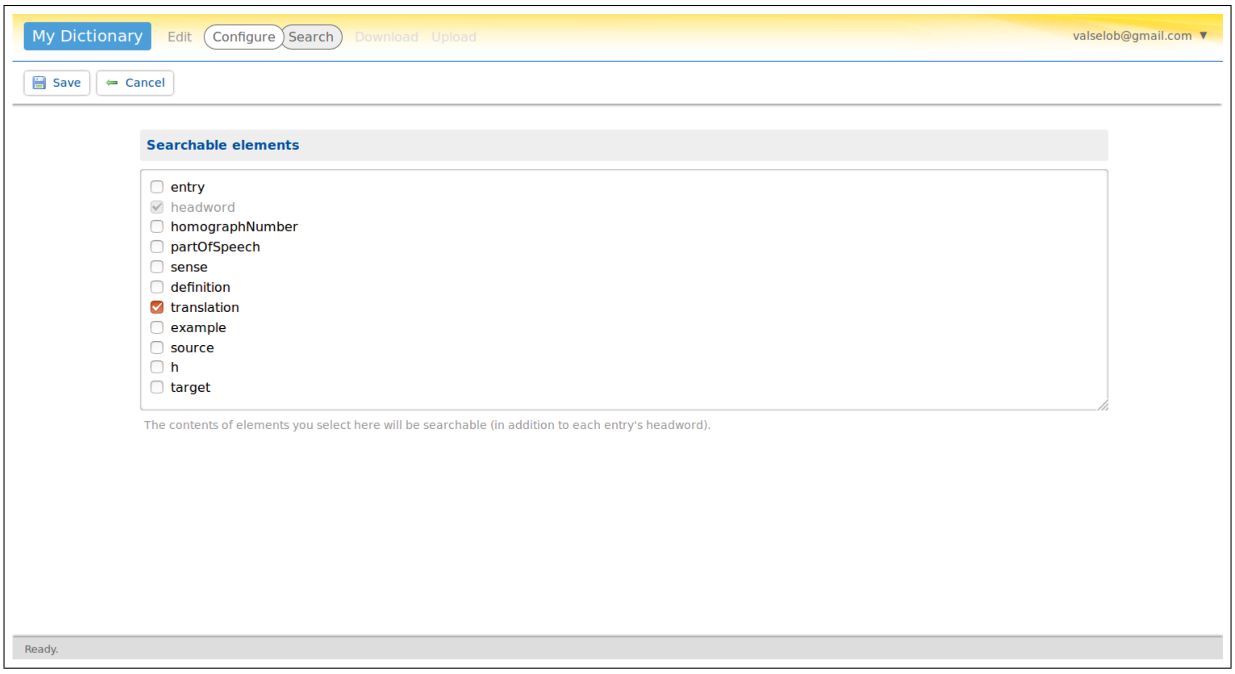
- [ ] entry
- [ ] headword
- [x] homographNumber
- [x] partOfSpeech
- [ ] sense

You can select any elements here whose content you want displayed beside the headword in the entry list, such as homograph numbers or part-of-speech labels.

Alphabetical order

```
a á à â ä ă ā ã å ą æ
b
c ć ċ ĉ č ç
d ď đ
e é è ė ê ë ě ē ę
f
g ġ ĝ ğ ģ
h ĥ ħ
i ı í ì i î ï ī į
j ĵ
```

Fig. 12: Headwords area of the configuration screen

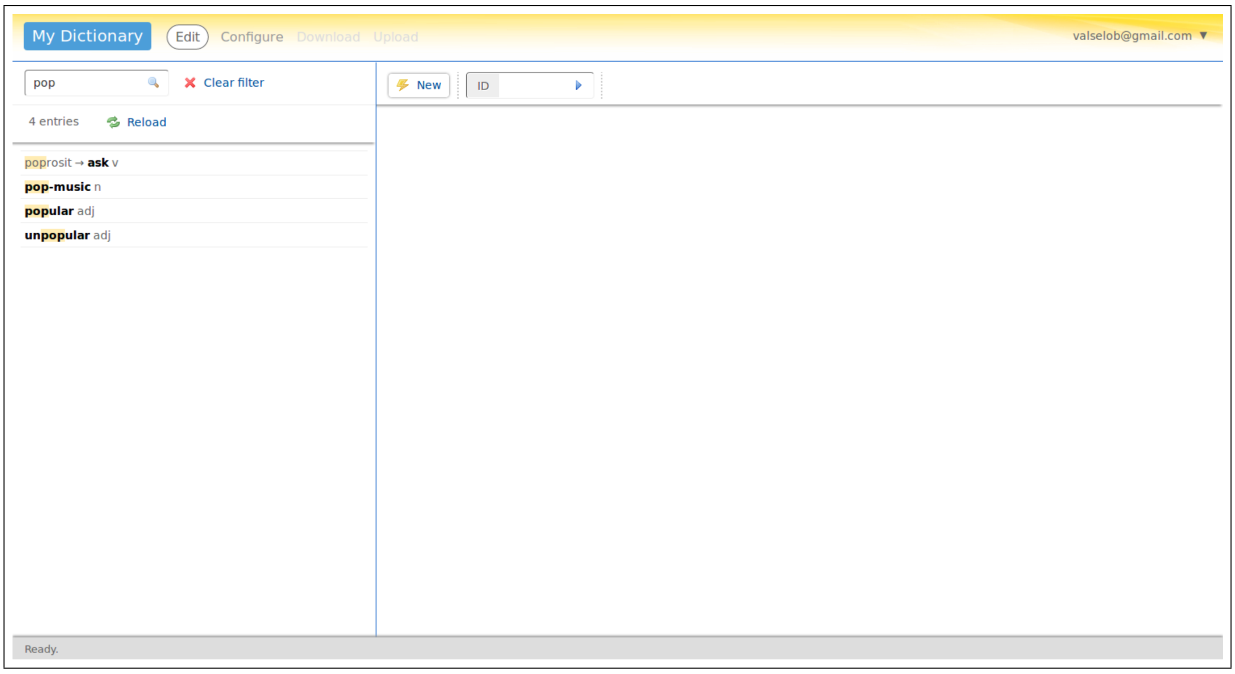


Searchable elements

- ☐ entry
- ✓ headword
- ☐ homographNumber
- ☐ partOfSpeech
- ☐ sense
- ☐ definition
- ✓ translation
- ☐ example
- ☐ source
- ☐ h
- ☐ target

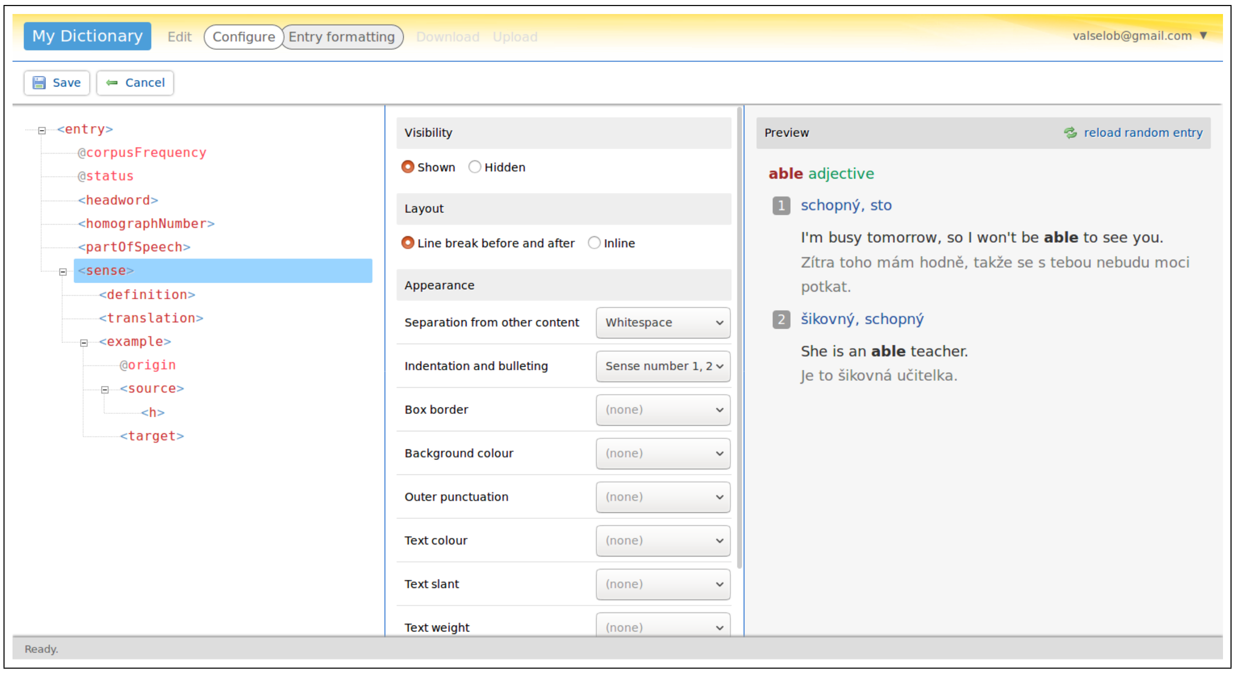
The contents of elements you select here will be searchable (in addition to each entry's headword).

Fig. 13: Search area of the configuration screen



poprosit → **ask** v
pop-music n
popular adj
unpopular adj

Fig. 14: Example of search results



```
<entry>
  @corpusFrequency
  @status
  <headword>
  <homographNumber>
  <partOfSpeech>
  <sense>
    <definition>
    <translation>
    <example>
      @origin
      <source>
        <h>
      <target>
```

Visibility

[✓ Shown] Hidden

Layout

[✓ Line break before and after] Inline

Appearance

Separation from other content	Whitespace
Indentation and bulleting	Sense number 1, 2
Box border	(none)
Background colour	(none)
Outer punctuation	(none)
Text colour	(none)
Text slant	(none)
Text weight	(none)

Preview — reload random entry

able adjective

1 schopný, sto

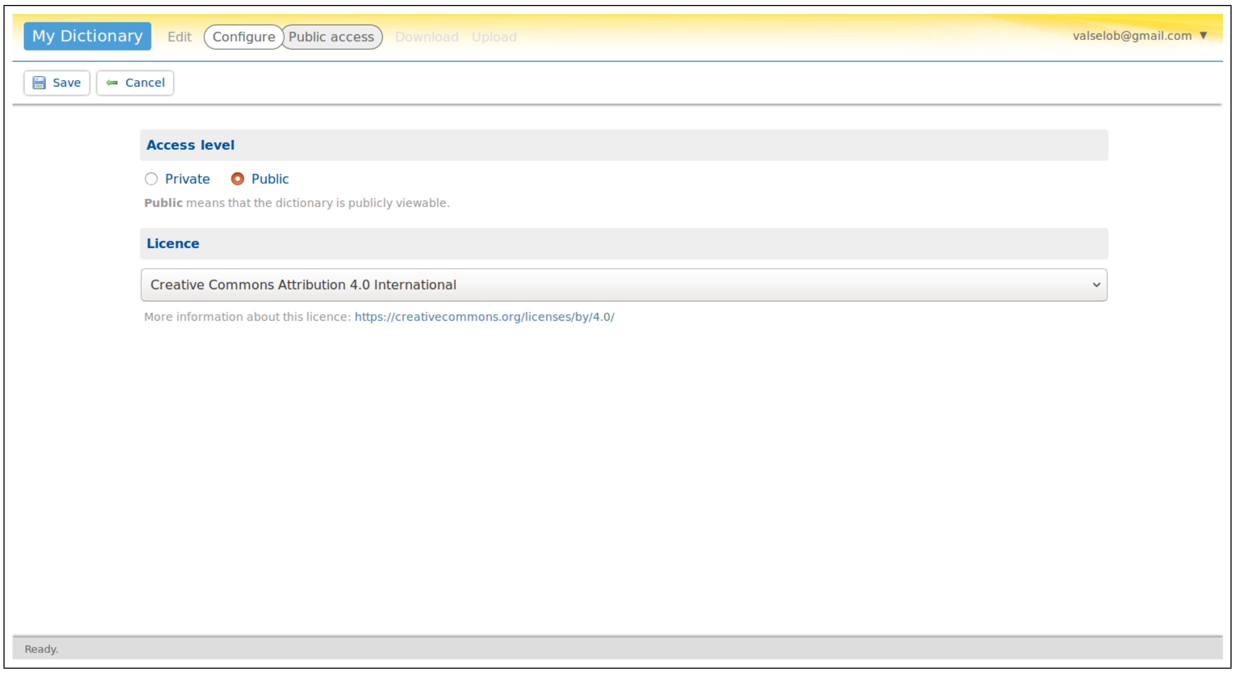
I'm busy tomorrow, so I won't be **able** to see you.
Zítra toho mám hodně, takže se s tebou nebudu moci potkat.

2 šikovný, schopný

She is an **able** teacher.
Je to šikovná učitelka.

Ready.

Fig. 15: Formatting designer



Access level

Private [✓ Public]

Public means that the dictionary is publicly viewable.

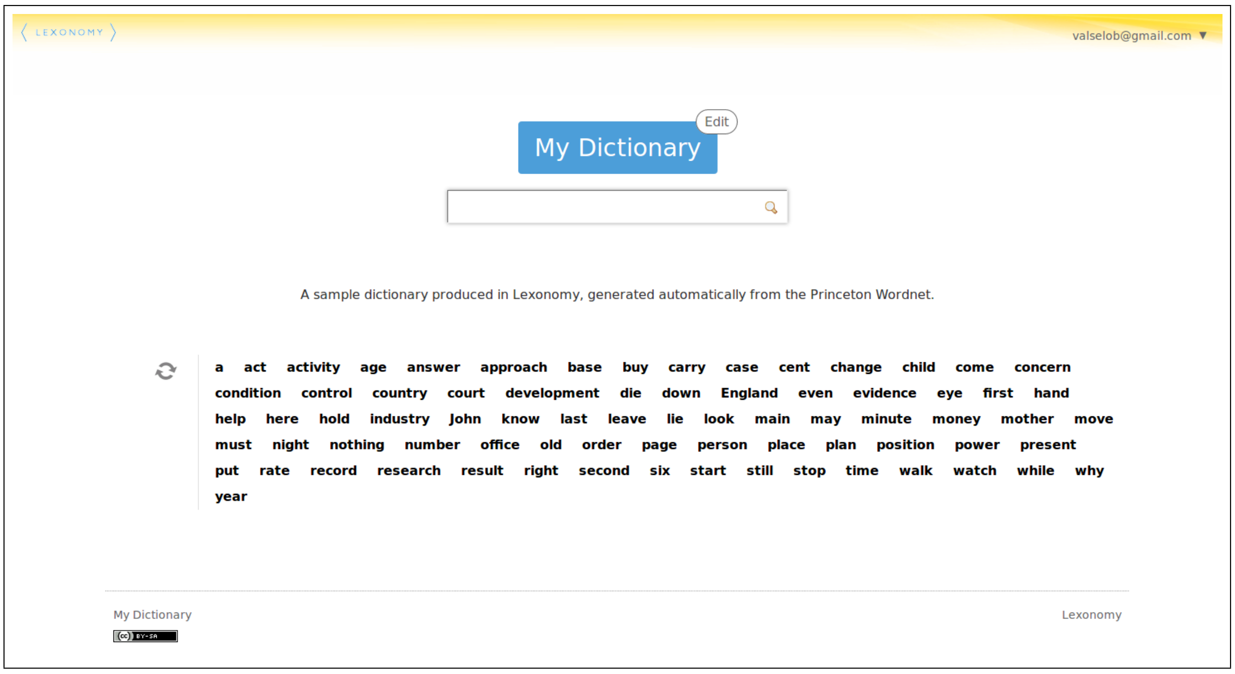
Licence

Creative Commons Attribution 4.0 International

More information about this licence: https://creativecommons.org/licenses/by/4.0/

Ready.

Fig. 16: Public access settings


〈 LEXONOMY 〉 valselob@gmail.com ▼

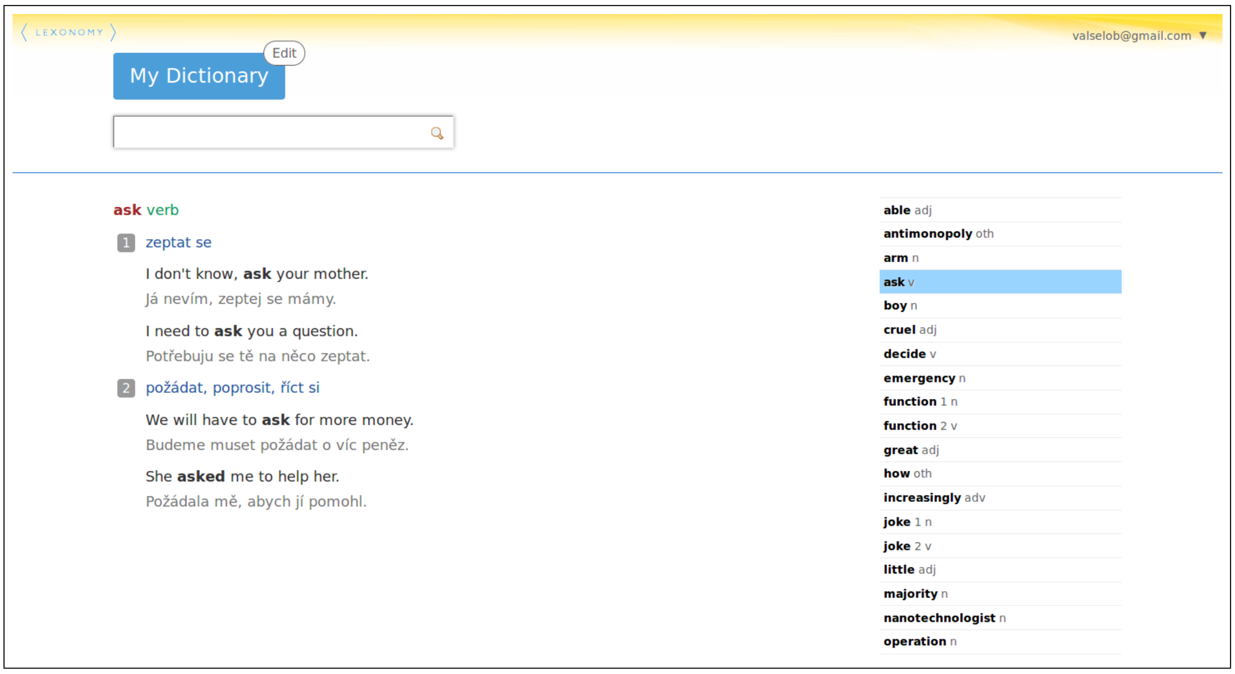
My Dictionary Edit

A sample dictionary produced in Lexonomy, generated automatically from the Princeton Wordnet.

a act activity age answer approach base buy carry case cent change child come concern condition control country court development die down England even evidence eye first hand help here hold industry John know last leave lie look main may minute money mother move must night nothing number office old order page person place plan position power present put rate record research result right second six start still stop time walk watch while why year

My Dictionary Lexonomy

Fig. 17: The public homepage of a dictionary


〈 LEXONOMY 〉 valselob@gmail.com ▼

My Dictionary Edit

ask verb

1 zeptat se

I don't know, **ask** your mother.
Já nevím, zeptej se mámy.

I need to **ask** you a question.
Potřebuju se tě na něco zeptat.

2 požádat, poprosit, říct si

We will have to **ask** for more money.
Budeme muset požádat o víc peněz.

She **asked** me to help her.
Požádala mě, abych jí pomohl.

able adj
antimonopoly oth
arm n
ask v
boy n
cruel adj
decide v
emergency n
function 1 n
function 2 v
great adj
how oth
increasingly adv
joke 1 n
joke 2 v
little adj
majority n
nanotechnologist n
operation n

Fig. 18: The public page of a dictionary entry