LX WordNet Browser Documentation

Release 1.0

Ruben Branco, NLX-Group, Faculty of Sciences, University of Lisb

Jul 03, 2018

Contents

LX WordNet Browser is a wordnet web browser, permitting the consultation of the contents of any wordnet that follows the Princeton wordnet format. The browser was designed with the functionalities and interactions necessary for a new notion of pluricentric global wordnet.

Two versions are distributed that aim to achieve different tasks. One makes it possible to consult wordnet contents of a wordnet of choice, as long as it follows the Princeton wordnet format. At any point of the search, the concepts that the user finds can be directly translated into other languages, if such concept is available. The minimalist user interface can be rendered in multiple languages. At this moment, there are two languages supported: Portuguese and English. With the efforts of the community, we wish to expand this offer to many more languages, if you are interested in helping translate the interface, read more about it here.

The other version of distribution was developed under the pluricentric global wordnet concept. It features all of the functionalities described above with a search function that is refined for multi-wordnet search. Instead of the search scope being limited to a single wordnet, the user can search for the concept in any language and have it being displayed. The browser will identify which language - wordnet the word belongs to, solving conflicts if necessary, and display the results.

CHAPTER 1

Feature Recap

- Single and Multi-Wordnet simultaneous search
- Lemma relations can be selected and explored.
- Multilingual user interface
- Translations into selected languages
- Web-based platform independent

LX-WordNet Browser Repository

1.1 Installation

The first step is to download the .zip file of the LX-WordNetBrowser and unpack it at your desired location. **Ignore the folder docs, you won't need it and can delete it.**

1.1.1 Requirements

To install and run this browser you need to have Python 3.6.3 installed (or the most recently available).

To install the required dependencies cd to the folder extracted from the .zip file you downloaded and run the following command: `pip3 install -r requirements.txt`.

With all the dependencies installed, you can start to configure the wordnet browser.

CD to the folder where your project is, and execute the command `django-admin.py startproject webapp`.

Copy the folders on the downloaded code (not requirements.txt) into the folder your project is on.

1.1.2 Configuration

General File Config

On settings.py under the webapp folder, insert the following:

- In INSTALLED_APPS list, add 'search' to the list
- At the end, paste the following code:

```
STATICFILES_DIRS = [
os.path.join(BASE_DIR, "static"),
]

STATIC_URL = '/static/'
```

substitute STATICFILES_DIRS if it's there already.

- **In TEMPLATES, you will find 'DIRS' likely empty, substitute it with the following:**
  [os.path.join(BASE_DIR, 'templates')]
- In ALLOWED_HOSTS, insert the IP you're going to use for your website.
- In webapp/urls.py, paste the following code:

```
from django.conf.urls import include, url
from django.contrib import admin
from search import urls

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^', include(urls)),

]
```

In /static/index.js search for "location.port", if you are using a domain you will likely not need the port so delete the ':' + location.port occurances and leave just the location.hostname. Still in index.js, for the pluricentric installation, you will want to edit the const "nodeText" to fit your institution name.

In langdata/main you should be putting your wordnet files in, along with the respective bridge between the pivot language.

Back to /static/views.py, search for 'display limit', which will be a comment section. There you will need to decide whether you will limit the results to be shown and the number of depth you will limit it by. If not, then you will set depth to None.

IMPORTANT Read the *Files* section in this document (*Developer Documentation - General*) of developer documentation to understand how the file locations and format works.

ALSO IMPORTANT At the end of these configurations run the command `python3 manage.py migrate` to migrate changes.

WordNet Content Delivery Server

To avoid opening and reading wordnet files everytime there's a request, we'll setup a WordNet content delivery server. The code for this server is contained in 'wordnet_server.py'.

Give run permissions to wordnet_server.py if on a unix system using `chmod u+x wordnet_server.py`. Here's the command line syntax: `wordnet_server.py [PORT] [BROWSER_PATH] [BROWSER_TYPE]`.

PORT is the port you'll be using for **Wordnet CDN Server, do not use the port you're using for the wordnet browser**.

BROWSER_PATH is the path to the project folder where you have your browser. If you have the wordnet_server.py file in it you can just use '.'.

BROWSER_TYPE has two options: pluricentric or basic. These referring to the two types of installations supplied. If you want to use this server to serve a pluricentric web browser, use pluricentric. If it's supplying a 'my_wordnet' type of installation, use basic.

Once you have configured your server, make sure to edit the file `/search/views.py` corresponding to your project to be able to access the right port. By searching 'wordnet_server', you will find the inicialization of the server. The port side of URL is empty so you can edit and put in the port you have chosen.

The configuration currently uses localhost as the IP of the CDN server. If you wish to have an external server as a host, you will have to edit the code in order to use the right IP and port.

If you have the two installations running you will have to have seperate CDN servers for each installation by using different ports and editing the views.py code to match them.

1.1.3 Apache Configuration

We are going to use Apache on the front end of the server that is going to act as a reverse proxy to a WSGI server running on gUnicorn.

gUnicorn Installation and Setup

To install gUnicorn simply run the command `pip3 install gunicorn`.

Setup/run is extremely easy, just run the command `gunicorn projectname.wsgi` in your project folder and it will start up the server. The server is started on 127.0.0.1:8000.

More info here: https://docs.djangoproject.com/ko/1.11/howto/deployment/wsgi/gunicorn/

Apache Reverse Proxy

If you don't have apache2 on your machine, you can either scour through the web for the installer if you're on a Windows machine or if you're on ubuntu you can most likely install it through `apt-get install apache2`.

Create a configuration file on sites-available, on ubuntu it's on `/etc/apache2/sites-available`, the content should be the following:

```
<VirtualHost *:80>

        ServerAdmin email

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        ProxyPass /static/ !
        ProxyPass / http://localhost:8000/

        Alias /static/ COMPLETE_PATH_TO_YOUR_PROJECT_STATIC_FOLDER

        <Directory COMPLETE_PATH_TO_YOUR_PROJECT_STATIC_FOLDER>
                Options Indexes FollowSymLinks
```

(continues on next page)

(continued from previous page)

```
                AllowOverride None
                Require all granted
        </Directory>
</VirtualHost>
```

After creating your config file run the following commands:

`sudo a2dissite 000-default` and `sudo a2ensite [config_file_name]` name without extension. Then to restart apache2 run `sudo apache2ctl restart`.

And it should be serving everything correctly.

Run

To run on a development environment, cd int the project directory and run the following command `python manage.py runserver IP:PORT`.

To run on a production environment, refer to the explanation above. The website will be reachable on your IP port 80.

1.2 Translations, a commmunity helping hand

In our project, we want to ensure that the interface menus can be displayed in multiple languages. For that, we need the community's help.

Here is the current list of languages that need help translating, if you're fluent in any of them, consider reading further to learn how you can help us, it takes less than 10 minutes!

Languages in need of translation:

- Finnish
- Afrikaans
- Arabic
- Asturian
- Azerbaijani
- Belarusian
- Bengali
- Breton
- Bulgarian
- Catalan
- Czech
- Simplified Chinese
- Welsh
- Danish
- German
- Greek
- Esperanto

- Estonian
- Basque
- Faroese
- Farsi
- French
- Scottish Gaelic
- Irish
- Galician
- Serbo-Croatian
- Hebrew
- Hindi
- Hungarian
- Indonesian
- Icelandic
- Italian
- Japanese
- Georgian
- Korean
- Latin
- Latvian
- Lithuanian
- Macedonian
- Dutch
- Nynorsk
- Bokmål
- Polish
- Romanian
- Russian
- Slovak
- Slovene
- Spanish
- Swahili
- Swedish
- Telugu
- Thai
- Turkish

- Ukraine
- Urdu
- Vietnamese
- Volapük
- Malaysian

1.2.1 What to translate & how to help

If you are interested in helping, here's how you can do so.

The list of translations are the following, all in English:

General Menu Interactions

- related concepts
- translations
- direct | transitive hypernyms
- direct | transitive hyponyms
- direct | transitive troponyms
- member holonyms
- substance holonyms
- part holonyms
- member meronyms
- substance meronyms
- part meronym
- antonyms
- instance hypernyms
- instance hyponyms
- attribute
- derivationally related forms
- entailment
- cause
- also see
- verb group
- similar to
- participle of verb
- pertainym
- derived from adjective
- domain category

- domain term category
- domain region
- domain term region
- domain usage
- domain term usage
- This synset doesn't have any registered relations with any others
- Related to
- concept
- overview
- sentence frames
- The search couldn't find the word you were looking for
- definition
- certainty
- translations to:
- search for a word
- search
- change the display language
- no languages selected
- The word you are searching for appears in more than one language. Choose which one you would like to explore in the dropdown menu below.
- Select a language

Languages

- English
- Finnish
- Afrikaans
- Arabic
- Asturian
- Azerbaijani
- Belarusian
- Bengali
- Breton
- Bulgarian
- Catalan
- Czech
- Chinese

- Welsh
- Danish
- German
- Greek
- Esperanto
- Estonian
- Basque
- Faroese
- Farsi
- French
- Scottish Gaelic
- Irish
- Galician
- Serbo-Croatian
- Hebrew
- Hindi
- Hungarian
- Indonesian
- Icelandic
- Italian
- Japanese
- Georgian
- Korean
- Latin
- Latvian
- Lithuanian
- Macedonian
- Dutch
- Nynorsk
- Bokmål
- Polish
- Portuguese
- Romanian
- Russian
- Slovak
- Slovene

- Spanish
- Swahili
- Swedish
- Telugu
- Thai
- Turkish
- Ukranian
- Urdu
- Vietnamese
- Volapük
- Malaysian

Part of Speech

- Noun
- Verb
- Adjective
- Adverb

1.2.2 How to submit your translations

You can submit your translations by clicking here and submitting an issue with the title [Language] Translations and label Translation.

It's best to submit with the English sentence on the left side and then a seperator of your wish and then the other language conterpart on the right side.

Thank you for your help!

1.3 User Documentation

1.3.1 Interface Language

You can change the interface language on the top side of the browser, in the menu like the one in this image. The first four languages are clickable, upon click the page will reload and the language will be displayed. On the '+' icon you will find a dropdown menu containing more languages that are browsable.

1.3.2 References & Download

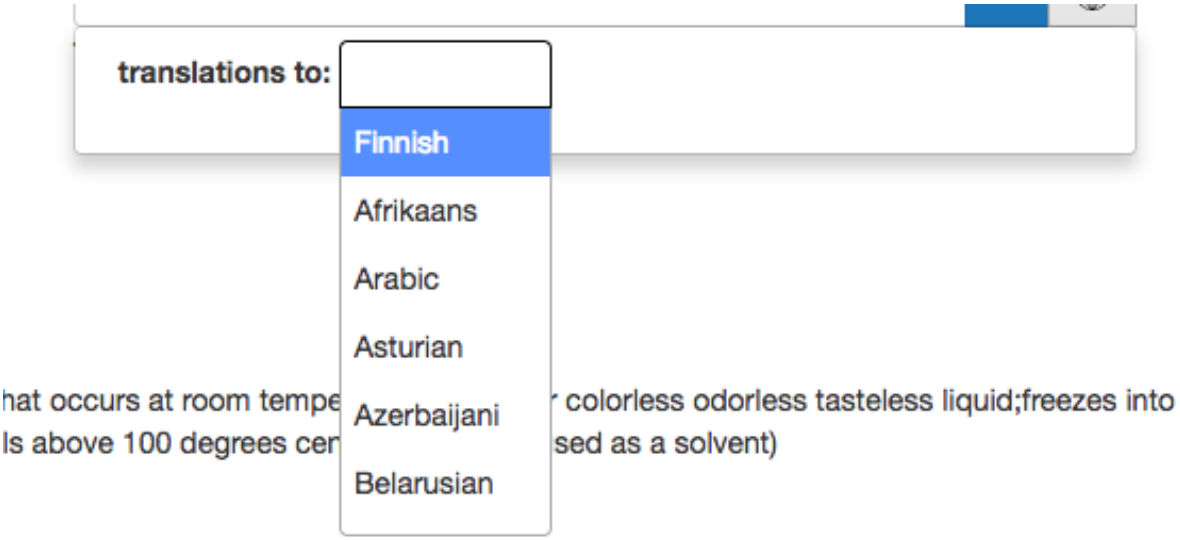
References to the resources used are on the references page which is accessed through the icon on the left.

The link to the download is on the icon on the right side.

1.3.3 Search

Search bar

By clicking enter or the magnifying glass icon the search will be triggered, given that there is a word being searched for. The icon on the right with the globe reveals the translation menu.



You can select however many languages you would like to see the translations of words on.

Related concepts

Noun

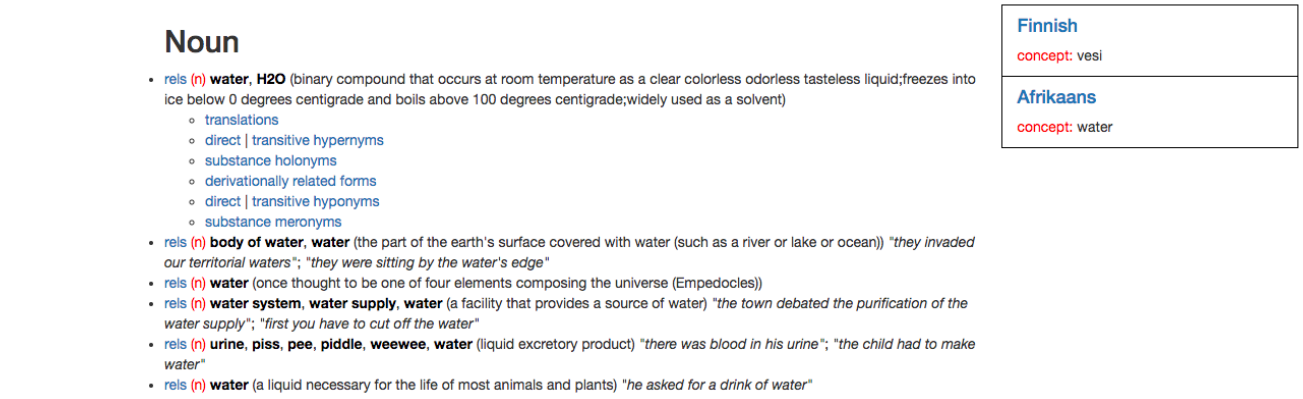
- rels (n) **water**, **H2O** (binary compound that occurs at room temperature as a clear colorless odorless tasteless liquid;freezes into ice below 0 degrees centigrade and boils above 100 degrees centigrade;widely used as a solvent)
- rels (n) **body of water**, **water** (the part of the earth's surface covered with water (such as a river or lake or ocean)) *"they invaded our territorial waters"*; *"they were sitting by the water's edge"*
- rels (n) **water** (once thought to be one of four elements composing the universe (Empedocles))
- rels (n) **water system**, **water supply**, **water** (a facility that provides a source of water) *"the town debated the purification of the water supply"*; *"first you have to cut off the water"*
- rels (n) **urine**, **piss**, **pee**, **piddle**, **weewee**, **water** (liquid excretory product) *"there was blood in his urine"*; *"the child had to make water"*
- rels (n) **water** (a liquid necessary for the life of most animals and plants) *"he asked for a drink of water"*

Any synset in the results is expandable, meaning that the relations to the synset can be displayed. By clicking on 'rels', the menu to target those relations is expanded, like so.

Noun

- rels (n) **water**, **H2O** (binary compound that occurs at room temperature as a clear colorless odorless tasteless liquid;freezes into ice below 0 degrees centigrade and boils above 100 degrees centigrade;widely used as a solvent)
  - translations
  - direct | transitive hypernyms
  - substance holonyms
  - derivationally related forms
  - direct | transitive hyponyms
  - substance meronyms

Selecting any relations will further expand search results. If you select translations, the results of the translations into the languages you have selected earlier in the translations menu will show up on the right side.



1.4 Developer Documentation - General

1.4.1 Back end

WordNet Content Delivery Server

The WordNet content delivery server is implemented using XMLRPC protocol, which despite worries about its unsafety, is a good candidate due to its simplicity of implementation and usability. It being hosted on localhost mitigates the safety worries.

As specified in the installation chapter, there's two modes to this server. It can either load and supply a pluricentric or a my wordnet type of installation. Upon launch of the server, it will call 3 loaders of its own, that load tab files, wordnet files and pair files. The difference between the modes are on the amount of wordnets it has to load and their locations, which are different from the pluricentric and my wordnet type of installations.

The structure in which the wordnet is stored is a dictionary with the following structure.

`language > pos > data & index > synset offset > whole line.

language on the my wordnet installation is main for the main language and pivot for the pivot language.

This implementation could be improved by instead of storing a whole line, storing a dictionary for each offset with the relations present and with what offsets, names, and other info that is further used in searches.

For vrb files, the structure is the following:

language > vrb > verb OR number of sent/frame > line.

For the tab files, it follows the following structure:

language code > pivot language synset offset > line.

Other methods, such as get_index and get_data are implemented for the main script /search/views.py fetch information needed relevant to the users search.

For pair files, the structure follows the similar logic to the others above. Pair files provide a much better relation between languages, since its a relation between synsets and not lemmas.

language code > language synset offset > pivot language synset offset.

Apache

Forthcoming

Django

Django is the framework chosen to power the interface. Since there's a server to deliver wordnet content, any functionalities related to databases aren't touched. There's little need for an admin portal as well, so it's up to other developers to implement one if there's any need for them.

Views.py

Classes

Parser

Defines a class for parsers of lines. This is where the wordnet lines get converted to HTML formatted lines with all the relevant information such as names and glosses. All search functions other than translations will eventually call this class to format their lines or obtain information.

Search Routines

This class holds all the search routines that are needed for the browser to work.

Single search is used only in one special case, which is when you explore derivationally related forms and you wish to see, for a given result, what its synset looks like. In that case, **single_search** will be called to deliver just the one synset its looking for, not a lemma driven search.

full_search is the most commonly used method for post initial lemma driven searches. After the user searches for a lemma and its results appear on the screen, they may expand the relations menu, as explained in *User Documentation*, and through that navigate the various relations that the synset may have with others. That relation targeting is handled by full_search alone. It will take the synset and search, recursively, through the synsets that hold the type of relation targeted in the search. It is essentially going through the graph all the way to the bottom of each branch. In the end, an HTML list is built by constantly concatenating the information of each synset. There's the notion of max_length and max_depth. Max_length is the maximum of local(synset) relations that can be displayed, and max_depth is the depth of the graph that you can get to. These can be equal to None if you don't want to maximize the display of results, this is just a "safety" feature to avoid scraping of wordnets that may be unwanted by the holders. Throughout the search, information on what relations are present in each synset will be stored and sent over. This information will be stored in the javascript process to be faster in understanding what relations are present in each synset upon a relations menu expansion.

expand_search is just a work delegator, it identifies what the request wants and distributes it to the functions inside the search routines appropriate to the request.

sentence_frame_search searches for a certain verbs sentence frames. They may be specific or general frames.

normal_search follows a similar strategy to **full_search**, albeit being a iterative, 0 depth search. It will search for the lemma being searched for in the index file and then with the offsets now in hand, the data info will be retrieved and parsed to an html line to be delivered. The information regarding present relations are also sent over.

Renders

The class renders has the methods that render the webpage to the user, upon a visit.

Files

My WordNet

There is a slight difference between both versions of the browser in terms of where the wordnet files are stored. Both of them make use of `langdata` folder to store other folders. My WordNet uses `main` folder to store the files for the language that you want to be browsable. `pivot` folder stores the wordnet files for the language you're using as a pivot for translations. 'tab files' stores the tab files necessary for translations.

Pluricentric

The pluricentric version uses `wordnets` to store wordnet files. Inside that folder, more folders with the name of the languages that then store the files. Example: `wordnets/English/*`. Like My Wordnet, `tab files` stores tab files.

Formats

This browser makes use of wordnets in Princeton Wordnet format. There is some tendency to use LMF format and XML format as well, which isn't supported. However, we have included some scripts and documentation for LMF to Princeton conversion and XML to Princeton conversion.

Forthcoming documentation on conversion

Tab files

Forthcoming documentation on tab file problems on generation

1.4.2 Front End

language.js

Due to how little text menu oriented the interface is, instead of having dozens of repeated files, one for each language that the interface offers, we decided to use a javascript object structured file with all the text. This can be done differently by other developers who may take on this interface by having actual URLs that direct to different configurations.

The text is then loaded by language and by context. It can be a part of speech translation, or a part of the menu relating to semantic relations, etc.

index.js

The **main** as one would expect loads all the dependencies, event listeners of all sorts that are needed for the UI to work. If the user searches for a lemma, **search** is called which appends a box for the results to be displayed and issues a GET request, whose response is treated in **formattedResults**.

If the user wants to expand the related concepts menu, **expand** is called which will check what relations are present in that offset and display the possible searches. If the user then selects one of those relations, **expandedSearch** is called. This function checks if the search was already done in the past, which if that's the case, it will hide the results (the user may not be interested in that result anymore and wants to keep his window clean of it), otherwise it will issue a GET request for the search. The result from that GET request then get treated by **expandedSearchFormatter** unless its a sentence frame, whose function **sentenceFrameFormatter** does the work for.

1.5 Developer Documentation - Pluricentric WordNet

1.5.1 Back end

Views.py

The main different between pluricentric and my wordnet type of installations is the fact that you can search in *any of the implemented languages* and it will display results if such are found. It will identify the language and then display

the results.

Language identifying is done with **language_identifier**, being as simple as going through index files and seeing if the word shows up on them. Due to being a single word search, other sofisticated techniques of language identification don't work as well. If there's a collision, as in more than one languages are detected, then a message will be sent to the client for him/her to decide the language he/she wants to explore.

The advanced search for a translation can be done either through a pair file if there's one present, where the translation is then between synsets. If one isn't present, a search will be done through tab files. If there's more than one sense detected, a pairing mismatch error will happen as it is uncertain which sense of the lemma the source language synset is in the target language.

1.5.2 Front End

language.js

There's additions to language.js as more menus come up, such as the language collision pop-up.

index.js

Similar to the views.py, the only addition done is the collision treatment. If javascript receives a collision message, it will render a pop-up for the user to decide which language he/she wants to explore, that gets sent back to the server. Now that the server knows the language, the processes are the same as a my wordnet type installation.