CogniVal

https://github.com/DS3Lab/cognival-cli

CogniVal is a framework for cognitive evaluation of word embeddings.

Installation

- Set up a **new** virtual environment. Make sure to set the Python version to ≥3.7.4.
- Download v0.1.4 of CogniVal and follow the installation instructions via pip from the README:
  https://github.com/DS3Lab/cognival-cli

Example Demo: Evaluate Glove 50 embeddings with EEG data from ZuCo

After successful installation you can start the tool by typing directly in the terminal:

> *cognival*

The welcome message contains a shorter version of this demo.

The goal of this demo example is to get familiar with the CogniVal tool and the available commands. As an example, we will evaluate pre-trained Glove word embeddings of 50 dimensions against the EEG features from the ZuCo dataset.

Step 1: Configure general properties

The configuration 'demo' serves to demonstrate the functionality of this tool. In order to use it, open the general properties of the configuration by executing:

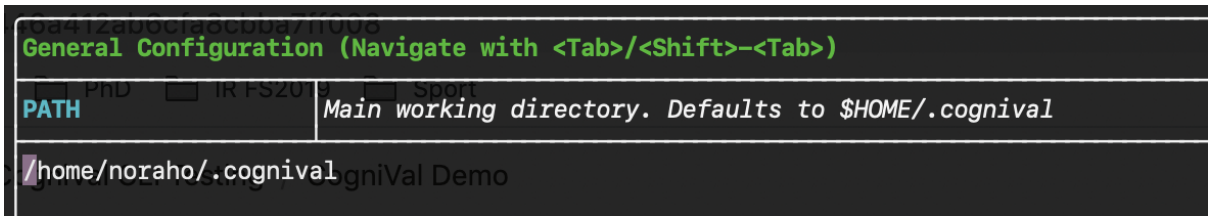
> *config open configuration=demo edit=True*

This command also sets the configuration to be the current one, applying all configuration-specific commands to it.

(If the error message "Window too small ..." appears, resize the terminal window.)

Navigate the form with (Shift-)Tab and the content of fields with the cursor keys.

Remove the placeholder for the PATH property (just leave the field empty) and it will be automatically set to the CogniVal user directory (by default: $HOME/.cognival). Example:



If necessary, adjust e.g. the number of parent process instantiated (reduce this number when experiencing problems).
Navigate to the bottom of the form and save the form.

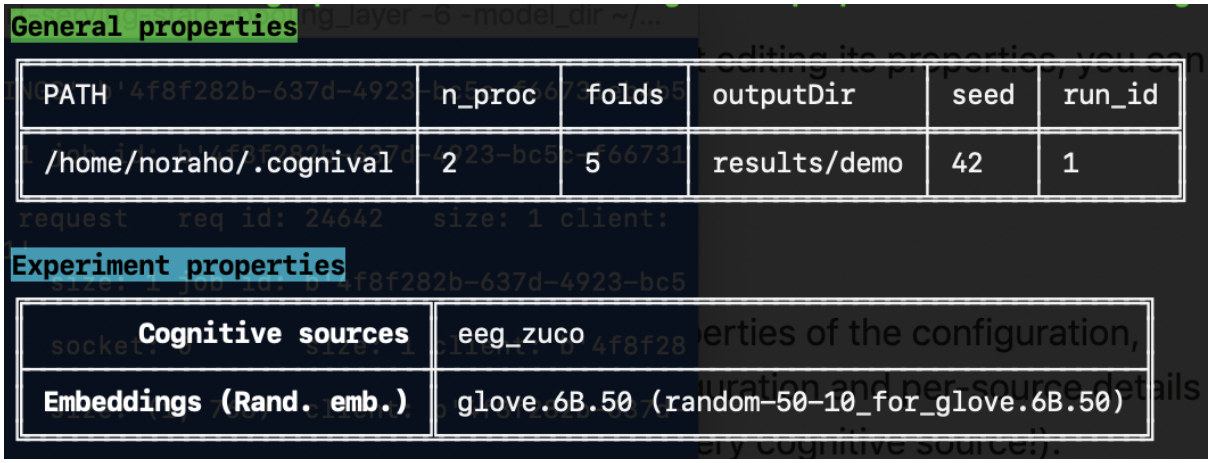
Then, in order to load the demo configuration without editing its properties, you can simply execute:

> *config open demo*

Using the following command, you can show the general properties of the configuration, cognitive sources and embeddings associated with the configuration and per-source details (Note: not every embedding is necessarily associated with every cognitive source!):

> *config show details=True*

The demo configuration contains only one cognitive source, EED data from the ZuCo corpus, and one embedding type, GloVe embeddings with dimensionality 50, associated with a set of random embeddings of matching dimensionality:



Once the configuration is loaded, we can proceed with the evaluation.

Step 2: Download cognitive sources and word embeddings

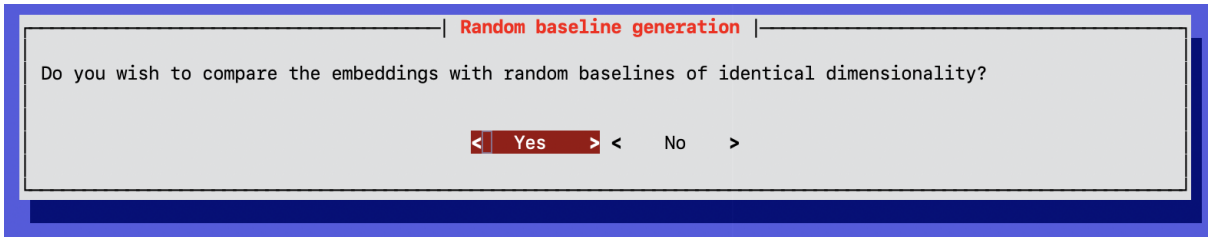
In order to execute to evaluation, both cognitive sources and embeddings need to be imported. All available CogniVal cognitive sources are imported by executing:

> *import cognitive-sources*

GloVe embeddings are imported by executing (note that this imports GloVe embeddings for all dimensionalities, as they are provided in one archive):

> *import embeddings glove.6B.50*

A prompt will be shown, asking whether the user wants to perform a random embeddings comparison. Make sure to respond with "Yes" (default), and random embeddings will be generated automatically.



Note that random embedding generation greedily uses at most *n-1* of *n* available CPU cores (up to 10 with default parametrization) for parallelized generation. By default, ten sets of random embeddings are generated, for which results are later averaged to improve robustness.

If needed, random embeddings can also be generated at any later point using the following command:

> *import random-baselines glove.6B.50*

Regeneration requires that *force=True* is added.

Step 3: Define experiments

Now we can move on to define the experiment details.

We want to evaluate Glove embeddings of 50 dimensions (glove.6B.50) against the EEG features from the ZuCo dataset (eeg_zuco).

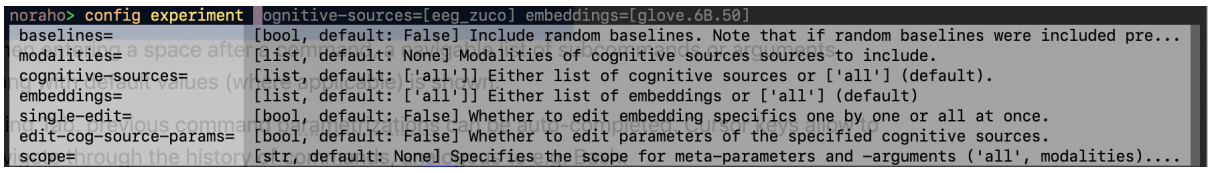
This can be edited using:

> *config experiment cognitive-sources=[eeg_zuco] embeddings=[glove.6B.50]*

(Note that CogniVal commands with multiple arguments always require that the arguments be given explicitly).

Save the experiment configuration without any changes. The changes are automatically propagated to the associated random embeddings. Note that if a cognitive source or embedding is not yet part of the configuration, it is automatically populated from the reference configuration, which contains default parameters for all cognitive sources and embeddings evaluated in the original CogniVal paper. Multiple experiments can be edited at once and if there are multiple values for a field, this is indicated in the editor.

When entering a space after a command, a navigable list of subcommands or arguments along with default values (where applicable) is shown. Example:



Using Tab, previous command parametrizations can be auto-completed. Cursor keys allow to navigate through the history of commands, analogous to e.g. Bash.

Step 4: Run evaluation experiments

The experiments can be run with the following command:

> *run*

You will see a loading bar counting 11 combinations (10 regression models for the 10 random embeddings and 1 for the Glove embeddings):

Be patient... it took 45 minutes for this demo using 2 CPU cores.
Better get a ☕...
(Note: as in any terminal, you can always abort a process with *Ctrl+C*.)

Step 5: Generate report with results

Finally, a HTML report can be generated and viewed in the default browser by executing:

> *report open-html=True*

If the report doesn't open directly (this depends on where you are running CogniVal), open the HTML file manually in a browser and have a look at the results. Obviously, with only one experiment the aggregation of the results is not very interesting, but at least you can see in the plot at the top how random embeddings compare to real word embeddings when predicting EEG data from ZuCo.

Evaluate your own combinations

Step 1: Decide which combinations to evaluate

Use the commands *list embeddings* and *list cognitive-sources* to see the word embeddings and cognitive sources that are available for evaluation in the CogniVal tool.

Use the same embeddings type (glove.6B.50) for your next evaluation, and add Glove embeddings of 100 dimensions (glove.6B.100). We work with these embeddings of only 50 and 100 dimensions so that you don't run into memory problem, but you can of course test any other embedding types, too.

Choose any EEG dataset or any eye-tracking feature from eye-tracking_zuco, eye-tracking_geco, eye-tracking_dundee, or eye-tracking_all.

Here's an overview of all eye-tracking features:

Description	Data source
First fixation duration	Dundee, GECO, Provo, UCL, ZuCo
First pass duration (first fixation duration in the first pass reading)	Dundee, GECO, Provo, UCL, ZuCo
Mean fixation duration	Dundee, GECO, Provo, ZuCo, CFILT-Sarcasm, CFILT-Scanpath
Fixation probability	Dundee
Re-read probability	Dundee
Total fixation duration	Dundee, GECO, Provo, ZuCo
Total duration of all regression going from this word	Dundee
Total duration of all regression going to this word	Dundee
Number of fixations	Dundee, GECO, Provo, ZuCo
Number of long regression (>3 tokens) going from this word	Dundee
Number of long regression (>3 tokens) going to this word	Dundee
Number of refixations	Dundee
Number of regressions going from this word	Dundee, Provo
Number of regressions going to this word	Dundee, Provo
The duration of the last fixation on the current word	GECO
Go-past time	GECO, Provo, UCL, ZuCo
No fixation occurred in first-pass reading	GECO, Provo
Right-bounded reading time	UCL

The embeddings and all cognitive sources have already been imported.

Step 2: Add the experiments to the configuration

Let's say we want to evaluate against some eye tracking features from the GECO corpus. So, we can define the experiment config for this combination:

> *config experiment cognitive-sources=[eye-tracking_geco] embeddings=[glove.6B.50,glove.6B.100]*

You can play around with parameters in this form if you want to experiment with different parameters. But you can also leave this configuration as it is with the default parameters, and save the form.

Check that this combinations has been added to the config:

> *config show details=True*

Step 3: Run the evaluation

The run the experiments with the features you want:

(Read step 4 before actually running it!)

> *run embeddings=[glove.6B.50,glove.6B.100] cognitive-sources=[eye-tracking_geco] cognitive-features= ["WORD_FIXATION_COUNT;WORD_TOTAL_READING_TIME"]*

Note: the cognitive features only have to be specified if you are using eye-tracking features, not for EEG. Also, if you only type *run* without any arguments it will run all experiments defined in the config, which would take a lot of time. If needed you can delete experiment from the config using e.g., *config delete cognitive-sources=[eeg_zuco] embeddings=[glove.6B.50] .* Check the README for more details.

Much faster! ...but that is because it was using 31 CPU cores :)

Step 4 (*optional*): Make it faster

If you are working on your own machine or the CPUs of the server you are working on are not too busy, check how many CPU cores are available and change the config to speed up the evaluation. You can always change the general properties with this command:

> *config open configuration=demo edit=True*

Change the value of *n_proc*, or leave the field empty (so that it will fall back to the default, i.e. number of CPUs -1). Save & exit, and go back directly to the *run ...* command.

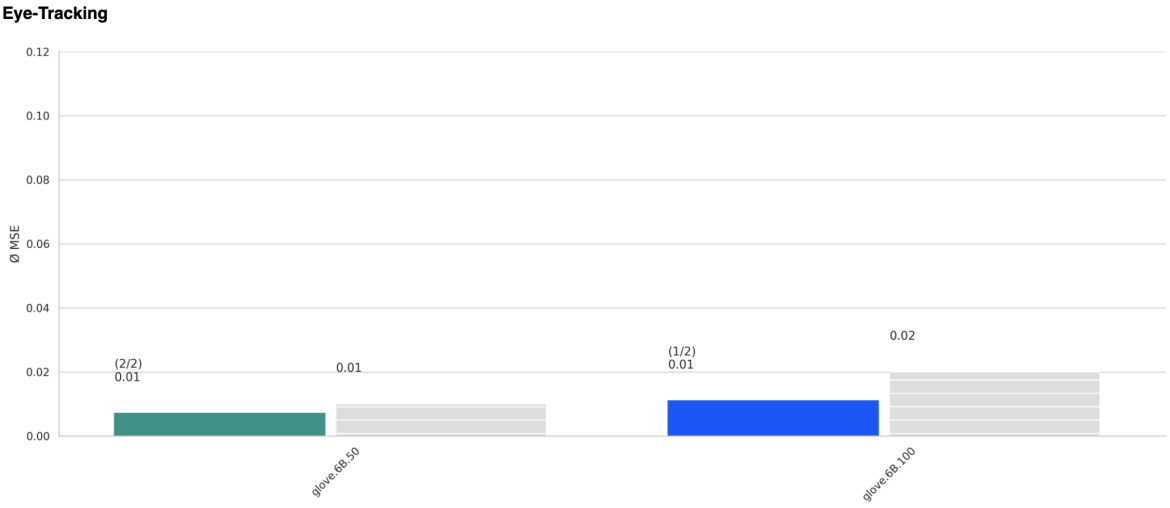
If you don't want to try this, just get another ☕, sit back and relax ....

Step 5: Generate report with results

Finally, generate a new report:

> *report open-html=True*

This time it should look more interesting!



Glove embeddings of both 50 and 100 dimensions are better than their corresponding random baselines!

Note: CogniVal automatically assigns run IDs to your different runs, and if there is more than one run available it will generate "statistics over time" plots, that you should now see in your report. These are useful if you are developing new word embeddings and want to track their improvements across different versions. For us, they are not useful at the moment and you can ignore them.

Useful tips

- The command *example-calls* list examples for all commands.
- The *welcome* command shows this welcome message and demo again.
- If you run intro memory problems, run single experiment configurations only.