WordWeaver Documentation

Aidan Pine

Oct 05, 2019

Contents:

Note: WordWeaver is UNDER CONSTRUCTION and should not be expected to be fully documented or even work as expected! Check back soon for more information.

WordWeaver is a tool for visualizing and interacting with computational linguistic models.

CHAPTER 1

Getting Started

1.1 Overview

1.1.1 What is WordWeaver?

WordWeaver is a Python library for turning an FST made with Foma into a RESTful API. It combines with the WordWeaver GUI to create an interactive web application for the data as well. WordWeaver was initially built for Kanyen'kéha but with all Iroquoian languages in mind. It will likely work for similar polysynthetic languages and Foma FSTs that model inflectional verbal morphology, but non-Iroquoian languages will likely have to modify the source in order to work.

1.1.2 Who made this?

WordWeaver is the outcome of collaborative research between the Onkwawenna Kenyohkwa Mohawk Immersion school and the Indigenous Language Technology research group at the National Research Council of Canada.

1.1.3 How do I make this for another language?

While we have made deliberate efforts to make WordWeaver simple to use, it will likely still require somebody with some experience with Natural Language Processing (NLP) in order to implement. The basic gist of it is that you will need to create or have the following:

1. An FST-based model of your language's inflectional verbal morphology (i.e. a model of how to make "conjugations"). Currently this must be a *.fomabin* file. Please contact us if you need other formats supported. (see *Data*)
2. Four configuration files for setting up your WordWeaver instance (see *Configuration*.)
3. JSON files containing all the verbs, pronouns and other affixes in your model (see *Data*.)
4. A Swagger Specification for your API (see *Data*)

For steps on what to do next, please visit the *Guides*.

CHAPTER 2

Installation

You can either install `wordweaver` with pip from PyPi:

```
pip install wordweaver
```

Or by cloning and installing from source:

```
git clone https://github.com/nrc-cnrc/wordweaver.git

cd wordweaver

pip install -e .
```

CHAPTER 3

CLI

3.1 wordweaver

Management script for WordWeaver

```
wordweaver [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Show the flask version

--version
Show the version and exit.

3.1.1 foma

Interact with foma through command line

```
wordweaver foma [OPTIONS] [up|down|lower-words] [INP]
```

Options

--plain, --no-plain

--txt, --no-txt

--pkl, --no-pkl

Arguments

COMMAND
Required argument

INP
Optional argument

3.1.2 routes

Show all registered routes with endpoints and methods.

```
wordweaver routes [OPTIONS]
```

Options

-s, --sort <sort>
Method to sort routes by. "match" is the order that Flask will match routes when dispatching a request.

Options endpoint|methods|rule|match

--all-methods
Show HEAD and OPTIONS methods.

3.1.3 run

Run a local development server.

This server is for development purposes only. It does not provide the stability, security, or performance of production WSGI servers.

The reloader and debugger are enabled by default if FLASK_ENV=development or FLASK_DEBUG=1.

```
wordweaver run [OPTIONS]
```

Options

-h, --host <host>
The interface to bind to.

-p, --port <port>
The port to bind to.

--cert <cert>
Specify a certificate file to use HTTPS.

--key <key>
The key file to use when specifying a certificate.

--reload, --no-reload
Enable or disable the reloader. By default the reloader is active if debug is enabled.

--debugger, --no-debugger
Enable or disable the debugger. By default the debugger is active if debug is enabled.

--eager-loading, --lazy-loader
Enable or disable eager loading. By default eager loading is enabled if the reloader is disabled.

--with-threads, --without-threads
Enable or disable multithreading.

--extra-files <extra_files>
Extra files that trigger a reload on change. Multiple paths are separated by ':'.

3.1.4 shell

Run an interactive Python shell in the context of a given Flask application. The application will populate the default namespace of this shell according to it's configuration.

This is useful for executing small snippets of management code without having to manually configure the application.

```
wordweaver shell [OPTIONS]
```

3.1.5 spec

Update Swagger Specification

```
wordweaver spec [OPTIONS]
```

CHAPTER 4

WordWeaver

4.1 Build Tools

These are tools that build and compile various files needed by WordWeaver.

The FileMaker class is used to create both docx and latex outputs of conjugations.

class wordweaver.buildtools.file_maker.**FileMaker**(*conjugations=[]*)
Takes conjugations and creates files (docx, latex or pdf)

class wordweaver.buildtools.file_maker.**DocxMaker**(*conjugations*)

class wordweaver.buildtools.file_maker.**LatexMaker**(*conjugations*)

4.2 FST

This folder deals with interactions between the API and the fomabin language model.

Requests must be encoded into tags for the FST:

class wordweaver.fst.encoder.**FstEncoder**(*args*)
A class for batch creating upper-side sequence of morphological tags to be submitted to the FST with down fst_tag Template follows spec from FST_CONFIG['template']

The output of the FST must be decoded into a response that the API returns:

class wordweaver.fst.decoder.**FstDecoder**(*fst_output*)
Turn FST Output like ^PP-^seni^R-^khonni^R^ into values for HTTP response

Translations into English are done through the EnglishGenerator class:

class wordweaver.fst.english_generator.**EnglishGenerator**
Generate basic plain English based on tag from FstTagMaker

4.3 Resources

This folder contains all resources for the RESTful WordWeaver API.

CHAPTER 5

WordWeaver UI

The WordWeaver User Interface is one way to *visualize* the data that WordWeaver makes accessible through its API.

The interface is an Angular web application and is available here at https://github.com/roedoejet/wordweaver-GUI.

Edits to the interface for your particular instance of WordWeaver should be purely stylistic, most of the other changes should be able to be done through the *Configuration* files.

CHAPTER 6

Guides

Here are some guides to help do some of the basic tasks required for creating a WordWeaver instance.

6.1 Running your instance

1. Make sure you have all the required data and configuration files described in *Data* and *Configuration*
2. Copy the sample directory and replace all of the data and configuration files with your own.
3. Set the environment variable WW_CONFIG_DIR equal to the absolute path to the folder called `configs` from step 2.
4. Set the environment variable WW_DATA_DIR equal to the absolute path to the folder called `data` from step 2.
5. Run the following python code, either from the interpreter or in a script:

```
from wordweaver.app import app
app.run()
```

If you have gunicorn installed on your machine, you can also run it from the command line:

6.2 Adding/editing a verb

1. Add/edit the verb in your `verbs.json` file. (see *Data*)

```
{
"display": "wake'nahsan\u00e9n:taks",
"eng-3": "is tongue tied; gets tongue tied",
"eng-inf": "be tongue tied; get tongue tied",
"eng-past": "was tongue tied; got tongue tied",
"eng-perf": "been tongue tied; gotten tongue tied",
```

(continues on next page)

(continued from previous page)


```
"eng-prog": "being tongue tied; getting tongue tied",
"gloss": "be tongue tied; get tongue tied",
"root": "'nahsanentak",
"state_type": "hab",
"stative-perf-trans": "",
"stative-pres-trans": "",
"tag": "7nahsanentak-b",
"thematic_relation": "blue"
}
```


2. Update your Swagger Spec with `wordweaver spec` (see *Data*)

That's it! Next time you run your WordWeaver instance, the verb will be there.

6.3 Adding/editing a pronoun

1. Add/edit the pronoun in your `pronouns.json` file. (see *Data*)


```
{
"person": "1",
"number": "SG",
"gender": "",
"inclusivity": "",
"role": "",
"value": "ke",
"gloss": "I",
"obj_gloss": "Me",
"tag": "1-sg"
}
```


2. Update your Swagger Spec with `wordweaver spec` (see *Data*)
3. Update the `pronoun` key in your interface configuration file. (see *Configuration*)

That's it! Next time you run your WordWeaver instance, the pronoun will be there.

6.4 Adding/editing an new temporal option

This step is for adding/editing a new aspect (or tense) to your model.

1. Ensure that you have added the affixes needed by your new aspect.
2. Add/edit your aspect/tense to `affix_options` in your language configuration file. (see *Configuration*)


```
affix_options:
  - tag: habpres
    gloss: Habitual (present)
    affixes:
      - habitual
      - pres
    public: true
```


3. Update your Swagger Spec with `wordweaver spec` (see *Data*)

That's it! Next time you run your WordWeaver instance, the tense/aspect will be there.

6.5 Adding an affix

6.5.1 Adding/editing an optional affix

This step is for adding/editing affixes that must be selected through 'affix options'.

1. Add/edit the affix to `affixes.json`. (see *Data*)

```
{
"gloss": "perfective",
"type": "aspect",
"morphemes": [],
"tag": "perf"
}
```

2. Add/edit the affix to under the proper type beneath the `affixes` key in your language configuration file. (see *Configuration*)

```
affixes:
  aspect:
    perf:
       tag: "+Perf"
       marker: "R"
```

3. Add/edit it for any tense/aspect affix options that require it.
4. Update your Swagger Spec with `wordweaver spec` (see *Data*)

That's it! Next time you run your WordWeaver instance, the affix will be there.

6.5.2 Adding an affix required by certain verbs

This step is for adding affixes that are required by verbs but cannot be optionally added through affix options.

1. Add/edit the affix to `affixes.json`. (see *Data*)

```
{
"gloss": "duplicative",
"type": "prepronominal_prefix",
"morphemes": [],
"tag": "dup"
}
```

2. Add/edit the affix to under the proper type beneath the `decoding` and `bundled_affixes` keys in your interface configuration file. (see *Configuration*)

```
decoding:
  bundled_affixes:
    dup: TE
```

3. Add/edit it for any verbs that require it.

```
{
    "display": "tekonia'ni\u00e1nawenks",
    "eng-3": "puts gloves on someone",
    "eng-inf": "put gloves on someone",
    "eng-past": "put gloves on someone",
    "eng-perf": "put gloves on someone",
    "eng-prog": "putting gloves on someone",
    "gloss": "put gloves on someone",
    "required_affixes": [
        "dup"
    ],
    "root": "a'nyanawenk",
    "state_type": "hab",
    "stative-perf-trans": "",
    "stative-pres-trans": "",
    "tag": "a7nyanawenk-p",
    "thematic_relation": "purple"
}
```


4. Update your Swagger Spec with `wordweaver spec` (see *Data*)

That's it! Next time you run your WordWeaver instance, the affix will be there.

CHAPTER 7

Configuration

There are four configuration files (yaml) that inform WordWeaver.

- A **build** configuration file that informs the Build Tools
- An **environment** configuration file that specifies certain run-time variables and security policies
- An **interface** configuration file that specifies the way that WordWeaver interacts with the language model
- A **language** configuration file that specifies variables about the language.

Below is an in-depth description of each configuration file. However, we recommend just altering the sample configuration files instead of writing yours from scratch.

For help on how to change specific things about your WordWeaver instance, check out the *Guides* section.

7.1 Build Configuration

7.2 Environment Configuration

7.3 Interface Configuration

7.4 Language Configuration

CHAPTER 8

Data

You must provide some static data for WordWeaver including the *fomabin* of your language model, a swagger template and flat files (json) containing data about your language's pronouns, verbs and affixes.

8.1 JSON data

8.1.1 Affixes

8.1.2 Pronouns

8.1.3 Verbs

8.2 Foma Binary

You must have a valid Foma binary.

8.3 Swagger

WordWeaver uses Swagger to document its API. We recommend using the default swagger spec instead of writing your own. In order to update your swagger spec, run the following code:

```
gen = SwaggerSpecGenerator()
gen.writeNewData()
```

You can also run the above code in the command line as follows:

```
wordweaver spec
```

We recommend integrating this into a CI/CD pipeline for your WordWeaver instance.

Note: This will only edit your swagger spec at $WW_DATA_DIR/swagger/swagger-pre.json

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index

Symbols

C

D

E

F

I

L

W