

# **Building annotation tools and demos with Flask**

Emiel van Miltenburg

# Rationale

Available tools aren't always enough.

## **Problems:**

- Multiple data sources
- Multimedia
- Special source format

# Requirements

- Quick to implement/modify (iterative process)
- Easily extensible
- Intuitive interface
- Allows any kind of media

# Solution



# Flask

web development,  
one drop at a time

# Advantages of Flask

## **General**

- Lightweight
- Easy to learn

## **Python**

- Re-use your code to load the data
- Turn any project into a demo

## **Web-framework**

- Layout in HTML/CSS
- Display anything your browser can handle

# Examples

<

>

127.0.0.1

↑

□

+

**Image 8245313425 (31778/31783)**

[<<< First](#) | [< Previous](#) | [Random](#) | [Next >](#) | [Last >>>](#)

kind of inference ▾

Feature:

No Feature:

Variation

Submit



© mark-shaikun - photography  
markshaikun.com

# Examples

The screenshot shows a web browser window titled "Sound Browser -- The VU Sound Corpus". The address bar displays "127.0.0.1:5000/browse/1". The main heading is "The VU Sound Corpus". Below it is a search bar with the text "all" and a red "Search" button. Under the search bar are four radio buttons: "Crowd-tags", "Author-tags", "Description" (which is selected), and "ID". The search results show "Sound #2300" as "Result 2/2133". Navigation links "Previous" and "Next" are present. The description of the sound is "small frequency change beat on a yamaha pss 480.". Below the description is a media player with a play button, a progress bar, a time display of "0:08", a volume icon, and a red "Save" button. Underneath the media player, there are two sections: "Crowd" with tags "electronic; electronic sounds; synthesiser; synthesizer; swoosh; whine; alien; alien sound effects; computer; computer game; robot; musical; techno sounds" and "Freesound" with tags "beat; keyboard; radio; space; synth". At the bottom, there is a "Citation information" section with two lines of text: "Annotations: Emiel van Miltenburg, Benjamin Timmermans and Lora Aroyo (2016) *The VU Sound Corpus: Adding more fine-grained annotations to the Freesound database*. To appear in: Proceedings of LREC." and "The Freesound database: Frederic Font, Gerard Roma, and Xavier Serra (2013) *Freesound technical demo*. In: Proceedings of the 21st ACM international conference on Multimedia. ACM."

Sound Browser -- The VU Sound Corpus

127.0.0.1:5000/browse/1

## The VU Sound Corpus

all Search

☐ Crowd-tags ☐ Author-tags ☒ Description ☐ ID

**Sound #2300**  
Result 2/2133

[Previous](#) - [Next](#)

small frequency change beat on a yamaha pss 480.

0:08 Save

**Crowd** electronic; electronic sounds; synthesiser; synthesizer; swoosh; whine; alien; alien sound effects; computer; computer game; robot; musical; techno sounds

**Freesound** beat; keyboard; radio; space; synth

**Citation information**  
Annotations: Emiel van Miltenburg, Benjamin Timmermans and Lora Aroyo (2016) *The VU Sound Corpus: Adding more fine-grained annotations to the Freesound database*. To appear in: Proceedings of LREC.  
The Freesound database: Frederic Font, Gerard Roma, and Xavier Serra (2013) *Freesound technical demo*. In: Proceedings of the 21st ACM international conference on Multimedia. ACM.

<http://cltl.nl:5432/>

# Examples

Home

Word similarity

List similarity

Similar words

Analogy

Outliers

Contact

## A distributional semantic model for Dutch

By [Emiel van Miltenburg](#)

Welcome to this demo of a distributional semantic model for **Dutch**. This model contains knowledge about the way Dutch words are related to each other. In the menu on the left, you can see several different options showcasing the capabilities of the model. This page is in English to also make it accessible to researchers who are interested in Dutch, but do not speak the language.

**Word similarity** is a simple tool to get a similarity rating for any two given words. A higher rating corresponds to a greater similarity.

**List similarity** allows you to enter a list of word pairs, and returns a table with the similarity values for those word pairs.

**Similar words** lets you enter one or more words from a particular category, and it will suggest similar words.

**Analogy** is a function that tries to complete analogies, e.g. *man* is to *king* as *woman* is to ... (answer: *queen*).

On the **Outliers** page you can enter a list of words, and the function will try to return the word that does not belong in the list.

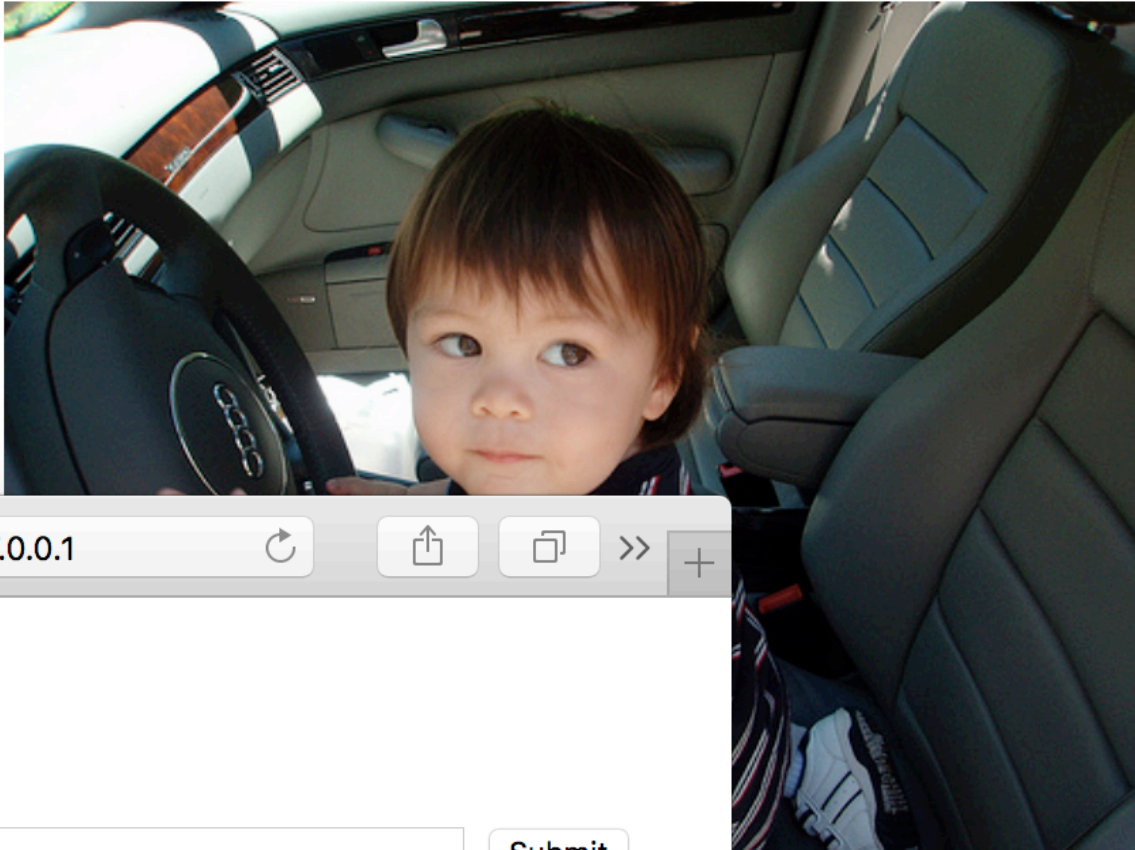


# Examples

127.0.0.1

Annotate the image

Click the relevant category.



Growing set of labels!

☒ Other, namely:

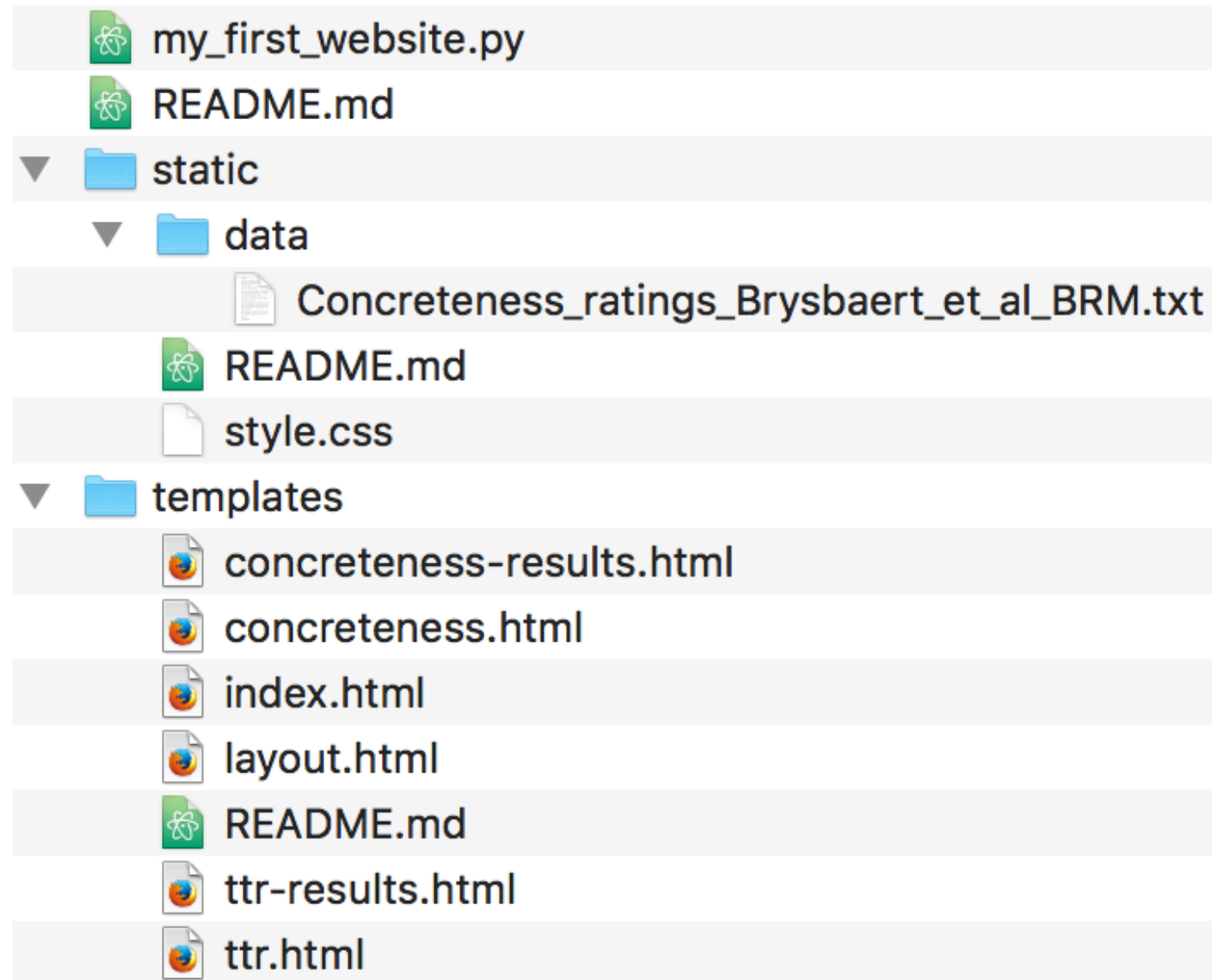
Submit

127.0.0.1

Submit file

Enter a filename, then click 'submit'.

# Anatomy of a tool



# Important parts

- **Main file:** `my_first_website.py`
- **Templates:** `templates/*.html`
- **Data:** `static/data/*`

# Main file

where the magic happens

1. Import statements
2. Load your data
3. Define your pages

# Templates

Always stored in `./templates/`

- Define what your page looks like.
- Either page-by-page, or *inherit* from master.
- Templates may contain code, in Jinja format.
- Access to variables passed with `render_template`
- Return results using web forms.

# Web forms

- Follow the HTML form standard.

```
10 <form action="/concreteness-results/" method="POST">
11 <textarea rows="4" cols="50" name="textfield">
12 Put some text here, and press submit to see the concreteness score!
13 </textarea>
14 <input type="submit" value="Submit">
15 </form>
```

- POST submits data to a web server.
- Trick: `<input type="hidden" name="country" value="Norway">`

# Tips

Things I've learned along the way

- Modifying your code will reload the entire script.
- Annotate in batches.
- Test your code before annotating larger amounts.
- Save intermediate results.
- Use Javascript for dynamic elements.
- Use Python 3. (Python 2 *will* give unicode errors.)

# Tips

- Draw your annotation screen before you build it.
- Modest color scheme = happier eyes.
- Use regular expressions to improve search.
- Demo's are not just for other people;
- Play around with your data!