DOTS - Drift Oriented Tool System

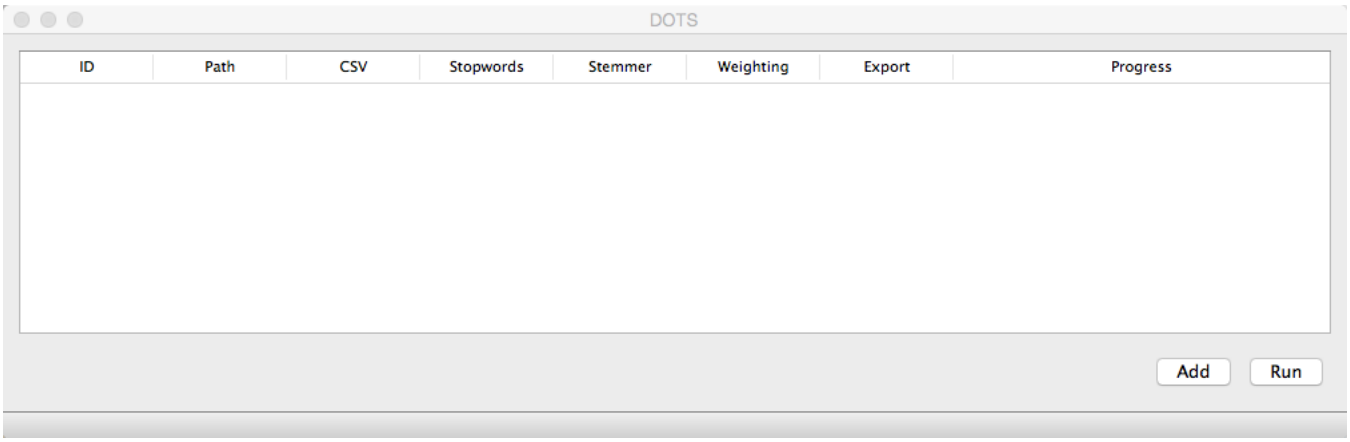
DOTS

DOTS is a drift oriented framework developed to dynamically create datasets with drift. The major goal of this framework is to create labeled datasets that can be used to simulate different drift patterns that will evaluate and validate learning strategies used in dynamic environments. It allows multiple tasks to be defined at once so user can test multiple scenarios.

For each task, user must supply as input a list of files containing documents, one file for each class, and a frequency table describing the frequency of each class over time. As output, it creates a directory with an Indri index directory, containing all the Indri files, a trainset directory containing all the trainset files and a testset directory, containing all the testset files.

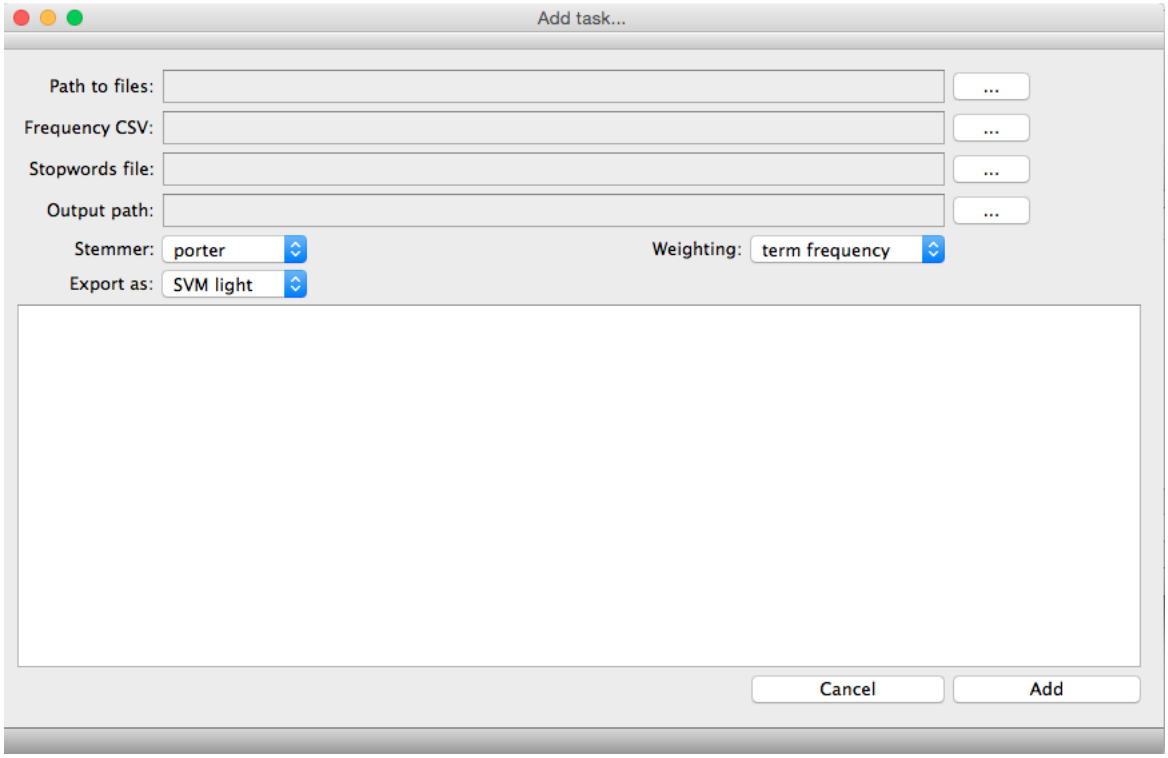
It is a very simple to use framework, with a user-friendly interface.

Note: The current release is only available for MacOS.



Add task

It is possible to add tasks to DOTS in two different ways. The first one is using the DOTS interface by using the button the "Add" in the main DOTS window.



"Add tasks" Parameters:

Path to files [mandatory] - contains the path for directory that contains the documents text files. The whole collection must be in the same directory and each file represent a class of documents. Each line of the text file represent a document and documents are sequential, which means that the firsts to appear in the file are the firsts to appear in the time sequence that is represented in the *frequency csv* parameter.

Example:

class1.txt containing

document_1_text
document_2_text
document_3_text

class2.txt containing

document_4_text
document_5_text
document_6_text

Document 1, 2 and 3 belong to class 1, while document 4, 5 and 6 belong to class 2.

Frequency CSV **[mandatory]** - contains the path for the frequency CSV table. This table must be in the comma-separated values format with the comma as the separation character. Each row correspond to a time instance and each column represent a class. The first row must contain the filename of the text file that contains the represented class. In each cell there is the number of documents of a given class that occur in a given time instance.

Note: A documents text file is only considered if it is represented in the frequency CSV, otherwise will be discarded. Two or more files must be represented in the frequency table along with two or more time instances, ie, rows beside the first one.

Example:

class1.txt	class2.txt
1	2
2	1

In the first time instance there is one document of class 1 and two documents of class 2, while in second time instance there is two documents of class 1 and one document of class 2. Considering the previous example documents 1, 4 and 5 belong to the first time instance, while documents 2, 3 and 6 belong to the second time instance.

Stopwords file **[optional]** - contains the path for the stopwords file. This file must be a text file containing one stopword in each line.

Output path [optional] - contains the path for the output directory where dataset files will be placed. If not specified, the application will use the *path to files* directory.

Stemmer **[mandatory]** - this element specifies the stemming algorithm to be used.Valid options are:

- *none* – no stemming algorithm
- *porter* – Porter stemmer
- *krovetz* – Krovetz stemmer

Export as **[mandatory]** - this element specifies the exporting file type. Valid options are:

- *SVM-light* – SVM-light format to be used with SVM-light software
- *ARFF* – Attribute-Relation File Format (ARFF) to be used with Weka software

Weightning **[mandatory]** - this element specifies the weighting scheme used in document representation. Valid options are:

- *term-frequency* – term frequency
- *tf-idf* – term frequency - inverse document frequency

Another possible way of adding tasks it using INI files. INI files are structured as sections (defined using []) with pairs of keys and corresponding values (defined as key=value). Comments are also possible (defined as lines started with ';').

Example:

```
[section_1]
key_a=value_a1
key_b=value_b1
 ; this is a comment
```

```
[section_2]
key_a=value_a2
key_b=value_b2
 ; this is another comment
```

To define a task using files there are pre-defined keys which values must be supplied. The name of each section is irrelevant, though they must be defined.

Mandatory keys are:

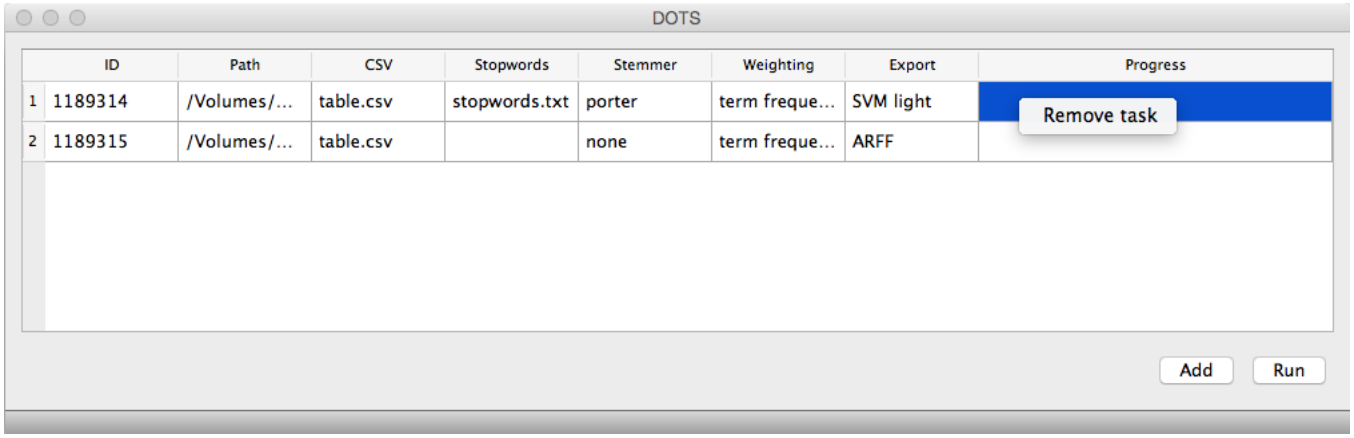
- **input_path** – complete path to the input directory
- **csv** – complete path to the CSV file containing the frequency table
- **stemmer** – stemmer algorithm, possible values: *'none'*, *'porter'* and *'krovetz'*
- **export** – export as, possible values: *'SVM Light'* and *'ARFF'*
- **weighting** – weighting scheme, possible values: *'term frequency'* and *'tf-idf'*

And optional keys are:

- **stopwords** – complete path to the stopwords file
- **output_path** – complete path to the output directory

Remove task

It is also possible to remove tasks that were incorrectly defined. After being defined, all tasks might be removed by using the mouse right-button over the defined task in the DOTS main window.



Run tasks

After defining all task, the user can run them at once, by using the button "Run" in the DOTS main window. Feedback will be given by the framework in order to follow the completeness state.