|  |
| --- |
| Floodtags |
| Project transfer document |
| Importance rating algorithm |

|  |
| --- |
| Bert van Nimwegen  16-6-2016 |

# Introduction

This document describes the code written during the internship of Bert van Nimwegen.

The algorithm is made to cluster and rate tweets to sort tweets based on importance.

The code can be found at <https://github.com/bertvn/floodtags>

# Code structure

The code is structured according to the guide found at the following link: <http://docs.python-guide.org/en/latest/writing/structure/>

The folder api contains all classes related to gathering tweets.

Datascience contains the folders clustering, which has all classes related to clustering, and filtering, which has all classes related to rating the clusters.

Linguistics contains all language related classes

language contains wordlist classes and files. Each supported language gets its own folder.

ner contains the handler class for the Stanford NER as well as the java files for the NER

nlcs contains the longest (most) common string classes

sanitizing contains the regular expression handler class

core contains the class for dependency injection, classes for formatting the output of the algorithm and a class containing static variables that are used throughout the application.

# Design pattern

To make the code more testable it has been build according to the dependency injection / inversion of control pattern. This uses a container to build each dependency instead of letting classes build their own dependencies. Normally this would be done by using interfaces, but as these are not available in python the choice has been made to use either an abstract base class or to not specify it at all. As python can work with each class if the right functions are implemented, the interfaces or abstract base class are only there to enforce it.

Inversion of control/dependency injection

The pattern is explained in the following article: <http://martinfowler.com/articles/injection.html>

This is made for Java not for python, the following link shows a python implementation:

<https://web.archive.org/web/20130703221553/http://www.emilmont.net/doku.php?id=python:design_patterns:inversion_of_control>

The version implemented in the code is an altered version of the code above. There it created every class beforehand; in the code it makes the class and it dependencies when that class is requested.

# Compatibility

The SSE which is used to push information to the web application doesn’t natively work on internet explorer/edge, to help this a polyfill is used. This simulates the build in implementation of SSE if a browser doesn’t have one. The one used is in the link below.

<https://github.com/remy/polyfills>

For more information on SSE the article below gives a good explanation

<http://www.html5rocks.com/en/tutorials/eventsource/basics/>

# Possible improvements

## Ner

The Named Entity Recognizer currently only works for English text, while polyglot is incorporated in the project it’s NER is not yet used when a different language is detected. This will greatly improve the performance when the algorithm is applied to languages other than English. Polyglot requires language files to be downloaded for each language that needs to be analyzed, please note that does not work on Windows at this moment, it does however work on Ubuntu and most likely on Mac as well. Windows users can use a VM to download it.

## Normalization

The current normalization is based on the perfect cluster; the problem is that most clusters are very far from this ideal. This means that importance values are very low, changing it to be more lenient would allow for more reasonable importance values.

## Rewrite cluster algorithm

Right now the tweets are converted to work with the vectorization and k-means algorithms of scikit learn. These are very good implementations of both, but are not designed to handle tweets. By writing a k-means algorithm specifically designed to tweets the clustering might be improved. This also allows for weighted variables, with the scikit version everything is seen as equal even though some attributes are of greater importance than others. Giving them a higher priority will let that importance come to the surface more easily.

## Rewrite in a different language

Python is amazing for making small test, but is lacking when it comes to maintaining large projects. There is also the problem with dependencies, as each has to be installed on the host computer. The might not be as easy to install on some systems, requiring some work to get the program working. Compiler languages have the option to embed most dependencies, requiring only a version of the code that is compiled for the operation system it has to run on or the java virtual machine in case of java. Compiler languages also tend to have higher performance and have better support for multithreading than python. This might help shave minutes or at least seconds from the run time of the algorithm. If java is used it would also open the option to integrate ELKI into the algorithm, which offers various additional algorithms for data science research.

Java would be a good option as the algorithm already uses java for running the Stanford Named Entity Recognizer. By incorporating it into the code itself it should run faster as it doesn’t have to start the Java Virtual Machine to run.