Introduction

The solution has been written in *Python 3.7.9*, using *pyspark 3.0.1*. *Spark* 3.0.2 were used, as the encouragement to use *Spark* 2.4.x or *Spark* 2.2.x were not noticed until the project was finished. The project has been run inside a *conda virtual environment*, and the *requirements.txt* file can be used to set up a new environment.

The tasks have mostly been solved by writing a sequential program, with some helper functions.

The script can be ran using the following command if the data is located in a folder called "data" in root of the project. Note that *post_id* must be specified.

```
$ python3 main.py --post_id some_post_id
```

If the data folder is located somewhere else, the *input_path* flag can be set. The script can then be run used the following command.

```
$ python3 main.py --input_path /path/to/data/folder --post_id some_post_id
```

Task walkthrough

The data preparation is mostly solved using *map*, *filter* and simple functions on the RDDs as well as pure python operations. The task walkthrough is quite short, as writing extensive descriptions of each task would be very time consuming. Have a look at the code for reference.

First data is loaded from a csv file into an RDD, where header is removed, then everything but the body is removed. The body is then decoded from base64 encoding, and then converted to lowercase.
The order of the following steps is made according to the task description. Bad strings are removed from the body text, while the body is still one big string. The string is split/tokenized on whitespace. The *RDD* now consists of one line, containing a list of terms. This line is parsed into a new *RDD* where each line is a term. This approach was chosen to follow the task description close but still utilizing the benefits of an *RDD*. Then terms shorter than three characters are filtered away, before each term is stripped from bad start/end strings.
The distinct terms are saved, to later be used for vertices when running *page rank.* Windows for sliding window algorithm are created, and then tuples, representing edges between terms, are created from the windows. The list of edge tuples is converted into an *RDD*. The unique terms are used to create a *DF* of vertices. The edge *RDD* is used to create a *DF* of edges. The edge- and vertices *DFs* are used for the creating of a *graphframes* graph. The *page rank* algorithm is run on the graph to find the importance of each term. The arguments of resetProbability=0.15 and tol=0.0001 were chosen according to task description and piazza discussion.