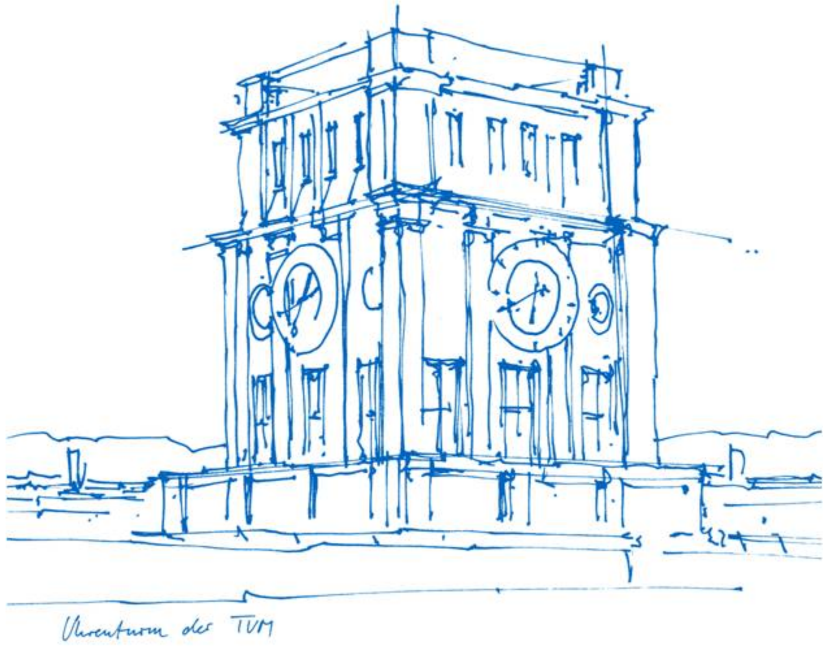
Exercises for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to Exercise 5



Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python and Network Visualization	May 24-30
2	• Centrality Measures	May 31 – June 6
3	• Finding Groups & Clustering Methods	June 7 – 13
4	• Predicting Social Tie Strength with Linear Regression	June 14 - 20
5	• Natural Language Processing: Hate Speech detection + Social Context Influence	June 21 - 27
6	• Natural Language Processing: Modern Machine Learning Methods and Explainable AI	June 28 – July 4

Exercise Sheet 5 – Hate Speech

Goals:

- Ability to use basic NLP to analyse tweets and other textual social media artifacts by using a Keras Neural Network.
- Compare two models to see the impact of including additional social context.

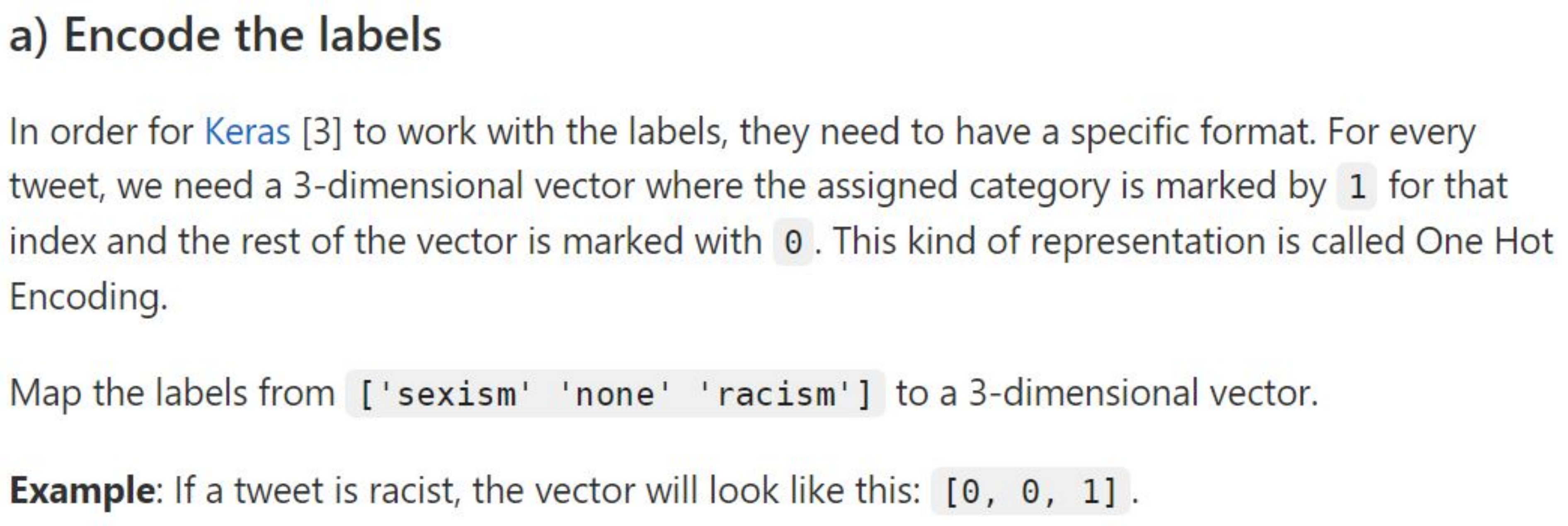
Data set:

- hand-labeled tweets by Waseem & Hovy [1]. Each tweet is either labeled as 'sexism', 'none' or 'racism'.

The Data

- src.evaluation.py: provides useful functions for you to use (F1-Score)
- pickle_files.users_data: provides social information about the authors
  - authorship.npy: author for each tweet
  - users_adjacency_matrix.npy: the follower network of the authors
- tweets.csv: provides you with the labeled tweets from Waseem & Hovy

Task 5.1: Prepocessing

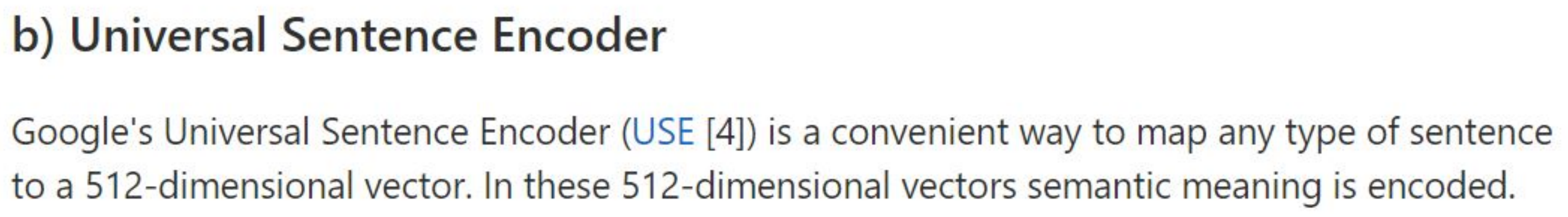


a) Encode the labels

In order for Keras [3] to work with the labels, they need to have a specific format. For every tweet, we need a 3-dimensional vector where the assigned category is marked by `1` for that index and the rest of the vector is marked with `0`. This kind of representation is called One Hot Encoding.

Map the labels from `['sexism' 'none' 'racism']` to a 3-dimensional vector.

Example: If a tweet is racist, the vector will look like this: `[0, 0, 1]`.



b) Universal Sentence Encoder

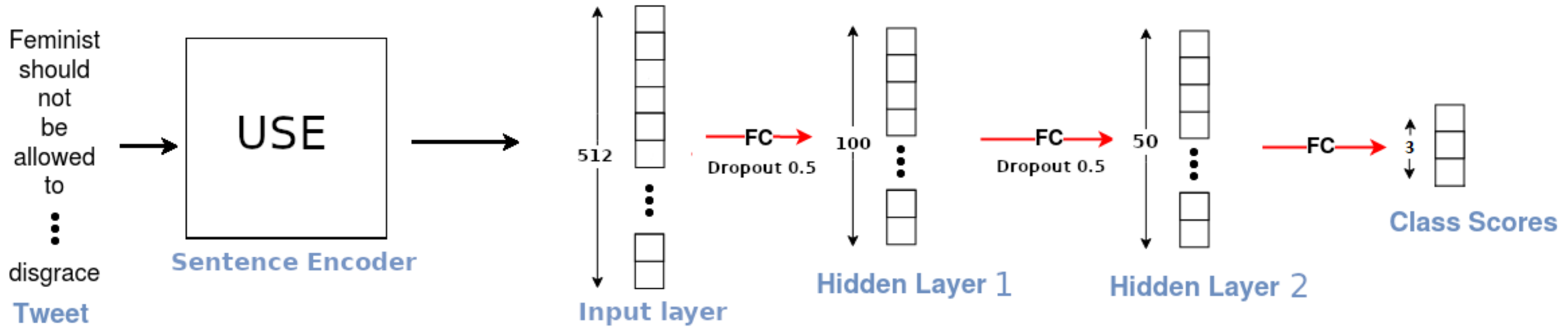
Google's Universal Sentence Encoder (USE [4]) is a convenient way to map any type of sentence to a 512-dimensional vector. In these 512-dimensional vectors semantic meaning is encoded.

Task 5.2: Base Model

a) Base Model creation

In this code we create our base model and train it afterwards using the Keras library. We use a simple Neural Network to build this model. You can read about Neural Networks here in case you are not familiar with them. You can get a basic intuition for Neural Networks here [5].

For the base model we have our 512 dimensional input layer. Then we have a fully connected layer with 100 nodes and with a dropout rate of 0.5 is added. For now, you do not need to know what dropout is. After the dropout, another fully connected layer with 50 nodes is added and we once again add a 0.5 dropout rate. Our output layer has 3 nodes: One for "sexism", "none" and "racism". The computed values for these last 3 nodes correspond to the probability of belonging to either one of our categories.



Task 5.2: Base Model

- Define the model‘s structure and compile it.
- Perform a Train-Test split on our data.
  - Train set 60%
  - Validation set 20%
  - Test set 20%
- Train the model.
- Evaluate the performance by using the F1-Score.

F1-Score

The F1 score is a universal measurement of a test's accuracy. It is calculated as the harmonic mean of *precision* and *recall*.

- **precision** refers to the number of true positives divided by the number of all positives
- **recall** refers to the number of true positives divided by the number of relevant elements

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

where

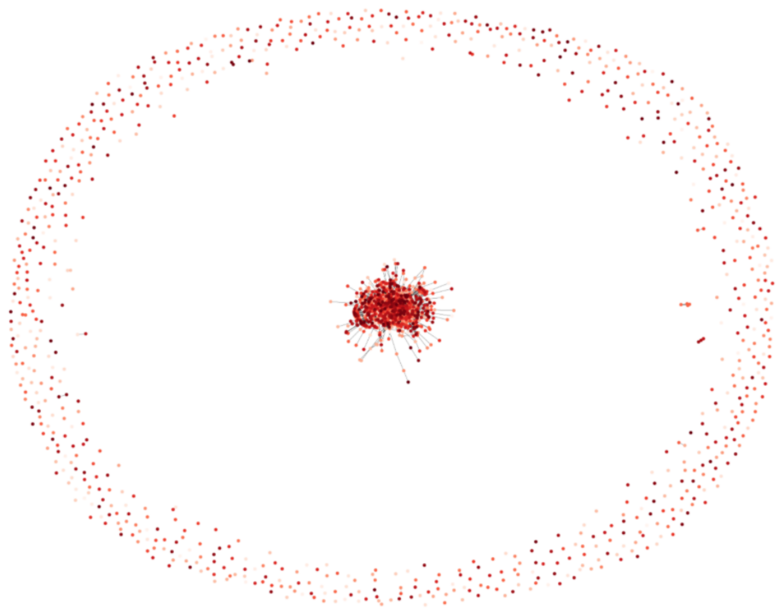
- tp = true positives
- fp = false positives
- fn = false negatives

Task 5.3: Preprocessing for Social Model

a) Graph visualization & manipulation

Now we are going to plot the previously loaded adjacency matrix. Since we are going to feed the matrix to the Neural Network later, and because the user network is just a tiny subset of the whole Twitter network it is important to check if the network contains any useful information.

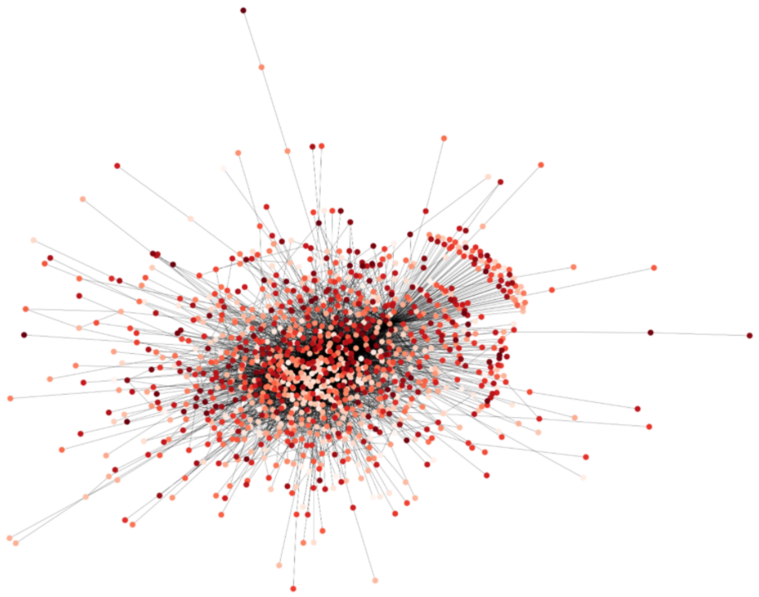
1. Plot the graph corresponding to the given adjacency matrix.
Note: For better visualization, you can color-code the nodes based on degree.



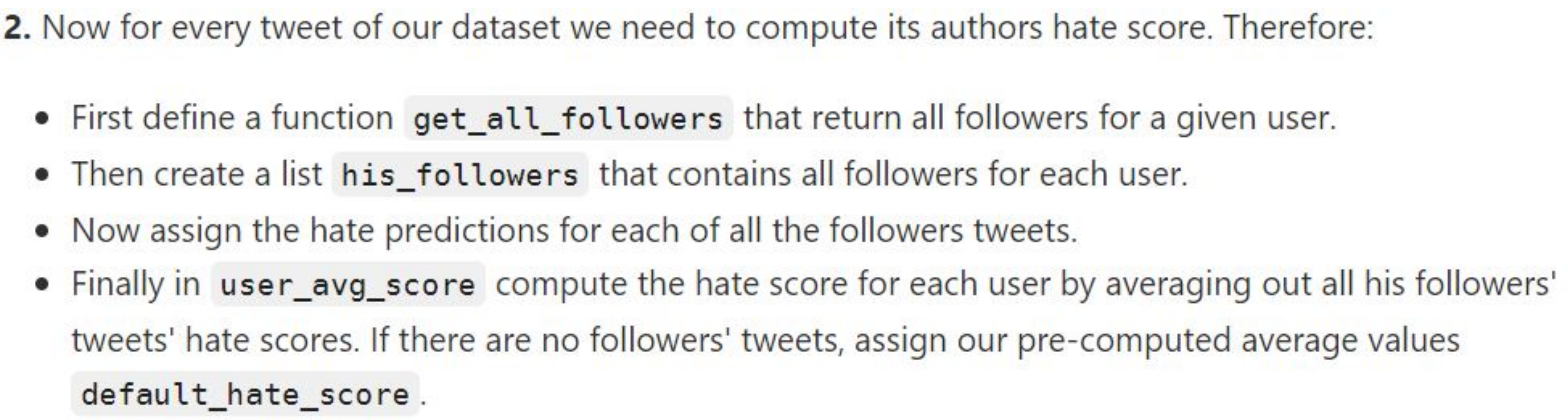
Task 5.3 a): Preprocessing for Social Model

3. Now let us try to actually calculate the number of communities within this graph. First, get rid of the uninteresting nodes that have zero or very few edges and just inspect the "core" graph. Expand on the code that you have written in the exercise above.
Hint: You can do this by excluding all nodes with an eigenvector centrality lower than 10^{-8}.

- NetworkX offers a pre-defined function: nx.eigenvector_centrality()



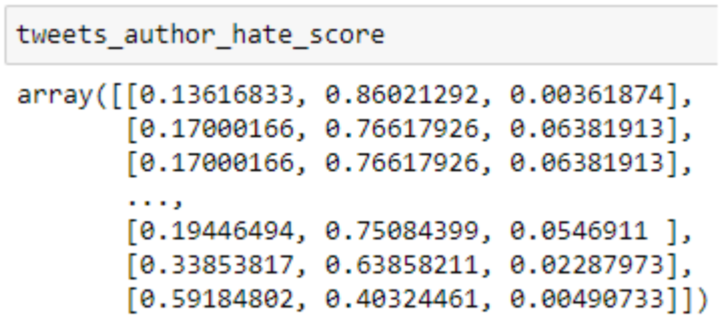
Task 5.3 b): Preprocessing for Social Model



2. Now for every tweet of our dataset we need to compute its authors hate score. Therefore:

- First define a function `get_all_followers` that return all followers for a given user.
- Then create a list `his_followers` that contains all followers for each user.
- Now assign the hate predictions for each of all the followers tweets.
- Finally in `user_avg_score` compute the hate score for each user by averaging out all his followers' tweets' hate scores. If there are no followers' tweets, assign our pre-computed average values `default_hate_score`.

- The default_hate_score values are the average values over all hate scores and have been precomputed for you.
- Your tweets_author_hate_score should look like this:



```
tweets_author_hate_score
```

```
array([[0.13616833, 0.86021292, 0.00361874],
       [0.17000166, 0.76617926, 0.06381913],
       [0.17000166, 0.76617926, 0.06381913],
       ...,
       [0.19446494, 0.75084399, 0.0546911 ],
       [0.33853817, 0.63858211, 0.02287973],
       [0.59184802, 0.40324461, 0.00490733]])
```

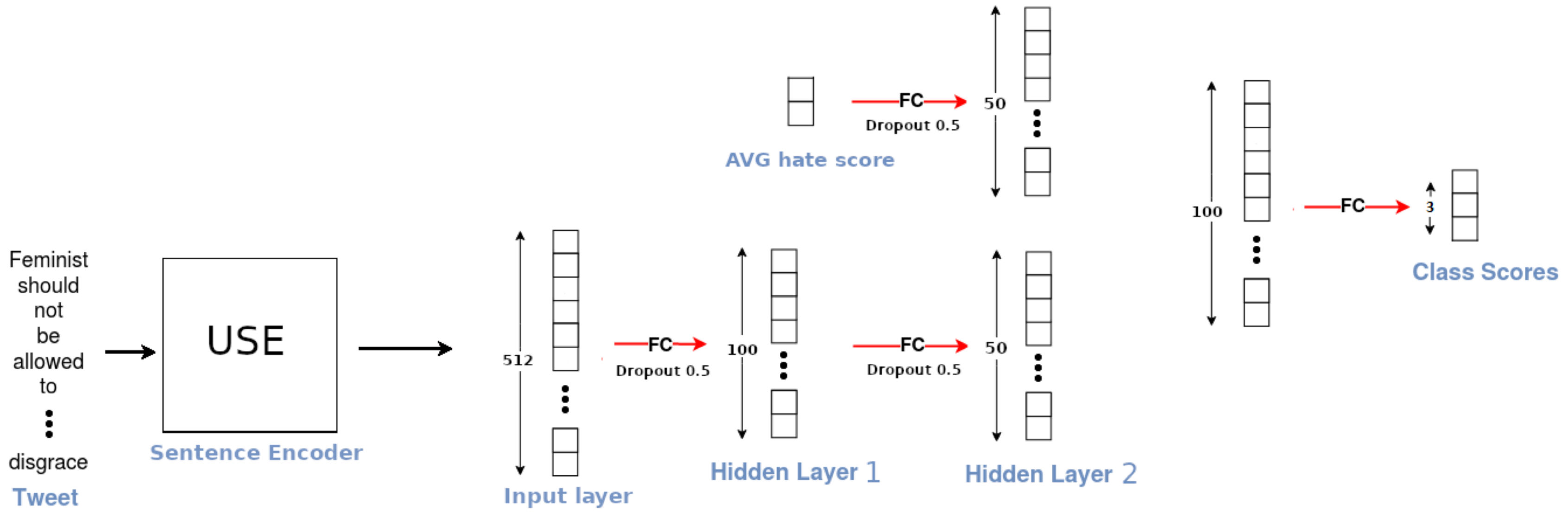
Task 5.4: Social Model

a) Social Model creation

With our social context we have 2 separate networks:

- Our text network that processes the tweet (the same as the base model from before)
- Our hate score network that basically decides how important the average hate score for the classification is

Our 2 separate networks are concatenated and one last hidden layer with 100 nodes is added.



Submitting Your Solution

As the output of sheet will be comparatively big

PLEASE HAND IN YOUR iPython notebook WITHOUT THE OUTPUT!

→ Cell → All Output → Clear

Submitting Your Solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle.

- save your expanded .ipynb iPython notebook in your working directory.
  Submit your .ipynb iPython notebook via Moodle (nothing else)

- remember: working in groups is not permitted.
  Each student must submit her own ipynb notebook!

- we check for plagiarism. Each detected case will have the consequence of 5.0 for the whole exercise grade.

- deadline: please check Moodle

Citations

[1] Waseem, Z., & Hovy, D. (2016). Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In Proceedings of the naacl student research workshop (pp. 88-93).