Table Of Contents

# Role matrices:

# Pseudo code:

This section contains the pseudocode of the function calls addressed in “‘*Are You Willing to Do Whatever It Takes?’: A Complex network model for Digital Narcissism*”

|  |
| --- |
| Algorithm used to get average number of likes per month |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **Input:** The posts from Instagram User | | | **Output:** likespm : average number of likes per month | | | **Method** The posts are sorted by dates , we aim to get the number of average likes per month | | |  | *#retrieve the number of likes* | | 1 | message = instantiate message | | 2 | likes\_count\_pm = 0 | |  | for each post in posts: | | 3 | message.contents = extract post contents here | | 4 | #every post is a stored as a dictionary | | 5 | items = get dictionary items | | 6 | if (dictionary key = ‘edges\_liked\_by’) | | 7 | count = get(‘count’) | | 8 | message.likes = count | | 9 | messages = add message to messages list | | 10 | monthly\_avg\_count = [] | |  | count\_list = [] | | 11 | for each message in messages: | | 12 | month = extract month | | 13 | for each month populate the count\_list | | 14 | avg = count\_list / messages per month | | 15 | monthly\_avg\_count = append avg | | 16 | likespm = monthly\_avg\_count | | |

|  |
| --- |
| Algorithm for hashtags |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **Input:** The sorted posts | | | **Output:** hashtagspm; frequency of posts containing hashtags | | | **Method** posts are scanned if posts have hashtag in caption or post-creators (profile) uses it in their conversation | | |  | *#retrieve the caption of the post* | | 1 | if (post.caption contains ‘#’) | | 2 | return True | | 3 | else | | 4 | convs = extract conversations from post | | 5 | for conv in convs: | | 6 | author = conv.author | | 7 | if(author name = profile name) | | 8 | text = conv.message | | 9 | if(text contains ‘#’) | | 10 | return True | | |

|  |
| --- |
| Algorithm used to detect selfies |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **Input:** The posts in sorted order | | | **Output:** Get selfies and other pictures | | | **Method** The pictures from sorted posts is obtained and analysed for selfie image | | |  | *#training* | | 1 | image = get image of a post | | 2 | classifier = knn\_classifier(training\_folder) | | 3 | diff = predict\_face(image,classifier) | | 4 | if (diff < *threshold*) | | 5 | return True | | 6 | else | | 7 | return False | | |

|  |
| --- |
| Algorithm used to get conversations and sentiments |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | **Input:** The sorted posts | | | **Output:** get sentiment of conversations | | | **Method** Posts were scanned for conversations for the sentiments | | |  | *#process conversations in each post to get sentiment* | | 1 | comments = post.get\_comments() | | 2 | for each cmt in comments | | 3 | comment = remove new lines from text | | 4 | message = message.create(name of the author, comment, date of posting) | | 5 | comment\_list = add\_message(message) | | 6 | for reply in cmt | | 7 | name = get owner of reply | | 8 | if(name == profile name) | | 9 | maveninvolved = True | | 10 | message = message.create(name, text of reply, date of posting) | | 11 | reply\_list = add\_message(message) | | 12 | if (maveninvolved) | | 13 | for each msg in reply\_list | | 14 | sent = getSetimentfromIBM\_ToneAnalyzer(msg) | | 15 | if(sent is not retrieved) | | 16 | sent = getSentimentfromVaderAnalyzer(msg) | | 17 | msg.sentiment = sent | | 18 | comments\_list.add\_conversation(reply\_list) | | |