Feature engineering – social media popularity prediction

Hot topic analysis:

- **Target_day_n**: number of articles mentioning the most popular named entity in the post

- **Comparison_metric:** compares the proportion of articles mentioning the most popular named entity on the post date versus a random day (between 7 and 14 days). The formula is:

  proportion target day/proportion random day

  (Entites without a single mention receive a proportion score of 0.001)

  E.g. : 20% of articles mentions 'Trump' on post date versus 5% on comparison date → metric = 4

- **comparison_metric_weighted:** an adjusted comparison metric weighted by engagement metrics. Each and every named entity receives an engagement score based on the total number of comments, likes and retweets the headlines received. The formula is:

  *(proportion target day/proportion random day) * sqrt(engagement for named entity on target day / engagement for named entity on random day)*

  (Engagement = total amount of likes, retweets and comments generated by headlines where the entity is featured. Entities with an engagement score of 0 on the random day receive an engagement score of 1 ; Weights lower than 1 (e.g. a named entity without any mention on the target day) receive a score of 1)

- **Hot topic:** After sorting all entities by their weighted metric score, the top 10 named entities receive a "hot topic" score of 1. Other entities are scored as 0.

If the text has several named entities, the top score is retained for all the abovementioned metrics.
If the named entity is not found in any of the headlines on target day, the scores return a missing value (na).

Example data: see the file *ners_headlines_2nov2020.csv*

functions:

connect_todb(info):

- The function expects a .csv file with the name of the database,host,port,username and password. You can find a preconfigured .csv file in *connecton_db.csv* (See Github).
- You only need to run this function once.

get_hottopic(text,connection,date,country):

- Text: string (text) to analyze
- Connection: stored connection object as defined by the connect_todb function (e.g. con = connect_todb('connection_db.csv') → con)
- Date: date of text. If None is given, Python will assume the date is *today* (!)
- Country: lowercase (e.g. 'belgium'). Relevant for filtering headlines (e.g. only from Belgian outlets)
- The function expects you to store the Dutch news corpora from Spacy in an object called "nlp", so perform *nlp = spacy.load('nl_core_news_sm')* before runing.

Moral foundations detection:

Counts the occurrence of particular moral values in a given text, as defined by the revised version of the Moral Foundation Dictionary (https://osf.io/ezn37/).

function:

morality(text)

- The text should be in English. Therefore, use the *translate(text)* function (also included) prior to running morality(text)
- The function expects you to load the English web corpora from Spacy, so run *spacy.load('en_core_web_sm')* first.

Sentiment analysis:

Returns the sentiment score (-1: very negative → 1: very positive) as produced by the Vader rule-based lexicon. The Vader lexicon is specifically designed for analyzing social media texts.
Ideally, you **don't** perform any preprocessing before analysis, since the Vader algorithm takes (1) word order (e.g. It's not bad) (2) expressive use of punctuation (e.g. !!!!!) and (3) smileys into account in their sentiment calculation. For more info, see http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

function:

sentiment(text)

- The text should be in English. Therefore, use the *translate(text)* function (also included) prior to running the sentiment function.

Toxicity analysis:

Uses the Google Perspective API to estimate the amount of 'toxicity' from a given text. The following attributes are returned (descriptions are from the perspective API page):

- Severe toxicity: A very hateful, aggressive, disrespectful comment or otherwise very likely to make a user leave a discussion or give up on sharing their perspective. This attribute is much less sensitive to more mild forms of toxicity, such as comments that include positive uses of curse words.
- Insult: Insulting, inflammatory, or negative comment towards a person or a group of people.
- Profanity: Swear words, curse words, or other obscene or profane language.

The scores represent the likelihood that a recipient perceives the text as toxic/insulting/...etc.

API limits:

There are no API limits as such, except for a 1 second latency between requests. This latency has been integrated in the toxic() function as such (sleep(1)), so you can use this function within a loop without any issue.

function:

toxic(text)

- The text should be in English. Therefore, use the translate(text) function (also included) before running the sentiment function.
- The function expects a text file in your work folder named *"perspective_api_key.txt"*, which only should contain the API key as such (no headers).

Example output:

running the demo in feature_engineering.py should give you the following output:

Text #1:
############
Text to analyze: De aanslag in Wenen maakt één ding duidelijk: de multiculturele samenleving werkt NIET en de Islam zal NOOIT onze waarden en normen respecteren!!!

Date: 2020-11-02

Hot topic metrics:

{'target_day_n': 57.0, 'comparison_metric': 51.58371040723981, 'comparison_metric_weighted': 1644.218126115884, 'hot_topic': 1.0}

Vader sentiment metric:

-0.7342

Morality metrics:

{'care': 1, 'fairness': 0, 'loyalty': 0, 'authority': 1, 'sanctity': 0}

Toxicity metrics:

{'PROFANITY': 0.21037905, 'SEVERE_TOXICITY': 0.43569338, 'INSULT': 0.3403218}

Text #2:
############
Text to analyze: Mijn medeleven gaat uit naar alle slachtoffers in Wenen. We zullen nooit zwichten voor terreur.
Date: 2020-11-03

Hot topic metrics:

{'target_day_n': 55.0, 'comparison_metric': 46.689303904923605, 'comparison_metric_weighted': 1353.1845858071633, 'hot_topic': 1.0}

Vader sentiment metric:

0.122

Morality metrics:

{'care': 2, 'fairness': 0, 'loyalty': 0, 'authority': 0, 'sanctity': 0}

Toxicity metrics:

{'PROFANITY': 0.06364202, 'INSULT': 0.1380331, 'SEVERE_TOXICITY': 0.06240886}

Text #3:
############
Text to analyze: Wenen is echt een mooie stad! Aanradertje!!!
Date: 2014-12-14

Hot topic metrics:

{'target_day_n': nan, 'comparison_metric': nan, 'comparison_metric_weighted': nan, 'hot_topic': nan}

Vader sentiment metric:

0.8139

Morality metrics:

{'care': 0, 'fairness': 0, 'loyalty': 0, 'authority': 0, 'sanctity': 0}

Toxicity metrics:

{'INSULT': 0.10104159, 'PROFANITY': 0.07284246, 'SEVERE_TOXICITY': 0.049371924}

Text #4:
############
Text to analyze: De interventie in Charleroi kwam neer op moord, er bestaat geen ander woord voor. Om van te kotsen!
Date: 2020-08-19

Hot topic metrics:

{'target_day_n': 27.0, 'comparison_metric': 25.519848771266542, 'comparison_metric_weighted': 512.3073887494213, 'hot_topic': 1.0}

Vader sentiment metric:

-0.8908

Morality metrics:

{'care': 1, 'fairness': 0, 'loyalty': 0, 'authority': 0, 'sanctity': 1}

Toxicity metrics:

{'PROFANITY': 0.3966318, 'INSULT': 0.55662084, 'SEVERE_TOXICITY': 0.30597532}