Profanity and Political check in Twitter for website feeds (Group 5)

Rose Kwamboka, Daisy Mboya, Juma Matundura, Jeremy Gachanja, Antoinette Mutugi, Bonface M. K.

August 11, 2021

- Social Media is distracting.
- Use cases where social media is useful:
  - Dashboard of major conferences/ workshops (FOSDEM)
  - Hackathons
  - As part of a website
- Is there a way to filter content to remove non-relevant content(politics/ profane sentiments)?

Example Politic content:

- Trump won/ lost the elections

Not relevant for a feed that's supposed to show Scientific data!
(Ambitious) AIM: Present a (re-usable) demo of the idea!

- Pipelines: Cleaning the data(pre-processing), storage, later retrieval
  - 2 Parts: Data retrieval– A daemon! Data Viewing– A feed!
- Scraping happens every 10 seconds.
- We use Python-twint
  - Open-source
  - No rate limitations

- No API required. Scrape anonymously!
- No upper hard limits(Twitter limits to 3.2k tweets only)
- You can scrape as far back as you want. No limits on 7-day old data!

- Remove stop-words
- Lemmatize– Reduce words to core meaning .e.g. Thieves -> Thief (important for following step)
- Check for profanity and political sentiments. We use better_profanity e.g
- How it's tied together Chain of abstraction!

```
# Get tweets(using twint)
tweets = [get_tweets(user_name=user,
                     search_term=SEARCH_TERMS,
                     limit=5) for user in USERS.

for tweet in chain(*tweets):
    # Pre-process the tweet before checking for
    if ((not redis_conn.exists(f"tweets:{tweet.i
        is_viewable_tweet(
```

```
        lemmatize(remove_stopwords(tweet.twe
    _tweet = {
        f"tweet:{tweet.id_}": {
            "id": tweet.id_,
            "tweet": tweet.tweet,
            "user_name": tweet.user_name,
            "date": tweet.date_,
            "likes": tweet.likes,
            "link": tweet.link,
            "replies": tweet.replies,
            "retweets": tweet.retweets}}
    # Actual Storage!
    with redis_conn.pipeline() as pipe:
        for tweet_id, content in _tweet.item
            pipe.hmset(tweet_id, content)
        pipe.execute()
    redis_conn.bgsave()
    redis_conn.expire(f"tweets:{tweet.id_}",
                      timedelta(days=2))
```

```
print("Done storing tweets!")
time.sleep(10000)
```

- Store data to REDIS using a daemon.
  Why Redis?
- Features for expiring data
- Light weight and open-source
- Easy to set-up
- Reduced code boiler-plate
- Data is displayed in a browser(demo)
- Same data could be displayed as part of another feed!
- Some words are anonymous like "resignation". False hits.
- How to rank tweets wrt popularity, say retweets and likes; and relevance
- Different formats of content from different spaces(like commits on public repos)!

See this link:
`https://github.com/BonfaceKilz/dsa8102-group-5-data-mining`

- Use a robust text-classifying model for ranking(see BioSentVec(26 GB!))
- Find a way to score tweets so that the most relevant data is displayed. Atm, only filtered out tweets are displayed. The list can get long!
- Fetch data from other social media platforms: Slack, IRC, Matrix, Fedi-verse, etc etc
- Aggregate data from code repositories
- Make this a library! And package it in GUIX (and Arch-Linux if usage is high)
- Port idea to other languages(like Scheme)!