Data mining of Information.dk Using Python

Eleftherios Manousakis - s141714
Sigurd Knarhøi Johannsen - s042910

***Abstract*— We describe a lightweight script that performs online article mining with sentiment analyzing, using standard components of Python. It first downloads the articles found on the main page of information.dk, then stores them in a local database, calculates their sentiment and plots them in a specific manner in a few hundred milliseconds.**

Keywords—data mining; python; sentiment analysis

I. Introduction

www.information.dk has different kinds of articles of which we choose three to compare. Using sentiment analysis we plot articles against their comments, compare article types and examine article subjects, using tags provided by www.information.dk. We hope to find correlation in article subjects and differences in sentiment among article types.

II. Data processing

Why and how do we choose the three different kinds of articles from www.information.dk?

Every day we parse the main page and have found that www.information.dk divides its news in the following categories: 'artikler', 'databloggen', 'føljeton', 'fotobloggen', 'kortfilmsbloggen', 'nyhedsblog', 'protokol' and 'telegram'.

All the news are categorized 'artikler' and then further subcategorized. However, regular articles are only subcategorized by a number, so we have chosen to name them articles and the subcategories by their subcategory only.

'databloggen', 'føljeton', 'fotobloggen', 'kortfilmsbloggen', and 'protokol' are each produced so few in numbers, that we during the relatively short extend of the course have not been able to accumulate enough, for a meaningful comparison. They have therefore been omitted from the project, leaving us with: 'artikler', 'nyhedsblog' and 'telegram'.

Articles consist of a body, comments and tags. Telegrams have a body and comments but no tags. News blogs consists of a headline, a link to another page and comments, but provides neither body nor tags.

Usually between one and four tags accompany an article. Each tag consists of one or two words or a name. By sorting articles by tags, we are somewhat able to compare articles on the same subject, allowing us to see if sentiments follow the subject.

How do we compare data?

By individually comparing the article body and its comments with a sentiment list, we get an average sentiment for each. We plot this correlation and compare the graphs for each of the three article types.

Afterwards we search our database by tags. The three most common tags are used - and articles with the same subject, according to tag, are compared.

III. Data representation

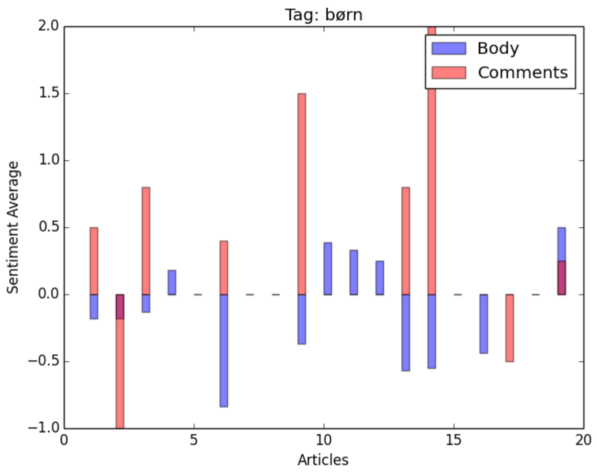

Figure 1 - Plotting by Tag

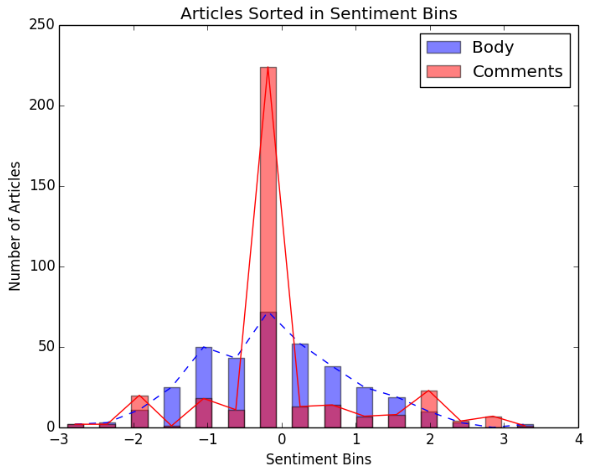

Figure 2 - Plotting body against comments

IV. Methods

After the first checks of the script are run, the main page of the news site will be parsed by using urllib [1] and BeautifulSoup[2]. The first function separates the articles based on their link length and parses them accordingly. Then, all the information is built in a dictionary and saved in the local database using couchdb [3]. CouchDB was chosen for its simplicity (handling documents as json/dictionaries) and its compatibility with Python.

From there on, the database is checked for any missing sentiments to be calculated. The calculation is done with the help of nltk (Natural Language Toolkit) [4] to tokenize the text and compare each word with the sentiment list (Nielsen2011Sentiment_afinndk).

Hereafter, the code will again connect to the news page to fetch any missing tags from the "normal" articles. We discovered that articles have tags so we decided to build on our current implementation to add those. However, this is not optimal since we parse the page again. By sorting articles by tags, we are somewhat able to compare articles on the same subject, allowing us to see if sentiments follow subject.

In the end, with the use of the plotting library matplolib [5] which produces publication quality figures, we are plotting our results. Three plots with tags will appear and after that three more by type. To choose the most common tags we are using the collections high performance container. [6]

Furthermore, the code was thoroughly checked with flake8 [7], which is a wrapper around tools like PyFlakes and pep8.

Last but certainly not least, the documentation was created by using Sphinx [8], which is a tool that makes it easy to create intelligent and beautiful documentation.

V. Discussion

The following part is based on plotting we have already made.

Articles have bodies with a wide range of average sentiments, they are slightly more negative. The negativity of articles may be contributed by the fact that bad news sell better. Comments for articles are surprisingly, primarily neutral. Also surprising is the fact that they are slightly more positive than negative; there are even more positive fanatics than negative.

Telegrams have a wider range of average sentiment, than that of articles. The structure however follows the same pattern. The body have a wide range, but are again, slightly negative. Comments for telegrams are very neutral.

News blogs do not contain bodies and we choose not to parse the link also in this report. However the abstract sections are slightly on the positive side. The comments are practically neutral, with a few fanatic comments among them.

Film subject bodies are slightly on the positive side, whereas their comments are usually neutral, with a few very positive ones where the bodies are a lot less positive. The spikes of difference between positive and negative could be explained by the fact that we're most likely talking about opinions for films. So when one critic praises a movie, another critic will criticize it.

Kids' subject bodies are on the negative side and their comments are on the positive side. It seems only natural that negative news about kids would call for caring comments.

Young subject bodies are usually negative. The comments are usually positive; however we also see a few negative spikes. This difference from kids is likely caused by the teenagers. People tend not to praise teenagers behavior. However, since kids and young are often treated as one subject, it would only make sense, that the results of the two are somewhat similar.

The difference in comment neutrality between articles and telegrams could well be explained by the fact, that whereas articles are supposed to catch the readers' interest and are therefore written with an angle – telegrams are merely supposed to deliver facts.

Error sources

Tags sometimes consist of two words; however we only read tags as single words. This can affect the meaning of a tag, however we do not take meaning of the tags into account and it will therefore not pose a problem for the purpose of our project. Even when we present two articles with two different tags as articles of the exact same subject - the two worded tag did in fact include the subject tag. So we end up only with a slight error margin. The same approach is used for the sentiment analysis of the body and the comments of articles.

VI. Conclusion

It would be interesting to compare these results with that of gossip magazines.

References

[1] Urllib https://docs.python.org/2/library/urllib.html
[2] BeautifulSoup http://www.crummy.com/software/BeautifulSoup/
[3] CouchDB https://py-couchdb.readthedocs.org/en/latest/
[4] Nltk http://www.nltk.org/
[5] Matplotlib http://matplotlib.org/
[6] Collections https://docs.python.org/2/library/collections.html
[7] Flake8 https://pypi.python.org/pypi/flake8
[8] Sphinx http://sphinx-doc.org/