creditrnews: News Analyzer for Credit Scoring

Capstone Project Summary

Contents

Introduction

Operating in more than 40 countries around the world, IBM Global Financing is the world's largest Information Technology (IT) financing company. IBM Global Financing's core business includes: Client financing, Commercial Financing and Global Asset Recovery Services (GARS) .One of the major functions in IGF Budapest center is the risk analysis of client financing and commercial financing deals.

Situation

Credit risk assessment of a company is a very complex problem. Besides overviewing the latest financial accounts, analysts also look for external sources of information, such as searching for relevant news articles. Information derived from the news can be a good indicator of the company's performance, because (1) it includes up-to-date facts, (2) comes from diverse external sources; and (3) includes industry specific analysis.

Currently the news search in the IBM Global Financing Credit Team is done manually: analysts skim through all the articles suggested by the search engine and spend around an hour on daily basis to be well informed.

Task

A news mining tool, which would conduct the basic sentiment analysis and report only the most important information from the articles would save a lot of time and make the assessment process more efficient. Therefore, my goal was to create a project, outcome of which (1)will be relevant for IBM, the project (2) will be easy to use, reproduce, edit, contribute and maintain.

Action

As I am not disclosing any confidential commercial information, I created a public R-package on GitHub (https://github.com/thegrigorian/creditrnews), (1) where anyone can download and install locally, and (2) contribute to make it more useful. The package can be installed running the following lines:

```
devtools::install_github("thegrigorian/creditrnews", dependencies = TRUE)
library(creditrnews)
```

Table 1: Functions from creditrnews.

get_links()	Links for the top news search results.
get_articles()	Articles from the top news search results.
get_sentiment_plot()	Sentiments across the length of the article.
get_sentiment_report()	Sentiment report for every article.
get_wordcloud()	Word cloud for the sentiment tagged words.
get_words()	Most common words by their frequency.
get_words_tfidf()	Most common words with tf-idf statistic
which_article()	Must -read articles based on extreme negative sentiments.

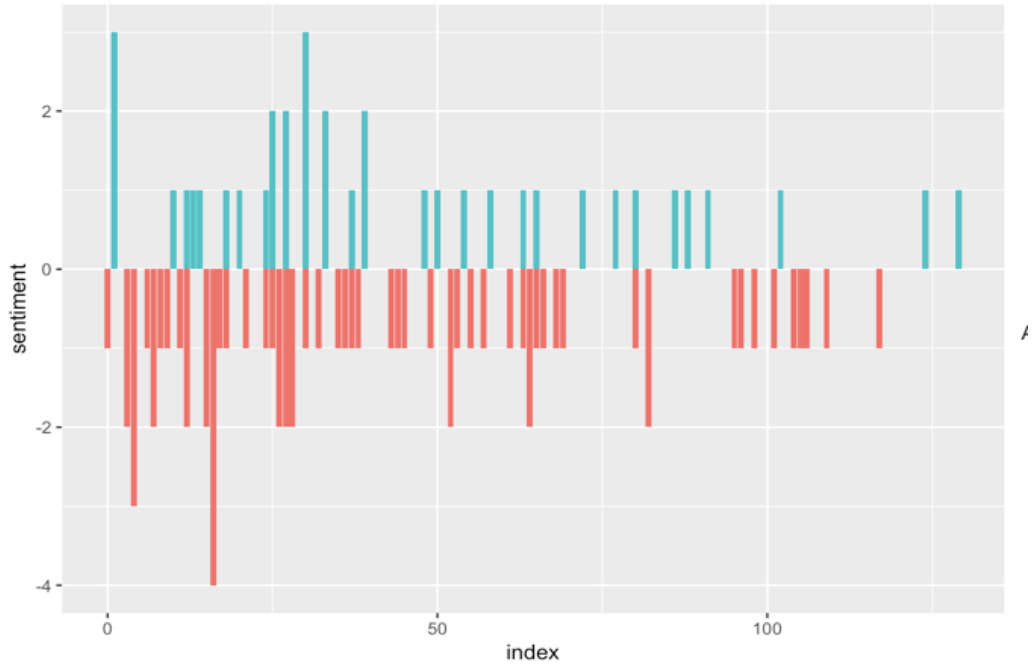
Results

Once I had the results I shared with the team to see their opinion regarding the use of it.
One of the main advantages of this package is being one-click away from the relevant news. Using the get_links() function, company name being the input parameter, it does a focused search, based on a keyword, time and relevance bringing back the top ten results. This function

is used for visiting the news article links one-by-one and scraping them. Finally, the list of ten articles is converted into a one-token per-row format using the tidytext package.

A data.table dataset containing a filtered version of Loughran & McDonald's (2016) positive/negative financial word list was used as sentiment lookup values.

Figure 1: Sentiment plot for Netflix with Loughran/McDonald lexicon



In the example above, the sentiments for the articles about Netflix are visualized: the frequency of a term is plotted against length of the article. Another great way to visualize the sentiments is using word-clouds or scoring. Both are available in the package, and can be accessed using the commands: **get_wordcloud()** and **get_sentiment_report()**.

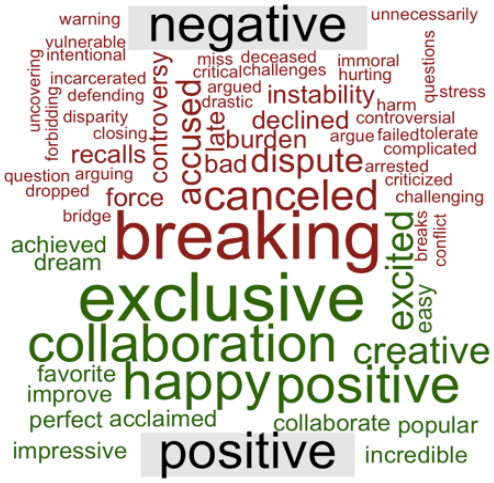


Figure 2: Word-cloud for Netflix

The size of a word's text in Figure 3 is in proportion to its frequency within its sentiment. We can use this visualization to see the most important positive and negative words, but the sizes of the words are not comparable across sentiments. For example, we can see that something has been canceled on Netflix or that there is a problem with accusation, etc. Although those sentiments don't tell much about the company's performance directly, they provide some insight on where exactly the analyst should do his research.

Should the analyst be interested in getting sentiments score the function **get_sentiment_report()** can be called.

If the analyst is not sure which article is worth the read, **which_article()** function will outscore all the must-read articles based on the sentiment report.

	article	positive	negative	sentiment	wordcount	sensitivity
	<int>	<int>	<int>	<int>	<int>	<chr>
1	1	5	-6	-1	438	-0.23%
2	2	0	-9	-9	176	-5.11%
3	4	5	-14	-9	600	-1.5%
4	5	3	-6	-3	204	-1.47%
5	6	5	-8	-3	391	-0.77%
6	7	3	-7	-4	236	-1.69%
7	8	7	-4	3	521	0.58%
8	9	8	-10	-2	652	-0.31%
9	10	3	-4	-1	204	-0.49%

Table 2: Sentiment report for Netflix

Another approach is to look at a term's inverse document frequency (idf), which decreases the weight for commonly used words and increases the weight for words that are not used very much. This can be combined with term frequency to calculate a term's tf-idf (the two quantities multiplied together), the frequency of a term adjusted for how rarely it is used. The statistic tf-idf is intended to measure how important a word is in a collection of articles.

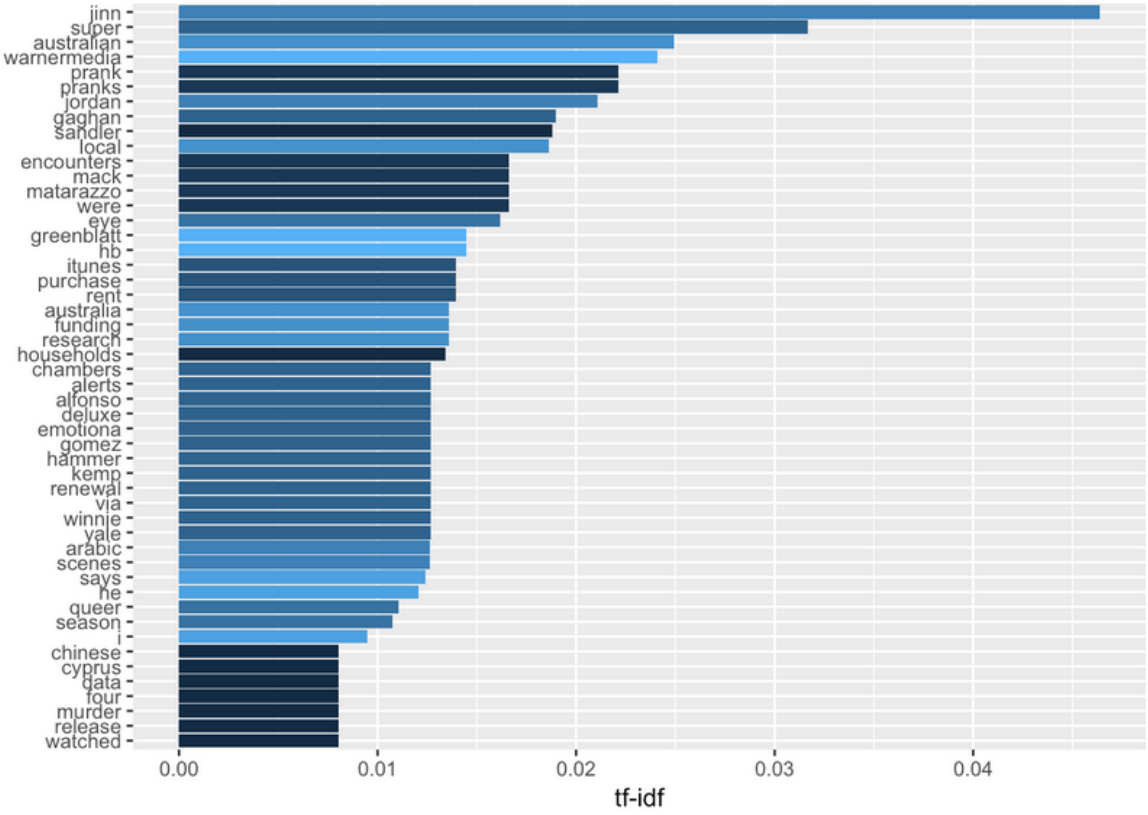


Figure 3: tf-idf for Netflix

Conclusion

As discussed with the team, the idea of automating the news search will be very useful for the company. The results from the functions in this package are relevant and easy to use. Improvements would include (1) an integration of a user-friendly interface and (2) industry news analysis functions to have a more complex view about the company.