Datascience Fundamentals with R - Reddit

Group 7

December 5, 2017

Goal

Analyze reddit news posts because they give us a lot of information about what is important to society at a particular time. We use reddit because it is one of the most popular websites in the world and has a very active user community, with 1.6 billion users. The news subreddit alone has 70,000 posts in a month.

We believe that the posts of the top authors (authors that have the most posts) are representative of the community since their content should be varied. Furthermore, given the variety of news posts and the sentiments associated with them (there is always good & bad news) we think that there should be a correlation between good news and positive sentiments and bad news with negative sentiments.

We can assume that a particular author may lean towards a more positive or negative sentiment - we thus assign a variable `author_score` to represent how positive or negative a particular author is. This measure is calculaetd as the *summed sentiments of all posts of this particular author* , the question now becomes:

For each post, can we determine whether their post will be well liked, given the author's general sentiment?. Our hypothesis is that posts with a positive sentiment will be received better than those with a negative one, so authors that have a generally positive sentiment would have much more upvotes (positive correlation between `author_score` and `like_score`)

Data

We decided to use one month of Reddit News data. We picked December 2016 for this as it was just after the US elections and that enough time has passed for us to (possibly) notice the effects of news that came out at that time.

Reddit news data usually comes in the form of a title and a link, the title is written by the user and links to the news website's page. It sometimes has thumbnails.

Reddit post data is available on google's bigquery database, it is saved in `news_2016_12.csv`. Following are the most relevant columns for our analysis:

- `time_created` (UTC timestamp) - when was the post created
- `author` - username of user that posted
- `domain` - which domain did the news come from?
- `url` - Specific URL of the news post
- `score` : upvotes - downvotes (renamed as `like_score`)
- `upvotes` : how many "likes" the post received
- `downvotes` : howmany "dislikes" the post received
- `title` : user-created title of the post

Due to an issue with the API, no downvote data is given to us. Thus the `score` is equal to the `upvotes` of a post. We will be using the `score` field moving forward.

Furthermore, we thought it would be interesting to compare the difference between the user-generated title and the Actual title posted by the news agency. For this, a (scrapy)[https://scrapy.org/] spider was created to crawl all the (cleaned) URLs and retrieve the title. We think this was rather successful since it retreived `24,045` titles form about `31,713` cleaned posts. This is saved as `titles.csv`.

Task

Structure of the project

1. Cleaning (in common with the other group)
2. Descriptive analysis of the data set: Exploring the data
3. Introduction of sentiment analysis
4. Prediction of reach with Perceptron
5. Convolutional network to predict subreddit classification

Due to the size and variety of elements in our dataset, we will apply a variety of methods to extract information, and may use external data to improve our analysis. However, this also implies that the performance measures of our algorithms will vary with the specific relationship under examination.

1. Cleaning `[1_clean.r]`

Everything related to cleaning the original dataset is defined in `clean.r`. Essentially, the script:

- saves `utc_created` as a POSIXct variable
- removes special characters from titles
- removes non-english titles, which halves our original dataset.

Getting relevant data - `[05_Classification_Data.R]`:

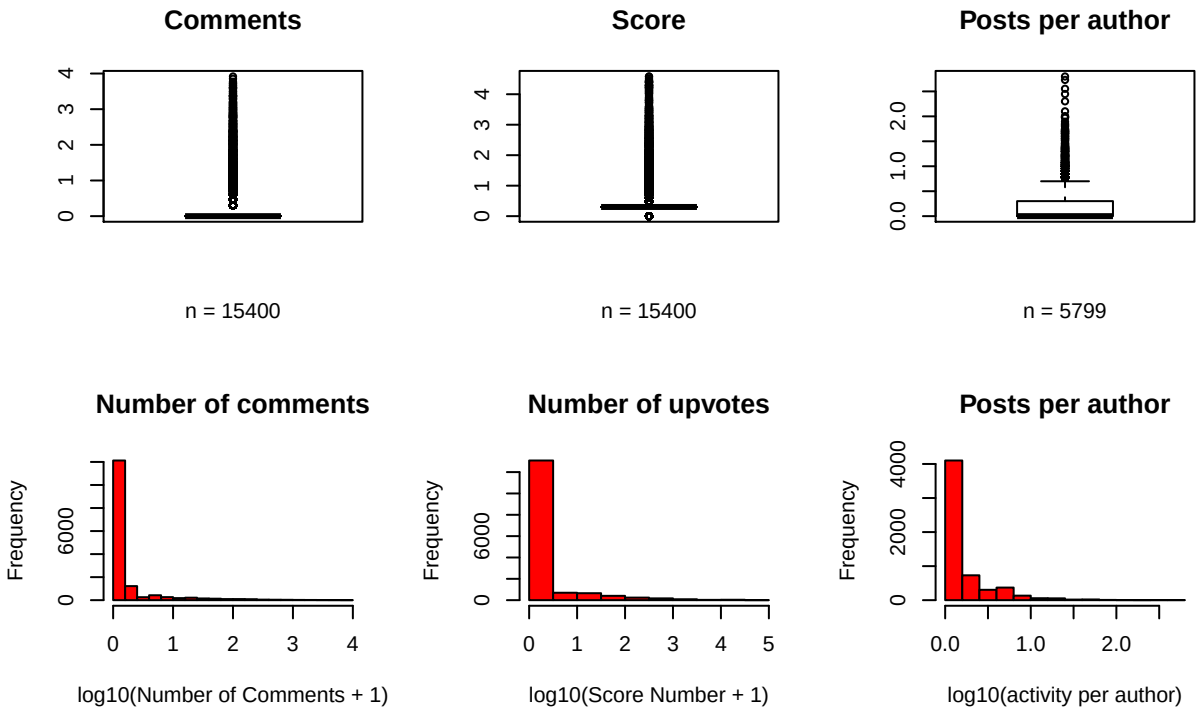
We remove all authors with the value `[deleted]`, because this is value given to users who have deleted their accounts, which mean that we become unable to different specific users.

After these two operations, our dataset is reduced to `3946` rows compared to the originial `31,713`, and this is further reduced to `394` upon generating the `author_score`

2. Descriptive analysis of the data set: Exploring the data `[2_analysis.r]`

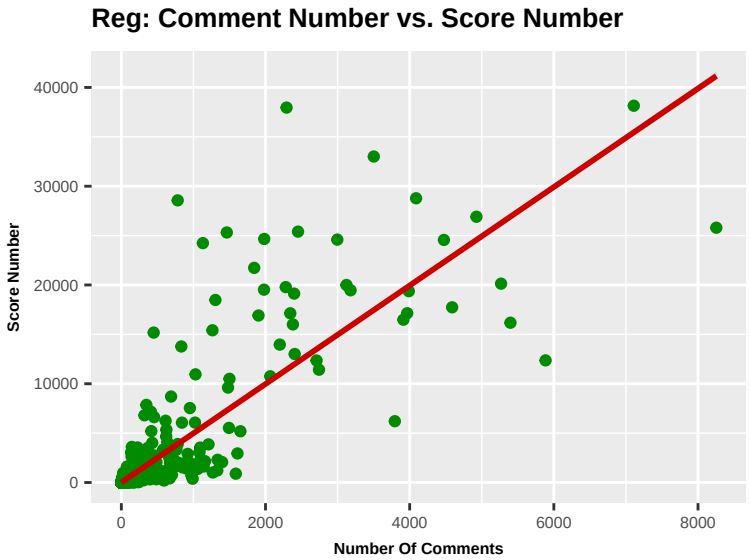
2.1 Analysis of comments, upvotes and posts per author:

Since the data is massively skewed, we set the proportion as log10.



Interpretation: the plots show a massive skew even though the data's proportion is log10. All ploted variables are right side skewed which shows that the sample consists many low discrete values and marginal number of outliers. This is very important since it has to be taken into consideration in order to adapt the data to get significant results later in this project depending on the models.

2.2 Regression: Analyze how activity, here defined as number of comments, influences score.



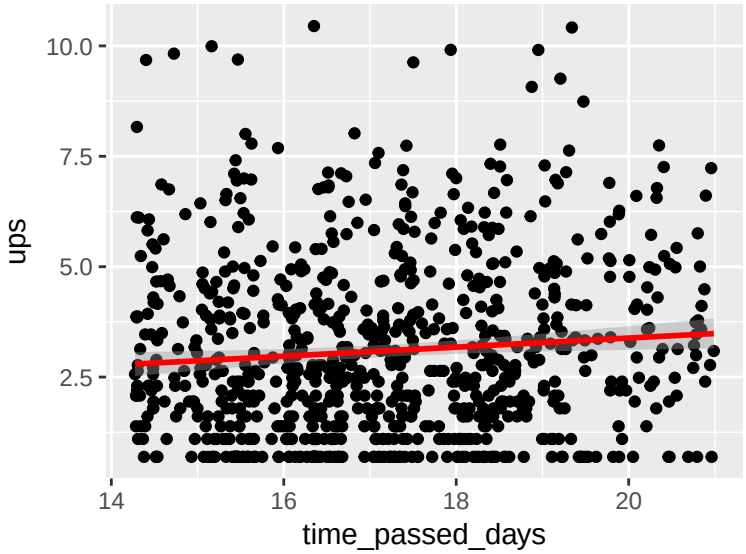
Interpretation: Here, we can see that an additional comment yields on average a 4.9 increase in the score number. The result is statistically significant given the low p-value of $< 2.2e\text{-}16$.

2.3 Regression determining the effect of features on up votes and number of comments

```
# Tatiana's regression
```

2.4 Regression: Analyze how the time passed influences the number of upvotes.

Explanations: here, we log the number of upvotes because the distance between the different number of upvotes is too high and we don't get any significant result. We also select only the posts which have more than one upvotes and focus on posts which are 7 days old because the upvotes usually happens in the next days following the posting date.



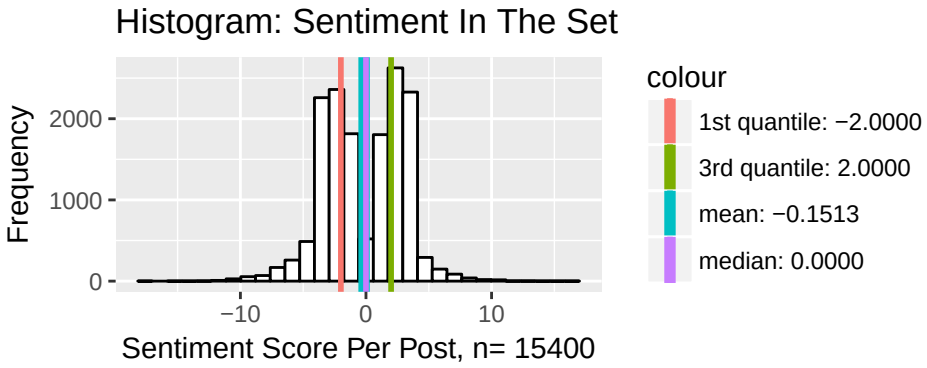
Interpretation: For every additional day, a post gets an additional 0.09013 upvote. The result is statistically significant given the p-value < 0.02902.

3. Sentiment analysis

Natural Language Processing with `tidytext`:

In this part, we will use 2 different libraries: NRC and Afinn in order to compute the sentiment of each word of the data set and ultimately be able to compute the sentiment of the different titles. These 2 libraries will be used for two very different purposes, which will be illustrated in the following examples.

3.1 Histogram of sentiment analysis of the data set using AFINN OR NRC?? alex part

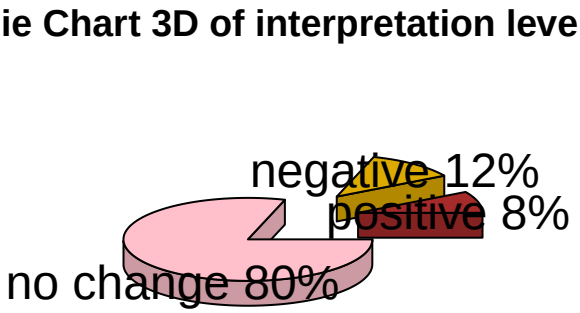


Interpretation:

3.2 Analyze posting behavior with sentiment analysis using Afinn:

In this part, the goal is to analyze whether people interpret news in a positive or negative way when reposting a news. This is done by analyzing the difference of sentiment level between the source title (title displayed on a news website such as on times.com) and the actual title of the reddit user's post.

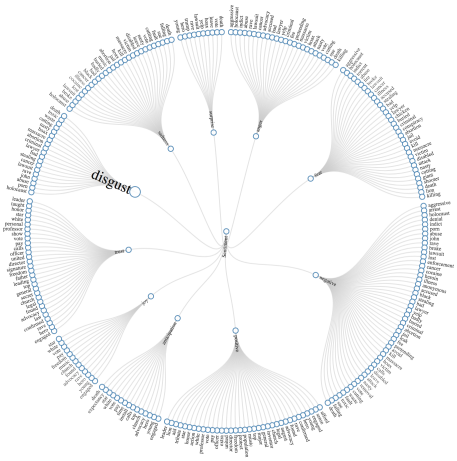
Approach: to do this, source titles have been retrieved with the help of their URL, then the sentiment score on both title lists has been calculated with the help of the Afinn lexicon assigning a weight between `-5 < weight < 5` for every word. We chose Afinn because it ranks the sentiment on a scala, making it easier to compare. Finally, the difference has been computed which enables to get some insights as seen in the following graphics: the histogram shows the proportions of posts changed, positive or negative while the 3D chart illustrates the percentage.



Interpretation: As observed on the pie chart, most Reddit users (4 out of 5 users) tend to repost exactly the same title that comes from the source title, while 12% of the users in this data set repost it in a negative way (when different in sentiment is negative) and the rest (9%) repost it in a positive way.

2.3 Estimation of proportion of each sentiment in reddit titles: how to illustrate sentiment through : Radial dendrogram

It's interactive. Click here to try it out online.



2.4. How the sentiments are related : network diagrame

```
# Tatiana's dendogram
```

4. Perceptron classification:

4.1. Machine learning with Perceptron: prediction of a successful post based on 3 features

Feature 1: sentiment of the post (afinn dictionnary) Feature 2: length of the post (number of characters) Feature 3: Posting time (within a day)

In this part, the three features illustrated above will be used to predict whether a post is successful or not. First, we define a successful post as a post getting at least 2 likes.
This splits the data in X% successful posts and X% not successful posts. Then, we prepare the data by assigning every successful post the category 1 and every non successful post the category -1. After this, we keep only the posting time in hours as well as the sentiment of each title. The Training data is then split in 70% training and 30% testing.

Let's have a look at the training

```
# histogram of positive and negative + choose the 3D pie chart (next to each other?)
```

Performance of the algorithm

```
# histogram of positive and negative + choose the 3D pie chart (next to each other?)
```

Discussion on the implementation