Text analysis of Trump tweets

Anton Antonov
MathematicaForPrediction at GitHub
MathematicaVsR project at GitHub
November, 2016

Introduction

This Mathematica notebook is part of the MathematicaVsR at GitHub project TextAnalysisOfTrumpTweets. The notebook follows and extends the exposition and analysis of the R-based blog post “Text analysis of Trump’s tweets confirms he writes only the (angrier) Android half” by David Robinson at VarianceExplained.org; see [1].

The blog post [1] links to several sources that claim that Donald Trump tweets from his Android phone and his campaign staff tweets from an iPhone. The blog post [1] examines this hypothesis in a quantitative way (using various R packages.)

The hypothesis in question is well summarized with the tweet:

Every non-hyperbolic tweet is from iPhone (his staff).
Every hyperbolic tweet is from Android (from him). pic.twitter.com/GWr6D8h5ed
— Todd Vaziri (@tvaziri) August 6, 2016

In this Mathematica notebook (document, post) we are going to use the data provided in [1] and confirm the hypothesis with same approaches as in [1] but using alternative algorithms. For example, the section “Breakdown by sentiments” contains Bayesian statistics visualized with mosaic plots for device-vs-sentiment and device-vs-sentiment-vs-weekday -- those kind of statistics and the related time tags are not used in [1].

Concrete steps

1. Data ingestion

   - The blog post [1] shows how to do in R the ingestion of Twitter data of Donald Trump messages.

   - That can be done in Mathematica too using the built-in function `ServiceConnect`, but that is not necessary since [1] provides a link to the ingested data used [1]:

```
load(url("http://varianceexplained.org/files/trump_tweets_df.rda"))
```

   - Which leads to the ingesting of an R data frame in this notebook using RLink.

2. Adding tags

   - We have extract device tags for the messages : each message is associated with one of the tags "Android", "iPad", or "iPhone".

   - Using the messages time-stamps each message is associated with time tags like month, hour, weekday, etc.

3. Classification into sentiment and Facebook topics

- Using the built-in classifiers of Mathematica each tweet message is associated with a sentiment tag and a Facebook topic tag.

- In [1] the results of this step are derived in several stages.

4. Device-word association rules

- Using Association rule learning device tags are associated with words in the tweets.

- In this notebook these associations rules are not needed for the sentiment analysis (because of the built-in classifiers).

- This association rule mining is done mostly to support and extend the text analysis in [1] and, of course, for comparison purposes. (See the corresponding R-part.)

In [1] the sentiments are derived from computed device-word associations, so the order of steps is 1-2-4-3. In Mathematica we do not need the 4th step in order to get the sentiments in the 3d step.

Load packages

These commands load the packages [2,3,4] used in this notebook:

In[1]:=
```
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   MathematicaForPredictionUtilities.m"]
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   MosaicPlot.m"]
Import[
 "https://raw.githubusercontent.com/antononcube/MathematicaForPrediction/master/
   AprioriAlgorithm.m"]
```

Getting Twitter messages

The blog post [1] shows how the ingestion of Twitter data of Donald Trump messages is done in R. This can be done in Mathematica too using the built-in function `ServiceConnect`, but since [1] provides a link to the ingested data used [1] we going use it through RLink.

Using ServiceConnect

This sub-section has a cursory list of commands for setting-up ingestion of Twitter messages with `ServiceConntect`.

```
twitterConn = ServiceConnect["Twitter", "New"]
```

```
ServiceObject[ Twitter
               Not Connected ]
```

"Thank you for authorizing WolframConnector. Your service object with id XXXX-XXXX-XXXX-XXXX is now authenticated. Return to the Wolfram Language to use it."

```
twitterConn["UserData", "Username" → "realDonaldTrump"]
```

```
{⟨|ID → 25 073 877, Name → Donald J. Trump,
  ScreenName → realDonaldTrump, Location → New York, NY,
  FollowersCount → 15 020 674, FriendsCount → 41, FavouritesCount → 46|⟩}
```

Directly from the Variance Explained blog post

This subsection shows the ingestion of an R data frame in Mathematica using RLink.

```
In[4]:= Needs["RLink`"]
        InstallR[]
```

Ingest the data frame from [1] in the R session set-up above:

```
In[6]:= REvaluate["load(url('http://varianceexplained.org/files/trump_tweets_df.rda'))"]

Out[6]= {trump_tweets_df}
```

Get the data frame object:

```
In[7]:= trumpTweetsDF = REvaluate["trump_tweets_df"];
```

Extract column names:

```
In[8]:= colNames = "names" /. trumpTweetsDF[[2, 1]]

Out[8]= {text, favorited, favoriteCount, replyToSN, created,
         truncated, replyToSID, id, replyToUID, statusSource, screenName,
         retweetCount, isRetweet, retweeted, longitude, latitude}
```

Make an `Association` object in order to have more clear code below:

```
In[9]:= aColNames = AssociationThread[colNames -> Range[Length[colNames]]]

Out[9]= ⟨|text → 1, favorited → 2, favoriteCount → 3, replyToSN → 4, created → 5, truncated → 6,
         replyToSID → 7, id → 8, replyToUID → 9, statusSource → 10, screenName → 11,
         retweetCount → 12, isRetweet → 13, retweeted → 14, longitude → 15, latitude → 16|⟩
```

Verify columns number and columns lengths:

```
In[10]:= Length[trumpTweetsDF[[1]]]

Out[10]= 16
```

```
In[11]:= Length /@ trumpTweetsDF[[1]]

Out[11]= {1512, 1512, 1512, 1512, 2, 1512, 1512,
          1512, 1512, 1512, 1512, 1512, 1512, 1512, 1512, 1512}
```

One of the columns is given as a vector object -- extract the values for that column in order to get a matrix/table form of the data:

```
In[12]:= trumpTweetsDF[[1, 5]] = trumpTweetsDF[[1, 5, 1]];
```

An R data frame is a list of columns. Using `Transpose` we get the usual Mathematica list of records form:

In[13]:=
```
trumpTweetsTbl = Transpose[trumpTweetsDF⟦1⟧];
```

Visualize the obtained, ingested data frame:

In[14]:=
```
Magnify[#, 0.6] &@
 TableForm[trumpTweetsTbl⟦1 ;; 12, All⟧, TableHeadings → {None, colNames}]
```

Out[14]=

text
My economic policy speech will be carried live at 12:15 P.M. Enjoy!
Join me in Fayetteville, North Carolina tomorrow evening at 6pm. Tickets now available at: https://t.co/Z80d4MYIg8
#ICYMI: "Will Media Apologize to Trump?" https://t.co/ia7rKBmioA
Michael Morell, the lightweight former Acting Director of C.I.A., and a man who has made serious bad calls, is a total Clinton flunky!
The media is going crazy. They totally distort so many things on purpose. Crimea, nuclear, "the baby" and so much more. Very dishonest!
I see where Mayor Stephanie Rawlings-Blake of Baltimore is pushing Crooked hard. Look at the job she has done in Baltimore. She is a joke!
Thank you Windham, New Hampshire! #TrumpPence16 #MAGA https://t.co/ZL4Q01Q49s
.@Larry_Kudlow - 'Donald Trump Is the middle-class growth candidate' https://t.co/YbqkhWNm0g
I am not just running against Crooked Hillary Clinton, I am running against the very dishonest and totally biased media - but I will win!
#CrookedHillary is not fit to be our next president! #TrumpPence16 https://t.co/I0zJO2sZKk
Heading to New Hampshire - will be talking about Hillary saying her brain SHORT CIRCUITED, and other things!
Anybody whose mind "SHORT CIRCUITS" is not fit to be our president! Look up the word "BRAINWASHED."

Data wrangling -- extracting source devices and adding time tags

Sources

As done in the blog post [1] we project the data to four columns and we modify the values of the column "statusSource" to have the device name.

In[15]:=
```
trumpTweetsTbl = Transpose[trumpTweetsDF⟦1⟧]⟦
   All, aColNames /@ {"id", "statusSource", "text", "created"}⟧;
```

Examining how the second column looks like:

In[16]:=
```
RecordsSummary@trumpTweetsTbl⟦All, 2⟧
```

Out[16]=
```
1 column 1
{ <a href="http://twitter.com/download/android"
    rel="nofollow">Twitter for Android</a>                          762
  <a href="http://twitter.com/download/iphone"
    rel="nofollow">Twitter for iPhone</a>                           628
  <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a> 120
  <a href="http://instagram.com" rel="nofollow">Instagram</a>        1
  <a href="http://twitter.com/#!/download/ipad"
    rel="nofollow">Twitter for iPad</a>                             1
```

we can just the take the device names using string pattern matching:

In[17]:=
```
sourceDevices = StringCases[trumpTweetsTbl[[All, 2]],
   RegularExpression["Twitter for (.*?)<"] :> "$1"];
Tally[sourceDevices]
```

Out[18]=
```
{{{Android}, 762}, {{iPhone}, 628}, {{}, 121}, {{iPad}, 1}}
```

Looking at the `Tally` result we see that there are strings that do not match the device source pattern -- we simply remove the corresponding rows:

In[19]:=
```
trumpTweetsTbl = Pick[trumpTweetsTbl, Length[#] > 0 & /@ sourceDevices];
```

and replace the values of the “statusSource” column with the values of `sourceDevices`:

In[20]:=
```
trumpTweetsTbl[[All, 2]] = Flatten[sourceDevices];
```

Time tags

The addition time tag (like “hour of the day” and “weekday”) is not necessary for the sentiment analysis over devices but would provide the time axis for useful or interesting statistics.

Next we convert the creation times in the column “created”:

In[21]:=
```
RecordsSummary@trumpTweetsTbl[[All, 4]]
```

Out[21]=

1 column 1	
Min	1.45012×10^9
1st Qu	1.4585×10^9
Mean	1.46299×10^9
Median	1.46353×10^9
3rd Qu	1.46764×10^9
Max	1.47067×10^9

to a list of date lists using Unix time conversion:

In[22]:=
```
dateLists = DateList[FromUnixTime[#]] & /@ trumpTweetsTbl[[All, 4]];
Shallow@dateLists
```

Out[23]//Shallow=
```
{{2016, 8, 8, 10, 20, 44.}, {2016, 8, 8, 8, 28, 20.},
 {2016, 8, 7, 19, 5, 54.}, {2016, 8, 7, 18, 9, 8.}, {2016, 8, 7, 16, 31, 46.},
 {2016, 8, 7, 8, 49, 29.}, {2016, 8, 6, 21, 19, 37.}, {2016, 8, 6, 21, 3, 39.},
 {2016, 8, 6, 20, 53, 45.}, {2016, 8, 6, 15, 4, 8.}, <<1381>>}
```

We join the rows of the tweets with the rows of the time tag matrix `dateLists`:

In[24]:=
```
trumpTweetsTbl = MapThread[Join, {trumpTweetsTbl, dateLists}];
```

In a similar fashion as date lists let us also add weekdays:

In[25]:=
```
weekdays = DateString[FromUnixTime[#], "DayName"] & /@ trumpTweetsTbl[[All, 4]];
Shallow@weekdays
```

Out[26]//Shallow=
```
{Monday, Monday, Sunday, Sunday, Sunday,
 Sunday, Saturday, Saturday, Saturday, Saturday, <<1381>>}
```

In[27]:=
```
trumpTweetsTbl = MapThread[Append, {trumpTweetsTbl, weekdays}];
```

Here we create an `Association` object for easy access to the columns of the data:

In[28]:=
```
aTrumpTweetsTblColNames =
  AssociationThread[{"id", "source", "text", "created", "year", "month", "day",
     "hour", "minute", "second", "weekday"} → Range[Dimensions[trumpTweetsTbl][[2]]]]
```

Out[28]=
```
<|id → 1, source → 2, text → 3, created → 4, year → 5,
  month → 6, day → 7, hour → 8, minute → 9, second → 10, weekday → 11|>
```

Here is a summary of the data:

In[29]:=
```
Magnify[#, 0.6] &@RecordsSummary[trumpTweetsTbl, Keys[aTrumpTweetsTblColNames]]
```

Out[29]=

1 id	
676494179216805888	1
676509769562251264	1
678442470720577537	1
678446032599040001	1
678490367285678081	1
680492103722479616	1
(Other)	1385

2 source	
Android	762
iPhone	628
iPad	1

3 text

```
MAKE AMERICA GREAT AGAIN!
"@007cigarjoe: #MakeAmericaGreatAgain #Trump2016 @realDonaldTrump IS THE ONLY DEAL !!! https://t.c
"@1lion: brilliant 3 word response to Hillary's 'I'm
  With You'  slogan https://t.co/dJL71jwK0g @realDonaldTrump https://t.co/Audi8h85qu"
"@1sonny12: @KSmith233035 @mitchellvii  FLORIDIANS ARE
  UPSET BECAUSE RUBIO DID NOT DO WHAT HE PROMISED ONCE HE WAS ELECTED! VOTE TRUMP"
2004 VIDEO:
Pocahontas describing Crooked Hillary Clinton as a Corporate Donor Puppet. Time for change! #Trump
https://t.co/rZ1MqUzpKU
"@60Minutes: DonaldTrump and his running mate @Mike_Pence
  to appear on #60Minutes in first joint interview. CBS https://t.co/lZH7qw9qmu"
(Other)
```

6 month	
Min	1
1st Qu	3
Mean	$\frac{6844}{1391}$
Median	5
3rd Qu	7
Max	12

7 day	
Min	1
1st Qu	8
Mean	$\frac{22238}{1391}$
Median	16
3rd Qu	24
Max	31

8 hour	
Min	0
1st Qu	8
Mean	$\frac{18692}{1391}$
Median	14
3rd Qu	18
Max	23

9 minute	
Min	0
1st Qu	14
Median	28
Mean	$\frac{40256}{1391}$
3rd Qu	$\frac{179}{4}$
Max	59

10 second	
Min	0.
1st Qu	16.
Median	29.
Mean	29.7764
3rd Qu	45.
Max	59.

11 weekday	
Tuesday	267
Wednesday	235
Friday	221
Monday	190
Saturday	176
Thursday	153
Sunday	149

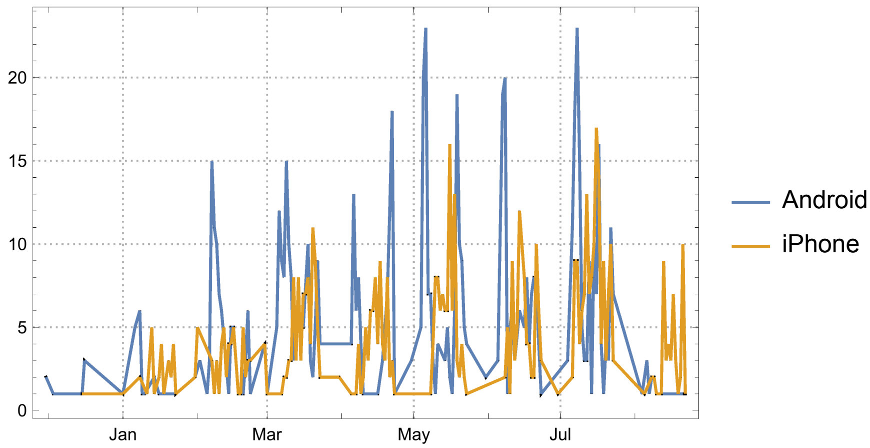
Time series and time related distributions

In this section we simply derive the time series discussed in [1]. The statistics in this section are not needed to do the sentiment analysis but they provide additional insight into the tweeting patterns in the data.

First, we can plot the time series of number of tweets per hour within the time span of the data:

In[30]:=
```
DateListPlot[Map[Tally[Cases[trumpTweetsTbl, {__, #, __}][[
      All, aTrumpTweetsTblColNames /@ {"year", "month", "hour"}]]] &,
  {"Android", "iPhone"}], PlotRange → All, PlotTheme → "Detailed",
 PlotLegends → {"Android", "iPhone"}]
```

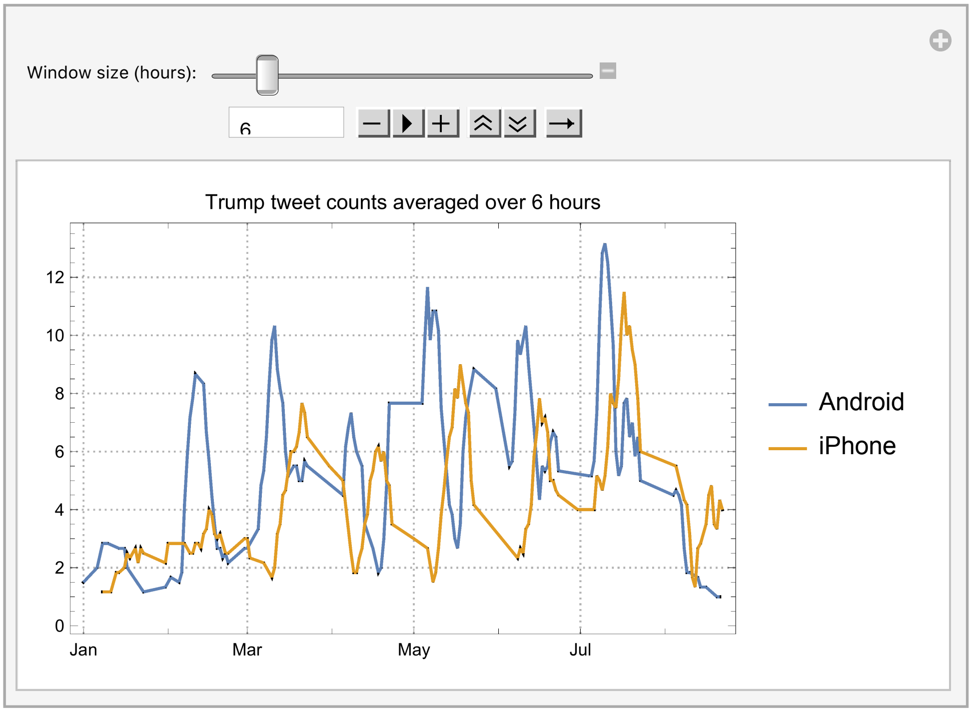
Out[30]=



The time series plot above can be modified into a more informative one using moving average (with `TimeSeries` and `Manipulate`):

In[31]:=
```
Manipulate[DateListPlot[
  Map[MovingAverage[#, w] &@TimeSeries@Tally[Cases[trumpTweetsTbl, {__, #, __}][[
       All, aTrumpTweetsTblColNames /@ {"year", "month", "hour"}]]] &,
   {"Android", "iPhone"}], PlotRange → All, PlotTheme → "Detailed",
  PlotLabel → Row[{"Trump tweet counts averaged over ", w, " hours"}],
  PlotLegends → {"Android", "iPhone"}],
 {{w, 6, "Window size (hours):"}, 1, 48, 1, Appearance → {"Open"}}, Deployed → True]
```

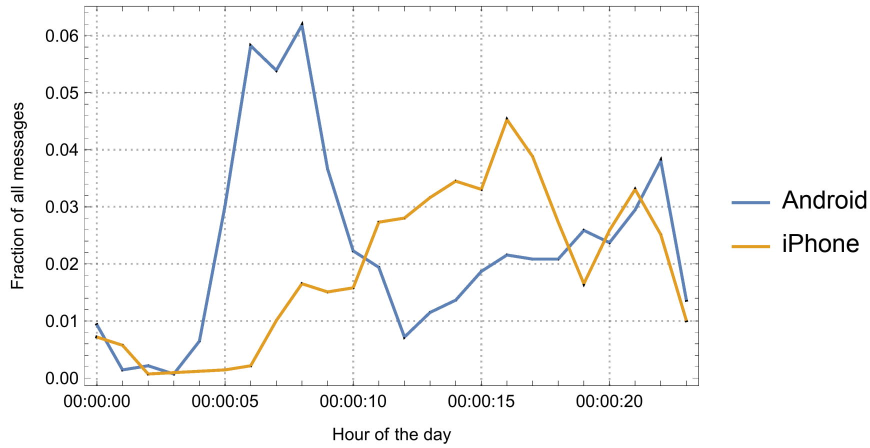
Out[31]=



Next, as in [1], for each device we can plot the fraction of the tweets made at different hours of the day:

In[32]:=
```
DateListPlot[{
  #/{1, Length[trumpTweetsTbl]} & /@
   Tally[Flatten[Cases[trumpTweetsTbl, {__, "Android", __}][[All,
      aTrumpTweetsTblColNames /@ {"hour"}]]]], #/{1, Length[trumpTweetsTbl]} & /@
   Tally[Flatten[Cases[trumpTweetsTbl, {__, "iPhone", __}][[All,
      aTrumpTweetsTblColNames /@ {"hour"}]]]]}, PlotTheme → "Detailed",
 PlotRange → All, FrameLabel → {"Hour of the day", "Fraction of all messages"},
 PlotLegends → {"Android", "iPhone"}]
```

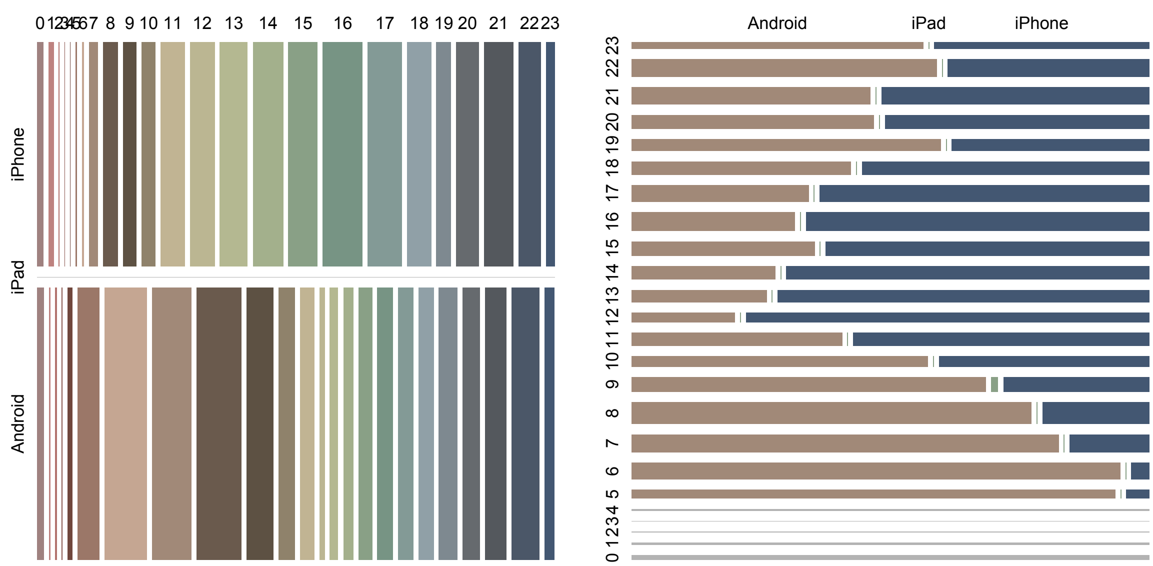
Out[32]=



Alternatively, we can use mosaic plots that fit better the discrete nature of those statistics:

In[33]:=
```
GraphicsGrid[{{
    MosaicPlot[trumpTweetsTbl[[All, aTrumpTweetsTblColNames /@ {"source", "hour"}]]],
    MosaicPlot[trumpTweetsTbl[[All, aTrumpTweetsTblColNames /@ {"hour", "source"}]],
     ColorRules → {2 → ColorData[7, "ColorList"]}]}}, ImageSize → 600]
```

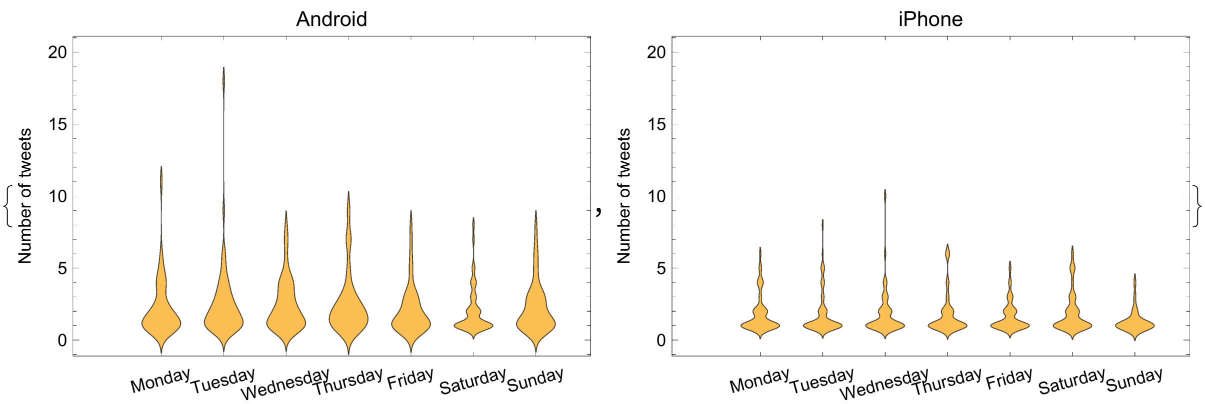
Out[33]=



Using the weekdays column we can plot the distributions of the number of tweets per day of the week:

In[34]:=
```
Block[{device = #, d},
   d = Cases[trumpTweetsTbl, {__, device, __}][[All,
     aTrumpTweetsTblColNames /@ {"year", "month", "hour", "weekday"}]];
   d = GatherBy[d, Last];
   d = SortBy[Map[{#[[1, 1, -1]], #[[All, 2]]} &, Tally /@ d],
     #[[1]] /. {"Monday" → 1, "Sunday" → 7, "Saturday" → 6, "Friday" → 5,
        "Wednesday" → 3, "Tuesday" → 2, "Thursday" → 4} &];
   DistributionChart[d[[All, 2]], ChartLabels → Map[Rotate[#, π / 12] &, d[[All, 1]]],
    FrameLabel → {None, "Number of tweets"},
    PlotRange → {0, 20}, PlotLabel → device, ImageSize → 300]
  ] & /@ {"Android", "iPhone"}
```

Out[34]=



Breakdown by sentiments

We can simply use the Mathematica built-in classifier for sentiments and plot some Bayesian statistics for sentiment-and-device pairs.

Note that this section uses alternative algorithms those of the section “Sentiment analysis: Trump’s tweets are much more negative than his campaign’s” in [1]. Also, note that because of Mathematica’s built-in classifiers we do not need to go through the steps in the section “Comparison of words” of [1].

Sentiments

First we add to the data a column with sentiments:

In[35]:=
```
trumpTweetsTbl = MapThread[Append, {trumpTweetsTbl,
    Classify["Sentiment", trumpTweetsTbl[[All, aTrumpTweetsTblColNames["text"]]]]}];
```

and extend the Association object for column names:

In[36]:=
```
aTrumpTweetsTblColNames = Join[
  aTrumpTweetsTblColNames, <|"sentiment" → Length[aTrumpTweetsTblColNames] + 1|>]
```

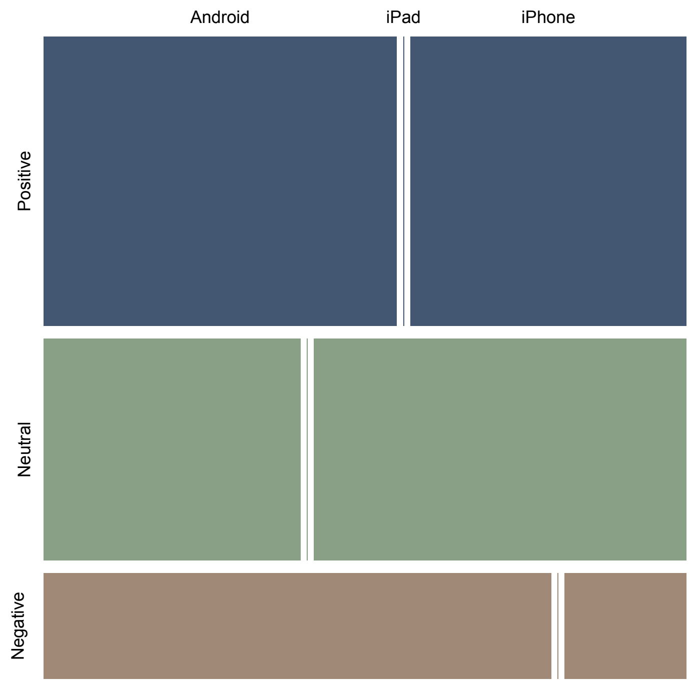
Out[36]=
```
<|id → 1, source → 2, text → 3, created → 4, year → 5, month → 6,
 day → 7, hour → 8, minute → 9, second → 10, weekday → 11, sentiment → 12|>
```

Sentiment vs. device

Then we make a mosaic plot to visualize the conditional probabilities:

In[37]:=
```
MosaicPlot[trumpTweetsTbl⟦All, aTrumpTweetsTblColNames /@ {"sentiment", "source"}⟧]
```

Out[37]=



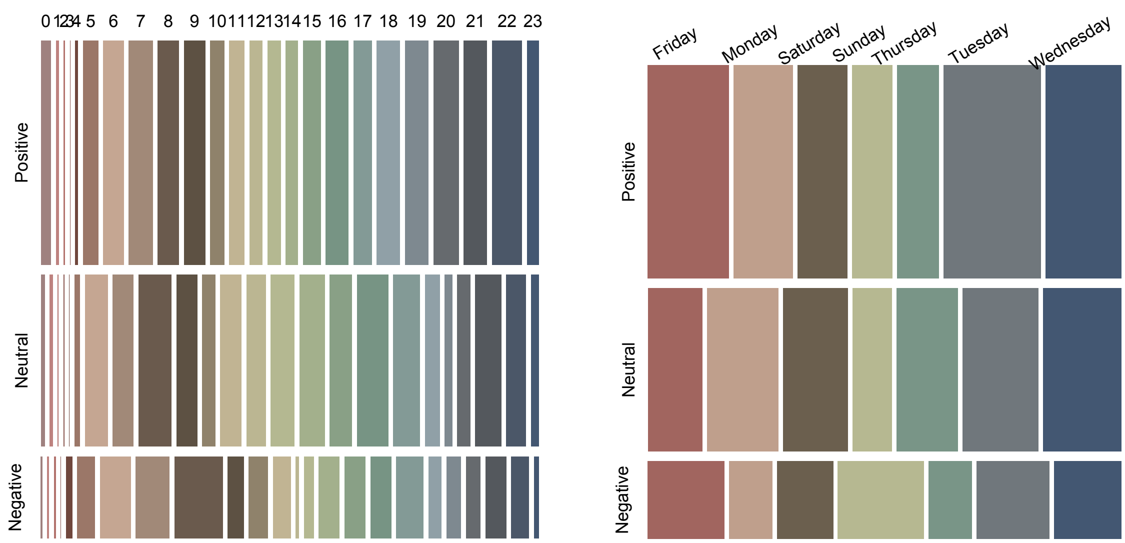
We can see in the plot above that there are much more negative tweets published through Android.

Sentiment vs. time

We can make a similar conditional probabilities plot sentiments and using time tags.

In[38]:=
```
GraphicsGrid[
 {{MosaicPlot[
    trumpTweetsTbl⟦All, aTrumpTweetsTblColNames /@ {"sentiment", "hour"}⟧],
   MosaicPlot[trumpTweetsTbl⟦All, aTrumpTweetsTblColNames /@ {"sentiment",
       "weekday"}⟧, "LabelRotation" → {{1, 0.6}, {0, 1}}]}}, ImageSize → 600]
```

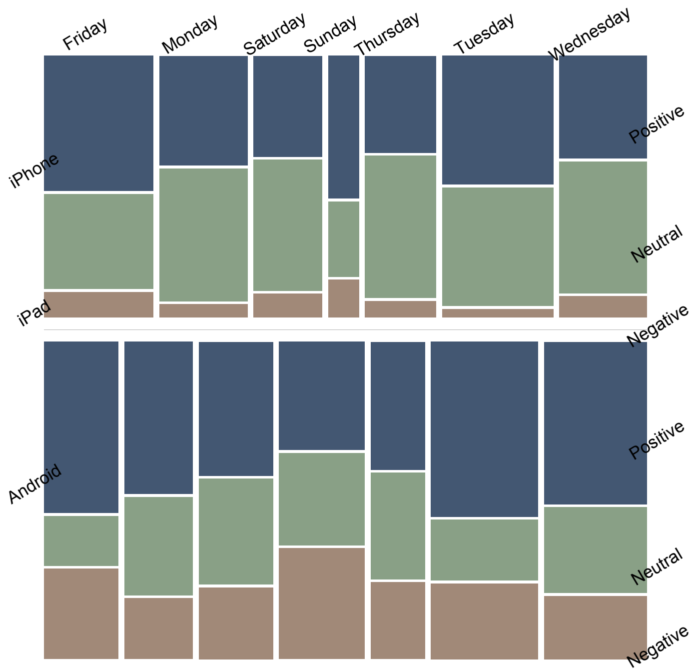
Out[38]=



Sentiment breakdown over devices and time tags

```
In[39]:= MosaicPlot[
  trumpTweetsTbl⟦All, aTrumpTweetsTblColNames /@ {"source", "weekday", "sentiment"}⟧,
  "LabelRotation" → {{1, 0.6}, {1, 0.6}}, ColorRules → {3 → ColorData[7, "ColorList"]}]
```

Out[39]=



Conclusions

We can see that the conjecture formulated in the introduction is confirmed by the sentiment-device mosaic plots in this section. We can see the Twitter messages from iPhone are much more likely to be neutral, and the ones from Android are much more polarized. As Christian Rudder (one of the founders of OkCupid, a dating website) explains in the chapter “Death by a Thousand Mehs” of the book “Dataclysm”, [10], polarization as a very good strategy to engage online audience:

[...] And the effect isn't small-being highly polarizing will in fact get you about 70 percent more messages. That means variance allows you to effectively jump several "leagues" up in the dating pecking order - [...]

Facebook topics

Another built-in classifier in Mathematica classifies to Facebook topics. Let us apply that classifier to the Trump Twitter data:

```
In[40]:= trumpTweetsTbl = MapThread[Append, {trumpTweetsTbl, Classify[
     "FacebookTopic", trumpTweetsTbl⟦All, aTrumpTweetsTblColNames["text"]⟧]}];
```

This extends the Association object for the column names to include the Facebook column:

In[41]:=
```
aTrumpTweetsTblColNames =
  Join[aTrumpTweetsTblColNames, <|"FBtopic" → Length[aTrumpTweetsTblColNames] + 1|>]
```

Out[41]=
```
<|id → 1, source → 2, text → 3, created → 4, year → 5, month → 6, day → 7,
 hour → 8, minute → 9, second → 10, weekday → 11, sentiment → 12, FBtopic → 13|>
```

Let us cross-tabulate the obtained topics vs. the device of publishing.

In[42]:=
```
ctRes = CrossTabulate[
   trumpTweetsTbl[[All, aTrumpTweetsTblColNames /@ {"FBtopic", "source"}]]];
Magnify[MatrixForm[ctRes], 0.6]
```

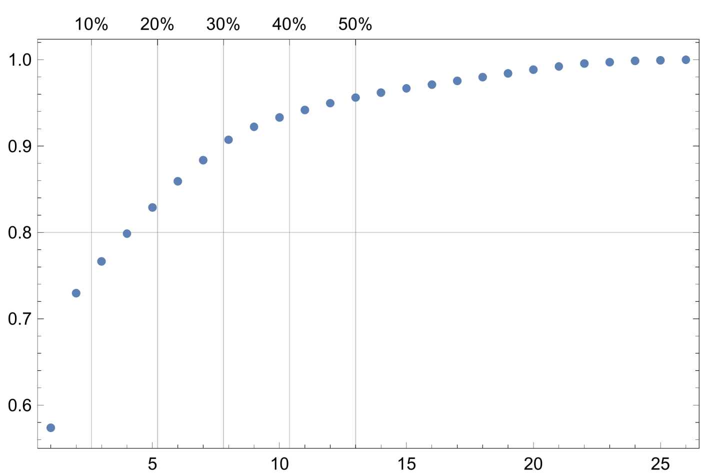
Out[43]=

	Android	iPad	iPhone
Books	1	0	0
CareerAndMoney	33	1	17
FamilyAndFriends	7	0	4
Fashion	2	0	0
Fitness	5	0	3
FoodAndDrink	3	0	2
Health	5	0	1
Leisure	8	0	7
Movies	8	0	1
Music	15	0	27
PersonalMood	4	0	2
PetsAndAnimals	0	0	1
Politics	508	0	290
QuotesAndLifePhilosophy	10	0	11
Relationships	4	0	2
SchoolAndUniversity	4	0	2
SocialMedia	6	0	1
SpecialOccasions	21	0	24
Sports	25	0	9
Technology	20	0	197
Television	23	0	10
Transport	1	0	1
Travel	6	0	6
VideoGames	4	0	1
Weather	4	0	2
Indeterminate	35	0	7

With the following Pareto law plot (for the function definition and other Mathematica examples see [6]) we can see that the top 4 topics are the class labels of 80% of the tweets:

In[44]:=
```
ParetoLawPlot[Total[ctRes["XTABMatrix"], {2}]]
```

Out[44]=



Similarly, the top 7-8 topics are the class labels of ≈ 90% of the tweets. Here are those topics (the fifth one is “Indeterminate” and is removed):

In[45]:=
```
topFBTopics = Drop[
  ctRes["RowNames"]⟦Reverse[Ordering[Total[ctRes["XTABMatrix"], {2}], -8]]⟧, {5}]
```
Out[45]=
```
{Politics, Technology, CareerAndMoney, SpecialOccasions, Music, Sports, Television}
```

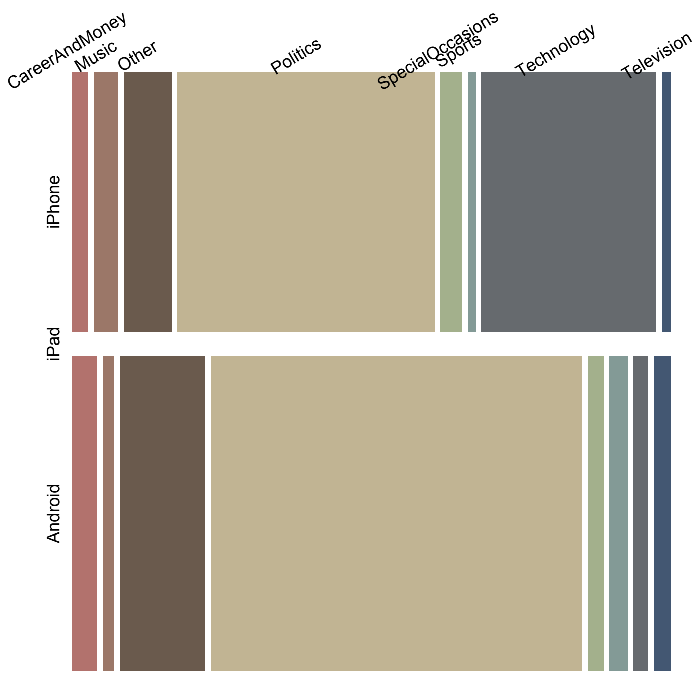
Here we make rules for mapping the non-top-Pareto topics into “Other”:

In[46]:=
```
toOtherTopicRules = Thread[Complement[ctRes["RowNames"], topFBTopics] → "Other"];
```

Now we can make a mosaic plot for device-vs-Facebook topic:

In[47]:=
```
MosaicPlot[trumpTweetsTbl⟦All, aTrumpTweetsTblColNames /@ {"source", "FBtopic"}⟧ /.
  toOtherTopicRules, "LabelRotation" → {{1, 0.6}, {0, 1}},
 ColorRules → {2 → ColorData[7, "ColorList"]}]
```

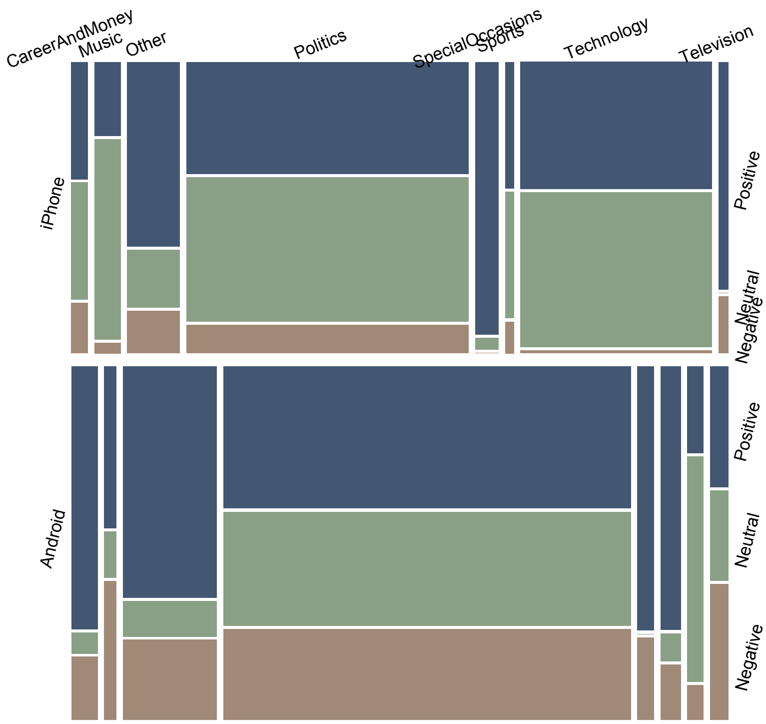
Out[47]=



Further, we can combine the sentiment classification results with the Facebook topics classification results:

In[48]:=
```
MosaicPlot[DeleteCases[trumpTweetsTbl, {___, "iPad", ___}][[All,
    aTrumpTweetsTblColNames /@ {"source", "FBtopic", "sentiment"}]] /.
  toOtherTopicRules, "LabelRotation" → {{1, 0.4}, {0.2, 0.8}},
 ColorRules → {3 → ColorData[7, "ColorList"]}, ImageSize → 400]
```

Out[48]=



Comparison by used words

This section demonstrates a way to derive word-device associations that is alternative to the approach in [1]. The Association rules learning algorithm Apriori is used through the package “AprioriAlgorithm.m”, [4]. For documentation and examples how the functions of the package [4] see [8].

First we split the tweets into “bags of words” (or “baskets”) and remove stop words:

In[49]:=
```
tweetWords = Select[#, StringLength[#] > 1 &] &@* DeleteStopwords@* StringSplit /@
    trumpTweetsTbl[[All, aTrumpTweetsTblColNames["text"]]];
```

(Further cleaning of the words can be done, but from the experiments shown below that does not seem necessary.)

Next to each bag-of-words we add the corresponding device tag:

In[50]:=
```
tweetWordsAndSources = MapThread[Append, {Union@* ToLowerCase /@ tweetWords,
     trumpTweetsTbl[[All, aTrumpTweetsTblColNames["source"]]]}];
```

We are ready to apply Apriori. The following command finds the pairs of words (frequent sets of two elements) that appear in at least in 1% of the messages (i.e. 13 messages):

In[51]:=
```
{ares, wordToIndexRules, indexToWordRules} =
  AprioriApplication[tweetWordsAndSources, 0.01, "MaxNumberOfItems" → 2];
```

The following commands find the association rules based on the found frequent sets.

In[52]:=
```
arules = AssociationRules[tweetWordsAndSources /. wordToIndexRules,
    ares⟦2⟧, 0.6, 0.007] /. indexToWordRules;
aARulesColNames = Association[MapIndexed[#1 → #2⟦1⟧ &,
    {"Support", "Confidence", "Lift", "Leverage", "Conviction", "Antecedent", "Consequent"}]];
```

These are association rules for “Android” being the consequent given in descending confidence order:

In[54]:=
```
Magnify[#, 0.7] &@Pane[
  GridTableForm[SortBy[#, -#⟦2⟧ &] &@Cases[arules, {___, {"Android"}, ___}, ∞],
   TableHeadings → Keys[aARulesColNames]],
  ImageSize → {800, 400}, Scrollbars → True]
```

Out[54]=

#	Support	Confidence	Lift	Leverage	Conviction	Antecedent	Consequent
1	0.0100647	1.	1.82546	0.00455118	-2.03649×10^{15}	{.m.}	{Android}
2	0.0172538	1.	1.82546	0.00780203	1000.	{@megynkelly}	{Android}
3	0.0805176	0.99115	1.8093	0.0360157	51.0978	{@realdonaldtrump}	{Android}
4	0.0172538	0.923077	1.68504	0.00701438	5.8785	{@cnn}	{Android}
5	0.0150971	0.913043	1.66672	0.00603913	5.20022	{wow,}	{Android}
6	0.0143781	0.909091	1.65951	0.00571405	4.97412	{beat}	{Android}
7	0.0215672	0.909091	1.65951	0.00857107	4.97412	{win}	{Android}
8	0.0129403	0.9	1.64291	0.00506388	4.52193	{u.s.}	{Android}
9	0.0222861	0.885714	1.61684	0.00850233	3.95669	{said}	{Android}
10	0.0107836	0.882353	1.6107	0.00408862	3.84364	{lyin'}	{Android}
11	0.0107836	0.882353	1.6107	0.00408862	3.84364	{won}	{Android}
12	0.0150971	0.875	1.59728	0.00564531	3.61754	{country}	{Android}
13	0.0143781	0.869565	1.58736	0.00532022	3.46681	{time}	{Android}
14	0.0237239	0.868421	1.58527	0.00875868	3.43666	{republican}	{Android}
15	0.0258807	0.857143	1.56468	0.00934011	3.16535	{big}	{Android}
16	0.0244428	0.85	1.55164	0.00868994	3.01462	{media}	{Android}
17	0.0115025	0.842105	1.53723	0.00401989	2.86389	{jobs}	{Android}
18	0.0150971	0.84	1.53339	0.00525149	2.8262	{job}	{Android}

These are the association rules for “iPhone” being the consequent:

In[55]:=
```
Magnify[#, 0.7] &@Pane[
  GridTableForm[SortBy[#, -#[[2]] &] &@Cases[arules, {___, {"iPhone"}, ___}, ∞],
   TableHeadings → Keys[aARulesColNames]],
  ImageSize → {800, 400}, Scrollbars → True]
```

Out[55]=

#	Support	Confidence	Lift	Leverage	Conviction	Antecedent	Consequent
1	0.0100647	1.	2.21497	0.00552075	-2.47034×10^{15}	{#trumppence16}	{iPhone}
2	0.015816	1.	2.21497	0.00867547	1000.	{#votetrump}	{iPhone}
3	0.0143781	1.	2.21497	0.00788679	4.94069×10^{15}	{#imwithyou}	{iPhone}
4	0.0194105	1.	2.21497	0.0106472	4.94069×10^{15}	{#americafirst}	{iPhone}
5	0.0301941	0.954545	2.11429	0.0159131	12.0676	{join}	{iPhone}
6	0.122933	0.944751	2.09259	0.0641864	9.92832	{#trump2016}	{iPhone}
7	0.0115025	0.941176	2.08468	0.00598486	9.32495	{#crookedhillary}	{iPhone}
8	0.0115025	0.941176	2.08468	0.00598486	9.32495	{soon!}	{iPhone}
9	0.0654206	0.91	2.01562	0.0329638	6.09474	{#makeamericagreatagain}	{iPhone}
10	0.0115025	0.888889	1.96886	0.0056603	4.93674	{#maga}	{iPhone}
11	0.140906	0.790323	1.75054	0.060413	2.61605	{thank}	{iPhone}
12	0.0136592	0.655172	1.45119	0.00424677	1.59073	{tonight}	{iPhone}
13	0.0366643	0.621951	1.3776	0.0100497	1.45094	{&}	{iPhone}

Note that an association rule with confidence 1 means that the rule is a logical rule.

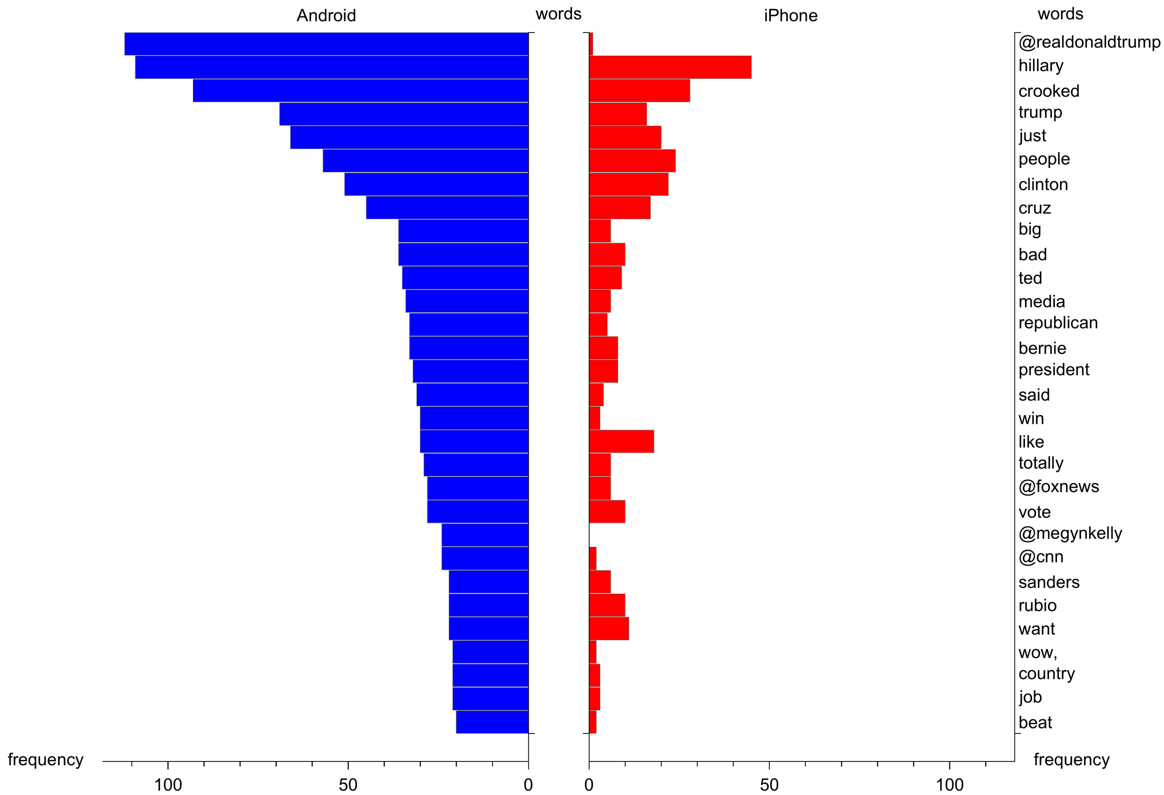
In order to make word comparison plots (paired bar charts) similar to those in [1] we can write the following function:

In[56]:=
```
Clear[DeviceWords]
DeviceWords[device_String,
   sortAxis_Integer, upTo_Integer, opts : OptionsPattern[]] :=
  DeviceWords[tweetWordsAndSources, device, sortAxis, upTo, opts];
DeviceWords[tweetWordsAndSources_, device_String, sortAxis_Integer,
   upTo_Integer, opts : OptionsPattern[]] := Block[{words, ctRes},
   (* Select the words most associatiated with the specified device. *)
   words = Flatten@Cases[arules, {___, {device}, ___}, ∞][[All, -2]];

   (* For each device and word find in how many tweets they appear together. *)
   ctRes = Outer[
     Function[{d, w},
      Length[Select[tweetWordsAndSources, Length[Intersection[#, {w, d}]] == 2 &]]
     ], {"Android", "iPhone"}, words];

   (* Sort the data according to the found frequencies. *)
   ctRes = Append[ctRes, words];
   ctRes = Transpose[
     Reverse@Take[SortBy[Transpose[ctRes], -#[[sortAxis]] &], UpTo[upTo]]];

   (* Make the paired bar chart. *)
   PairedBarChart[{ctRes[[1]]}, {ctRes[[2]]},
    ChartLabels →
     {Placed[{"Android", "iPhone"}, Above], None, Placed[ctRes[[3]], "RightAxis"]},
    AxesLabel → {"frequency", "words"},
    ChartStyle → {{Blue, Red}, None, None},
    opts]
  ];
```

Using the function defined above we can plot this comparison bar chart based on words most associated with Android:

In[59]:=
```
DeviceWords["Android", 1, 30, ImageSize → 600, AspectRatio → 0.8]
```

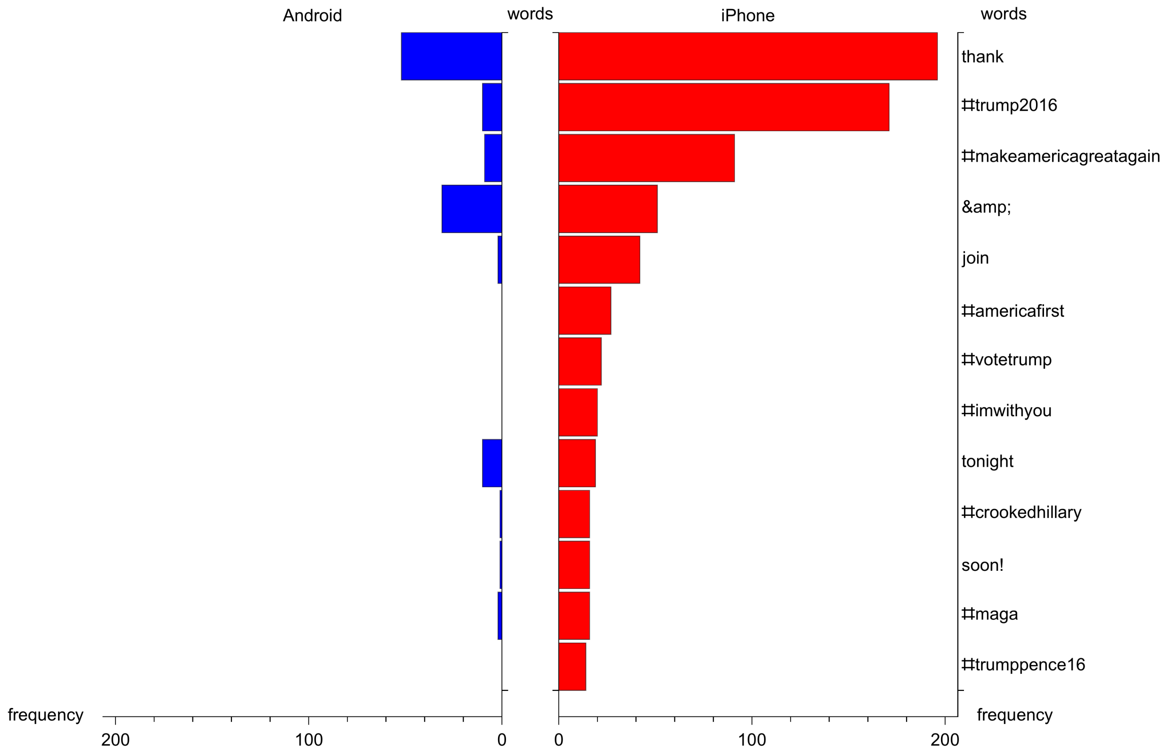
Out[59]=



And here is a comparison bar chart based on words most associated with iPhone:

In[60]:=
```
DeviceWords["iPhone", 2, 20, ImageSize → 600, AspectRatio → 0.8]
```

Out[60]=



We can see the from the paired bar charts that the frequency distributions of the words in tweets from Android are much different from those coming from an iPhone. The messages from iPhone seem more “official” since their most frequent words are hash-tagged.

Summary and further analysis

For general observations and conclusions over the data and the statistics see [1]. In this document those observations and conclusions are supported or enhanced with more details.

Using Mathematica’s built classifiers it was easier to do the sentiment analysis proposed in [1]. With the more detailed time tags the mosaic plots provided some interesting additional insights. Using association rules mining is a more direct ways of investigating the device-word associations in the Twitter data.

Possible extensions of the analysis are (1) to do dimension reduction over the “bags of words” derived in the previous section, and (2) to apply importance of variables investigation, [9], to the “bag of words” records. That latter analysis extension is more in the spirit of the text analysis in [1].

References

[1] David Robinson, “Text analysis of Trump’s tweets confirms he writes only the (angrier) Android half”, (2016), VarianceExplained.org.

[2] Anton Antonov, MathematicaForPrediction utilities, (2014), source code MathematicaForPrediction at GitHub, package MathematicaForPredictionUtilities.m.

[3] Anton Antonov, Mosaic plot for data visualization implementation in Mathematica, (2014), Mathemati-

caForPrediction at GitHub, package MosaicPlot.m.

[4] Anton Antonov, Implementation of the Apriori algorithm in Mathematica, (2014), source code at MathematicaForPrediction at GitHub, package AprioriAlgorithm.m.

[5] Anton Antonov, “Mosaic plots for data visualization”, (2014), MathematicaForPrediction at WordPress blog. URL: https://mathematicaforprediction.wordpress.com/2014/03/17/mosaic-plots-for-data-visualization/ .

[6] Anton Antonov, “Pareto principle adherence examples”, (2016), MathematicaForPrediction at WordPress blog. URL: https://mathematicaforprediction.wordpress.com/2016/11/17/pareto-principle-adherence-examples/ .

[7] Anton Antonov, “Mosaic plots for data visualization”, (2014), MathematicaForPrediction at WordPress blog. URL: https://mathematicaforprediction.wordpress.com/2014/03/17/mosaic-plots-for-data-visualization/ .

[8] Anton Antonov, “MovieLens genre associations” (2013), MathematicaForPrediction at GitHub, https://github.com/antononcube/MathematicaForPrediction, folder Documentation.

[9] Anton Antonov, “Importance of variables investigation guide”, (2016), MathematicaForPrediction at GitHub, https://github.com/antononcube/MathematicaForPrediction, folder Documentation.

[10] Christian Rudder, Dataclysm, Crown, 2014. ASIN: B00J1IQUX8 .