SYDDANSK UNIVERSITET
UNIVERSITY OF SOUTHERN DENMARK

FF501 First-Year Project

Social Networks Information Retrieving and Analysis

Mining and Analysing Publicity of Operating Systems on Social Networks

group 86a

Brian Pedersen <brped13@student.sdu.dk>
Morten Rovelt Hansen <mohan12@student.sdu.dk>
Apinayan Mohanathas <apmoh13@student.sdu.dk>

Supervisors:
Kaiji Chen
Li Su

June 5, 2014

1 Table Of Contents

Contents

Abstract

In our society a lot of products are bought on a daily basis. The companies that advertise and sell these products, often have little clue on how their products and services are received by their customers, and how their experiences with the products and services evolve. This is something businesses are commonly trying to get an idea of, by generating surveys for their customers and reading reviews. Many websites gives consumers the possibility to express themselves about their experiences with certain products or businesses, but the reviews and scores generated from these websites might be doubtful, as consumers often only visit these websites due to more extreme cases, e.g. a very positive or very negative experience. Many users use Twitter to write about their daily life, including the purchase of a new product, but also to proclaim their feelings for a certain company or product.

In this report we're taking a look at how data from Twitter can determine the publicity and popularity of three operating systems. The methods described include every aspect from the collection of data from Twitter and the creation of a usable model to analysis of the data itself.

Twitter is a social networking and micro-blogging website, where people can post messages of a maximum of 140 characters per post, for their followers to see. These posts are called tweets, and the meta data stored within these messages, or tweets, will be the main data source used in this study.

2 Introduction

The goal for the project is to give an idea on how data from Twitter, or other social networks, can be used to figure out the publicity and popularity of one or several products, or businesses. Because of the limited time in the making of this project, it was decided that we would look into the publicity and popularity of operating systems. This choice was made, since many Twitter users tweet about operating systems, and therefore it would be easy to collect data on the subject on the limited time. However the steps made in the making of this study is made with the original idea in mind; that it should also be possible to use our model on products or businesses.
One of the problems doing this, will be what happens to a company or product in the timespan we do the analysis. As an example, if a major scandal occur in a company, the rating will most likely be worse, than if they just released a very good product. Everything that happens in the company have a chance of influencing the rating drastically, if the analysis is done over a short period of time. Another parameter for it is the user who posted, e.g. his/her number of followers, count of followings and the number of time the tweet was favorited and retweeted.

2.1 Problem Formulation

By mining data in tweets on Twitter, we can do an analysis on the data to show a rating on a product or company, that would be similar or better to what other ratings made by other sites or with other methods would be.

By gathering specific tweets on keywords defining the product, the analysis will be more accurate on what the product publicity and popularity really is, based on real users, rather than looking at user reviews.

To get the right parameters for the modeling, first off is the keywords on the raw data, namely tweets and their metadata. This ensures we only get the data that mentions the specified product or company.

Based on Twitters Streaming API we gather information in a Python[1] script running on our cloud computer to store the tweets live. From here we take the data gathered and put it in our database, then run it through analysis using our mathematical model and parameters.

The output here will be the goal of the project, an overall customer analysis with a rating based on public opinions from Twitter.

[1]Python programming language

3 Methods

Overall, methods in data mining is all about getting the data you want, using the tools and skills at your disposal.

First off, you have to select what it really is you want to find. For that you have to pick a subject you want to analyse. For our matter, we picked operating systems but the choices are endless, looking at what data you can get from the Internet.

You could look at a smaller group of things, one with more keywords or less, but given the time we got to collect data, Operating Systems was a good choice, because we were able to get a reasonable amount of data, withing a relatively short period of time. From here all the data have to be stored, and then processed through various methods, mentioned later to get a better look at the overall becoming of the model.

When the data is processed and evaluated through analysis the results can finally be deployed.

3.1 Data Mining in general

Data mining in general is all about looking at big datasets, and finding patterns using statistics, keywords, analysis and machine learning.

When looking at it the overall goal, it is to extract the right information, and be able to analyse it and narrow it all down to get the best out of the collected data. There is a lot of complexity looking at the raw data, but the trick is to sort out all the unwanted data.

There's a lot of methods to be used for data mining, most of them is very efficient. The term datamining have been around for very long, but the methods used today is a lot more complex and faster. Today's datasets are usually very large, and hard to narrow down. For getting the related data you want and get rid of all the "noise", data cleaning is used. This gets rid of all the unrelated, hence unwanted, data. After that a more productive approach is taken, looking at what is really wanted from the data, and sorting out the unwanted data. At last running it through a model can help get analyzable results from the data.

3.1.1 Twitter API

The Twitter Streaming API makes it possible to sample around one percent of tweets being sent at the moment in real-time. Even though the text in a tweet only makes up 140 bytes there is almost 5 kilobytes of metadata in a tweet sampled through the API. The metadata can be about the tweet, the user and the retweeting user. Because the Streaming API is only allowing sampling in real-time we chose not to use the completely new and original tweets since they haven't had time to get retweets and favorites. But the Streaming API is also sampling retweets in real-time and these have had time to get favorited and retweeted.

We used a Python module named Twitter which functions as a wrapper around the the Twitter API. Using this we could avoid dealing with raw HTTPS connections and API calls. In addition the package make it possible to get a tweet as a dictionary, a native Python data type.

3.2 Storing and Retrieving

Since it would be unpractical to have our own computer turned on over the span of several days to sample tweets we have had to use a Virtual Private Server. We accessed the server through SSH which allows us to run terminal commands on the remote server.

3.2.1 VPS-Server

A Virtual-Private-Server is a service used for projects like this. Its a physical other machine you rent, to get a server that is active at all times so you dont have to have your own computer running. As we are using the Twitter Streaming-API, this is necessary to have, as it would'nt be efficient to have it running on our own computers. The VPS gives a lot of advantages, as we can have it running on a script collecting the data wanted, 24/7 without any interruptions, only the boundaries of the API from Twitter itself.

3.3 Mining Tweets

The mining was handled by a Python script, which connected to a Twitter Stream through the Twitter API.

Because of problems at the beginning of our mining phase, our mining script is not sending the tweets directly to MongoDB, but instead saving the Tweets into text files.

The script starts by connecting to the Twitter API through OAuth. Afterwards, it gets a live stream from Twitter, which filters the stream to only give us the Tweets which contains the keywords, and only Tweets in English.

We came up with saving the Tweets in piles of 5000 Tweets, meaning that the mining script saves one Tweet per line in a text file. When 5000 Tweets is reached, it automatically creates a new file and saves the next 5000 Tweets in that file.

This was an intended design choice. We were concerned of the possibility of suddenly corrupting all of our collected data if we had simply saved all our data to a single file. If something wrong would happen, either by human mistake, or an error in the Streaming API, there was a slight chance that this could corrupt all data in the file. Another possibility could have been saving each Tweet in its own file, but this could also have led to problems after creating hundred of thousands of files, since some file systems have a limit on how many files is allowed per directory.

By using the algorithm below, if anything would go wrong or get corrupted, it would most likely only be the file the mining script was currently working on, meaning that a maximum of 5000 Tweets would get corrupted, which is clearly better than our whole dataset.

```
file_number = 0
tweet_number = 0

foreach (Tweet from Twitter Stream) {
        if(tweet_number is zero or dividable by 5000) {
                open a new file
                file_number + 1
        }
        write Tweet to file
}
```

More formally, the script saves tweets to a file, then closes the current file and opens a new file to write to if

$$\text{Number of Tweets} \pmod{5000} = 0$$

3.4 About MongoDB and PyMongo

To quickly search, filter and retrieve tweets we store them in an instance of the MongoDB database. MongoDB is a so-called NoSQL database that is easy to use especially for data like a tweet where metadata is stored in a structured and "attribute: value"-style format known as JSON.
Using the PyMongo distribution, which contains tools making it possible to work with MongoDB in Python. We chose MongoDB mainly because it made it easy to save the data *as it is*, and read each Tweet from the database, without actually changing the structure of the Tweet. If we had used one of the more popular SQL databases, e.g. MySQL, we would have had to extract each part of each Tweet, inserting them into the correct places in the database. Also it would have meant the creation of multiple tables, since each Tweet does not necessarily contain the same number of parameters. MongoDB creates its parameters *on the fly*, and scales for each Tweet, making it ideal for our purpose.

Since our remote server, which mined all data, wasn't able to run MongoDB we had to do all steps of the analysis on a local machine, meaning everything that had something to do with MongoDB.

3.5 Storing Tweets in MongoDB

We chose to store the collected Tweets in MongoDB for the ease of use, and because the Tweets we collected from the Twitter Streaming API outputted Python Dictionaries, which works perfectly in combination with MongoPy (the Python implementation of MongoDB).
We were able to just insert the whole dictionary into MongoDB, and afterwards being able find each Tweet, as it was given to us by the Twitter Streaming API. This extract-and-insert approach was handled by script simple file-to-db-script.

The script is simple, and is made simply to extract the Tweets from the files to the database, to make the data easier to work with. It simply goes through every file in a directory, and for each file insert each line into the database using PyMongo's build-in `insert()` method.

3.6 Pre-analysis

The pre-analysis, or rather the generation of the numbers of data was handled by a pre-analysis script written in Python, and is made as a script to run through all the Tweets stored in the MongoDB instance. The script is given the same keywords as used to filter the Twitter Stream in the mining script, and saves several informations about the data during runtime:

- Total count of Tweets.
- The score of a Tweet, using the mathematical model.
- Highest and lowest score of a Tweet.
- Total scores of keyword.
- Total score of keyword for each day.
- Total number of Tweets for each keyword.
- Total number of Tweets for each keyword for each day.
- Average score per Tweet for each keyword.

For the sake of convenience, the script writes all the above computed numbers to a text file. Ideally these numbers could also be used for other things, for example auto-generation of graphs, however this is not implemented at the moment.[2]

The script is rather static, but could have been made more dynamic for the sake of scalability and ease of using the script for analysing other types of tweets, e.g. other keywords, over longer periods of time and more calculations. Nevertheless the script is kept as simple as possible, making it easy to use different mathematical models on the Tweets or change anything else regarding the analysis.

[2]See the Future work and Improvements section 6

3.6.1 How it works

The pre-analysis script runs through all Tweets in a MongoDB instance, and analyse each Tweet, giving it a score, count keyword up, and so on.

Although, since our mathematical model uses the *number of favorites* and *number of retweets* as parameters to calculate the score of a Tweet, it is not possible to take these two parameters from any Tweet from the stream directly. The problem is that the Twitter Streaming API streams the data in real-time, meaning that none of the Tweets would contain any count of Retweets nor Favorites. Fortunately if a Tweet is a Retweet, parameters are passed, giving access to the original posts number of favorites and retweets, at the time of the Retweet. Because of this, the script filters to only get Retweets, and no original Tweets.

Another filter in the script is to only get Tweets from users with a Followers count and Following count both above zero, as our mathematical uses these two parameters to calculate the "Impact" of a user. If a user doesn't have any followers, the user can't have any impact, and should not be used for generating the statistics.

3.7 Analysis

To guarantee the reliability of the tools we have used in the project, we had to run some tests on the different methods. We ran all our data through our mathematical model:

$$\sum_{N=1}^{Tweets} : (Polarity \cdot Subjectivity) \cdot \left(\left(\frac{Followers}{Following} \right) + \left(\frac{Favorites}{2} \right) + (Retweets) \right)$$

Polarity and Subjectivity in this matter comes from Sentiment-Analysis TextBlob, Rating the content of the data written in the real post on Twitter. It looks at how much opinion a tweet has rating it from zero to one, and also gives it a score from minus one to one, looking at the positivity/negativity rating in the post. The rest can be split up in rating the post itself, and rating the user. Here the relationship between a users followers/following is the user rating, and favorites/retweets is used to up the score of the post itself.

4 Results

4.1 Testing

Before starting our analysis we needed to collect a couple of various tools and features to look at when doing it.
For this we chose a sentiment analysis, and some meta-data. Looking at the tests in general we got a good picture of how we wanted the final model to look like, and how it should be done. A lot of factors was not really known to us before starting the tests we ran on every part of the mathematical model in parts.

4.2 Features and Modeling

Features in Data Mining is the attributes and values you use to state how you rate the content of the data. for looking at this, we selected a good working Sentiment analysis and a few of the datas stored from the Twitter API in the meta-data of a tweet, everytime we get one. For getting a realistic picture of this, we are weighting the features differently corrosponding to how much it means for the data, and how much they individually has meaning.

4.2.1 Sentiment-Analysis

Our Sentiment Analysis tool TextBlob[3] output the parameters *subjectivity* and *polarity*. The *subjectivity* has a range of $[0.0; 1.0]$ and indicates how much opinion there is in a text. The *polarity* has a range of $[-1.0; 0.0; 1.0]$ which indicates whether the opinion is negative, neutral or positive.

We are weighting the polarity along with the subjectivity by multiplying them, and from this we get a *sentiment*. This sentiment is then used to give the tweet an influence score with either a negative or positive polarity. A neutral sentiment would nullify the tweet's influence score.
Some basic statistics we got from our sentiment analysis by analyzing a sample of some 15,000 tweets from our data set:

On the subjectivity:

- Around 40% of the tweets have a subjectivity of 0.0 or in other words they're neutral or "opinionless" according to TextBlob.
- 60% of the tweets have a subjectivity above 0.0.
- 48% of the later tweets have a subjectivity of 0.5 or above.

[3]A module for the Python programming language

On the polarity:

- Around 51% of the tweets have an absolute (either positive or negative) polarity above 0.0.
- 39% of the tweets have a positive polarity above 0.0.
- 12% of the tweets have a negative polarity below 0.0.

As can be seen above a good amount of the tweets are without any opinion according TextBlob. However the majority of tweets seem to have some kind of opinion. This is likely because many people mentioning "Ubuntu", "OS X" or "Windows" are more likely to do it when they mention a specific opinion about these products.

4.2.2 Features from metadata

From the metadata we get from each tweet we can look at the three following types of relationships on each post/user, to get a better insight on how to rate it:

Retweets: A Retweet does not just show that other users in general likes the post the corresponding user has posted, but also wants their entire audience to see the post as well. Its a great thing to look at, and is weighing a lot in the final score, because it defines that the content in the post is either good or bad, and the user agrees with it.

Favorites: Adding a post to favorite's is like saying that the corresponding person likes the thing he favorited. Although a user does this and agrees or likes the post, it does not weigh as much as a Retweet, because it does not show that the user wants his audience to see it too.

Followers/following: Looking at this form of data, were not just looking at the post itself like with the other features, but we are making a rating for the user posting the tweets reliability. a bigger audience given from the users followers will give a higher rating.

4.2.3 Weighting the features

To get a better view on whats important and not important in the dataset, we have been looking at how to weigh the features we have selected. All features is a key part of the final analysis, but some have more meaning to be looking at than others. For example, a Retweet is showing that a Person

not only likes the post he has seen, but also wants his entire audience to see it and rate it too. Different from Favorites that does not show on his own posts, but only show that he likes the post. We decided that Favorites would have only half the meaning that a Retweet has, and therefore divided it by two in the final analysis.
Looking at the user, we went through various testings on how to rate this. While PageRank[4] not being an option at hand, we went on and found out that taking the relationship between a users followers and the people he is following, would give the best overall picture of his reliability rating.

4.3 Data Collection

Our final data set contains 183388 tweets and the tweet scores based on our model is in the interval $[-14396; 74829]$ both endpoints being rather extremes.

[4]Refering to Future work and improvements section 6

4.3.1 Distribution of keywords

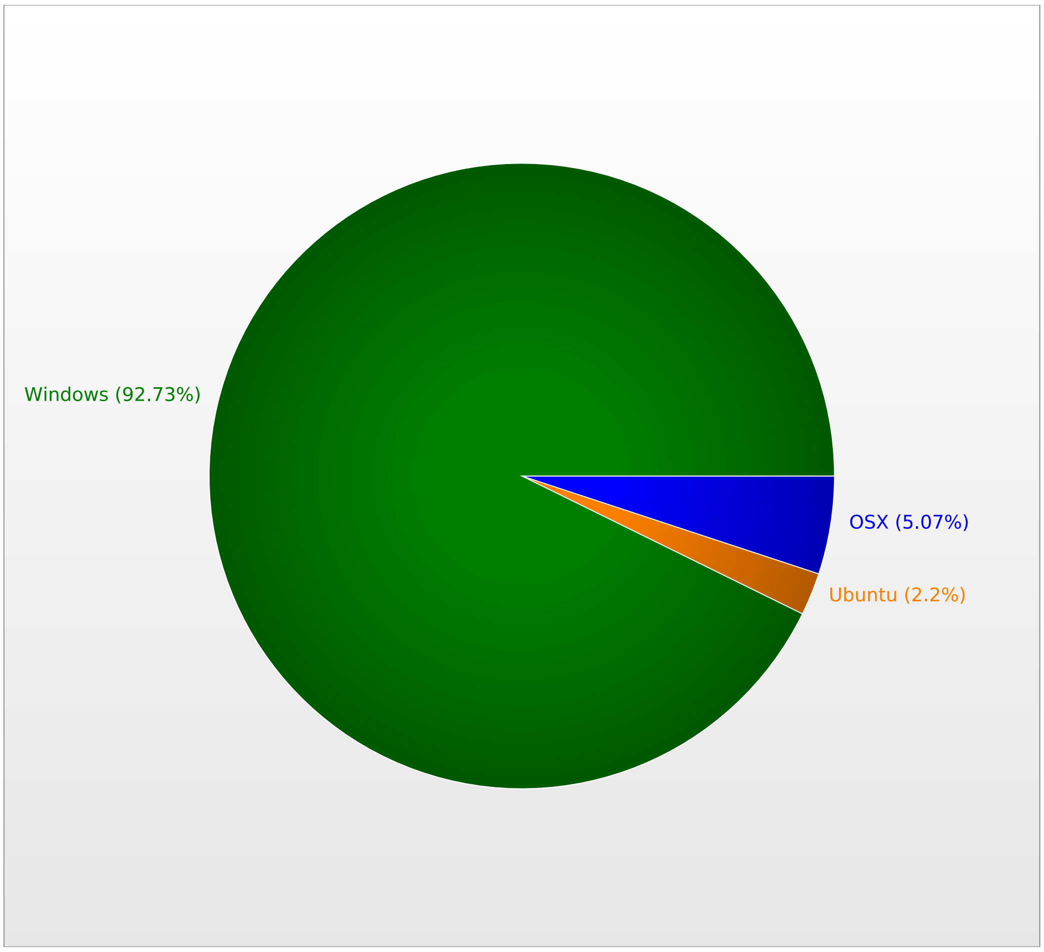

Figure 1: Distribution of collected keywords (total)

On Figure 1 we can see that Windows gets tweeted about the most, with the keyword contained in 92.73% of the tweets. OSX appears in 5.07% of the tweets, while Ubuntu only appears in 2.2% of the tweets. According to several other sources specializing in market shares in technology[5], the shares of OS X, Windows and Ubuntu, is a close match of the data presented in Figure 1. Concluded from this, the data collected most likely resembles real-life, and gives a reliable impression on what people think of some of the operating systems on the market.

[5]netmarketshare Desktop Operating System Market Share, May 2014

Keyword:	
% of Ubuntu	2.20%
% of Windows	92.73%
% of OSX	5.07%

4.3.2 Distribution of keywords over 5 day period

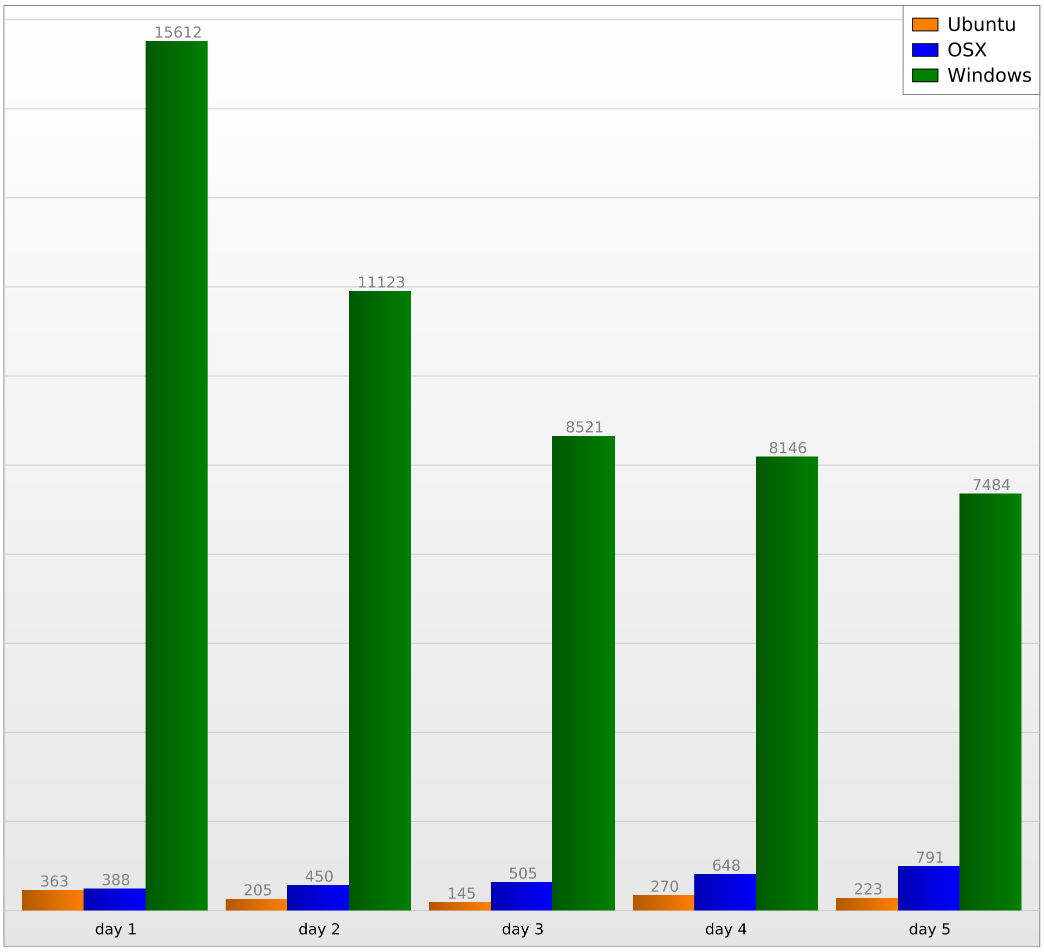

Figure 2: Distribution of collected keywords over 5 day period

Figure 2 gives an impression on how total count of keywords from Figure 1 was distributed over the time of the data collection. As can be seen, Ubuntu is somewhat stable, with a average of 241 tweets per day, and with a low range from the highest count, to the lowest. OSX is increasing a bit for each day, which could mean an increasing interest, but is most likely because of a temporary interest on the subject. This could be influenced by a certain post, for example a share of a media article involving OSX, being retweeted a lot of times.

However, the most noteworthy thing about Figure 2, is that Windows is decreasing. We aren't sure of the reason to this decrease in interest, but it is possible that an controversy about an update deadline on May 13[6], and the release of new updates on May 16[7] have triggered a sudden increase in the interest of Windows, which have, while we collected the data, started to decrease again.

Keywords count:	1	2	3	4	5	Total hits/key
Ubuntu	363	205	145	270	223	1206
Windows	15612	11123	8521	8146	7484	50886
OSX	388	450	505	648	791	2782
Total keywords/day	16363	11778	9171	9064	8498	54874

[6]Softpedia.com, *One Day Before the Deadline, Some Users Still Can't Install Windows 8.1 Update*, May 12, 2014

[7]Infoworld.com, *Microsoft acknowledges more errors, 80070371 and 80071A91, when installing Windows 8.1 Update/KB 2919355*, May 16, 2014

4.3.3 Total scores

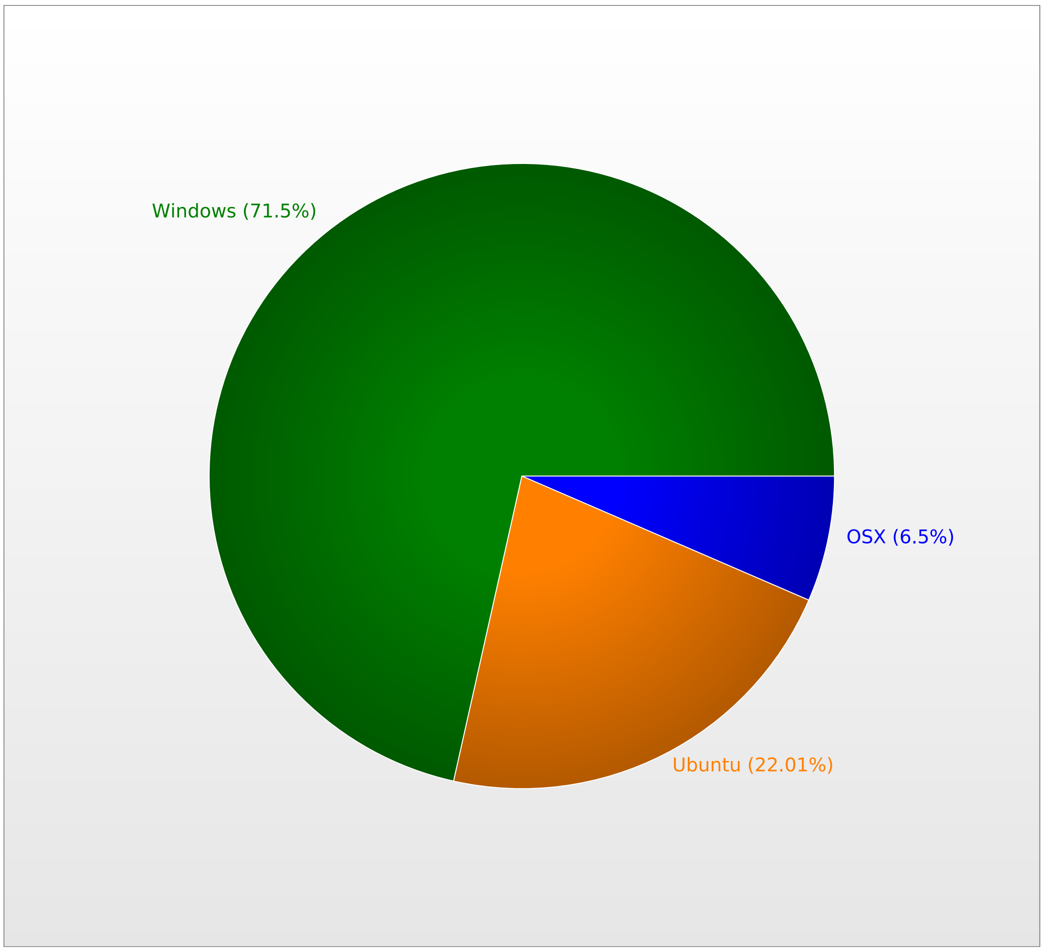

Figure 3: Distribution of the scores on the different keywords

Figure 3 shows the results of the Tweet's score after they were run through our model. Not surprisingly, Windows got the highest overall score with 71.5%, but with the fact in mind that Windows was represented by 92.73% of the tweets, this score is rather low. Ubuntu however, which was represented by only 2.2% of the tweets, got a score above 22%, after being run through our model.

Total scores:	
Ubuntu	22610
Windows	73455
OSX	6676
Sum of scores:	102741
% of Ubuntu	22.01 %
% of Windows	71.5%
% of OSX	6.5%

4.3.4 Average score per tweet

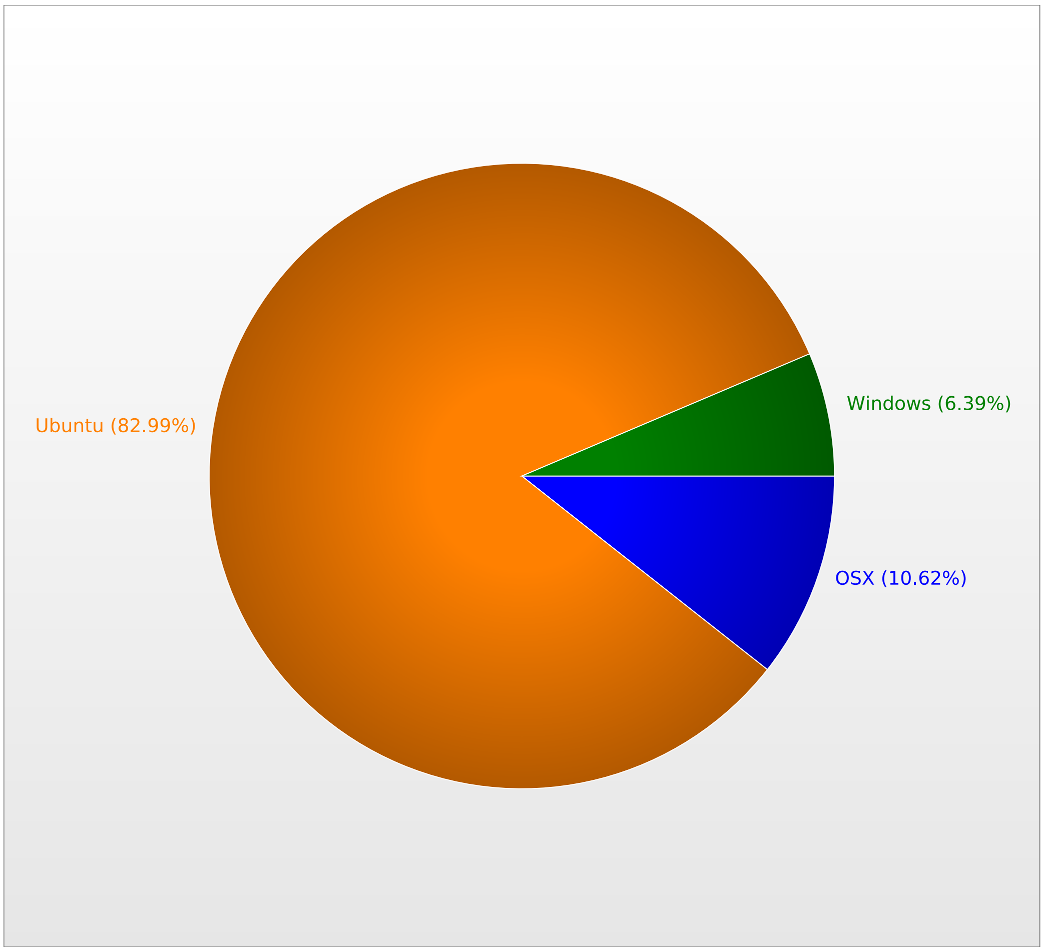

Figure 4: Distribution of the average score of tweets on the given keyword

What isn't surprising about Figure 3 is that Windows, while being the most mentioned of the keywords in our dataset, is the most popular according to our model. However, Figure 4, which shows the average score per tweet on the specific keyword, tells a whole different story. Figure 4 clearly shows that even though Ubuntu was the least mentioned of the keywords in our dataset, the tweets are generally a lot more positive than the tweets regarding OSX and Windows.

Does this mean that Ubuntu users are generally happier with their operating system than users of OSX and Windows? There could be several flaws with this theory. One is that, as mentioned earlier, Windows' recently controversy about their recent update could have led to lower average score per tweet, because of a lower sentimental score of the tweets. Another theory could be the on-going war between Microsoft and Apple, which means tweets of Windows users criticizing OSX and vice versa.

However, we have concluded that users of Ubuntu probably is happier with their operating system, or is at least more eager to share their satisfaction on Twitter, than the users of OSX and Windows. While the mentioned problems with this theory may have had an influence on the dataset, it's difficult to acknowledge the huge difference in the distribution of the average score of tweets, on the different keywords.

Average scores:	
Ubuntu	18.75
Windows	1.44
OSX	2.40
Average:	7.53

4.3.5 Average Tweet score over time

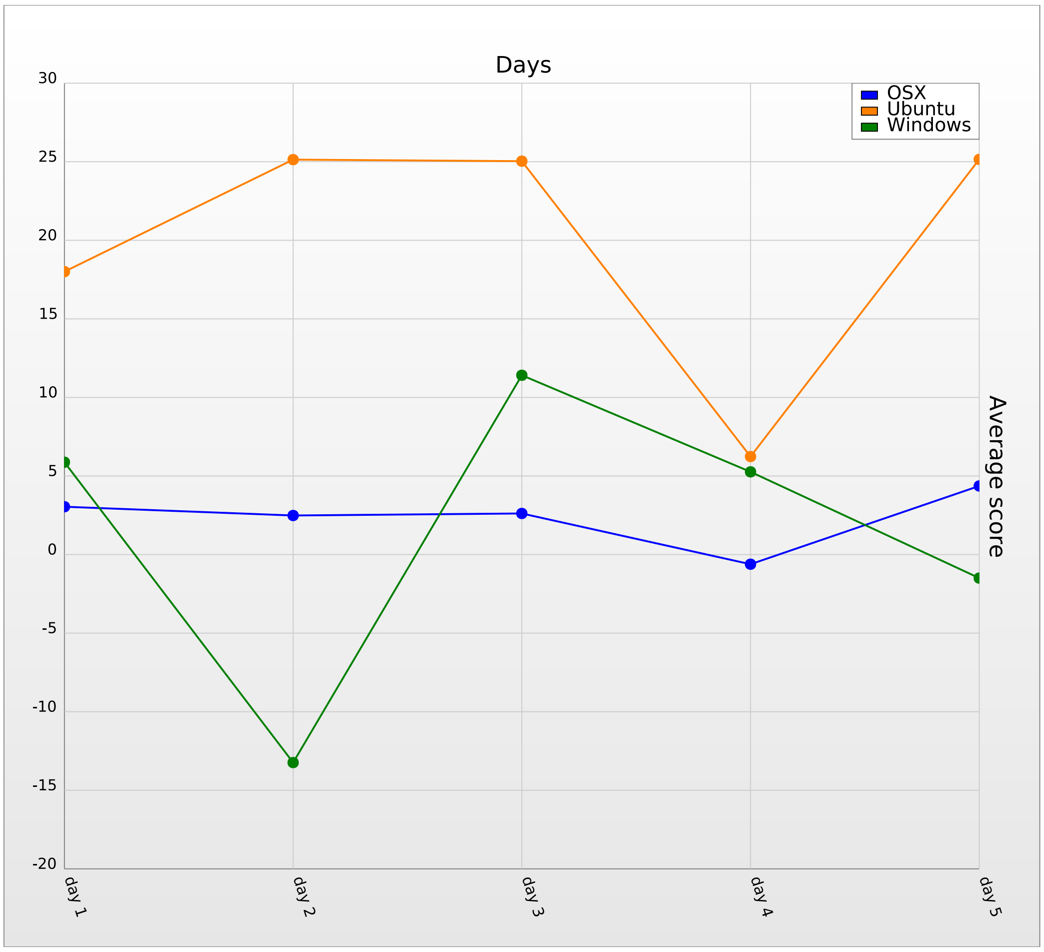

Figure 5: The average score (in thousands) of the tweets on the given keywords over the 5 day mining period.

Figure 5 gives a more detailed view of the things stated by Figure 4. This shows that the tweets mentioning Ubuntu, in the 5 day mining period, had a general higher average score, than the tweets mentioning OSX and Windows, every day of mining period. The average score of the tweets mentioning OSX was the most stable, while Windows was the most unstable from day to day. The reason why Windows was the most unstable, could also have had something to do with the aforementioned Windows controversy.

5 Conclusion

The results of the project has shown that the distribution of tweets mentioning products within the same category can be very close to the actual market shares of these products. Looking at the model we have derived that it is appropriate to look at how several products within the same category is mentioned on Twitter although it can also be used on a single product to show whether the product is overall positively or negatively mentioned.

Based on our model and results it seems likely that this can be used as a replacement for product rating websites, due to the fact that the results is made entirely from the posts the users makes themselves. Looking at our comparison of operating systems, the representation of tweets mentioning Windows is clearly the largest, compared to the other two, but as seen in our results, this does not necessarily have any influence on the rating of the product.

6 Future work and Improvement

Many things were discussed during the time of our project, and unfortunately we didn't have time to implement everything we talked about. Some of the most interesting ideas, that we did not have time to implement is:

Implementation of the Page Rank Algorithm Rating the user who wrote a Tweet depending on the rating of his/hers followers.

Auto-generated graphs Implementation of *python-gnuplot* or *matplotlib* to automatically create graphs of the generated data.

Make the pre-analysis script more scalable Make pre-analysis script more scalable, for use on e.g. other Tweets or over a user-defined period of time.

References

[1] Netmarketshare.com
Desktop Operating System Market Share, May, 2014
http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpcustomb=

[2] Infoworld.com
Microsoft acknowledges more errors, 80070371 and 80071A91, when installing Windows 8.1 Update/KB 2919355, May 16, 2014
http://www.infoworld.com/t/microsoft-windows/microsoft-acknowledges-more-errors-80070371-and-80071a91-when-installing-windows-81-updatekb-2919355-2426?source=rss_infoworld_blogs

[3] Softpedia.com
One Day Before the Deadline, Some Users Still Can't Install Windows 8.1 Update, May 12, 2014
http://news.softpedia.com/news/One-Day-Before-the-Deadline-Some-Users-Still-Can-t-Install-Windows-8-1-Update-441744.shtml

[4] Matthew A. Russell
Mining the Social Web, October 2013, O'Reilly

[5] Matthew A. Russell
21 Recipes for Mining Twitter, January 2011, O'Reilly

[6] Gundecha, Pritam and Huan Liu
Mining social media: A brief introduction, 2012, Tutorials in Operations Research

[7] Twitter Developer API Documentation
http://dev.twitter.com/docs, Twitter API General Documentation
http://dev.twitter.com/docs/api/streaming, Twitter Streaming-API Documentation