Sporty twitters

Is it possible to correlate the physical activity of a person with his psychological well-being?

Virgile LANDEIRO DOS REIS
Illinois Institute of Technology: Computer Science
Chicago, IL, USA
vlandeir@hawk.iit.edu

***Abstract*—This document is a report for a 2 month project for a machine learning and social media course at IIT. The first objective of this project was to establish a correlation between the physical activity of a twitter user and his psychological well-being. However, as such a study would require more than 2 month to be correctly carried out, the objective was revised down to a sentiment overall comparison between users that utilize exercise trackers and users who don't. The results do not show a difference in the overall sentiment but they are based on a classifier with a 59% accuracy.**

Keywords—machine learning; twitter; exercise trackers; NLTK; naïve Bayes; maxent

I. INTRODUCTION

This project is organized in three main parts. First of all, data has to be collected in order to run the experiments. To do so, the Twitter Streaming API is used to collect tweets that are automatically published by exercise trackers. From those tweets, a list of unique users is extracted. They are qualified as the "sportive users" (SU). By getting the friends of these users that do not use exercise trackers, a list of "non sportive users" (NSU) is created. Finally, by collecting the last 200 tweets for each user (SU and NSU), the corpus used in our experiments has been built. In the second part of this project, sentiment polarity classifiers are trained using the data from Sentiment140 and the Natural Language ToolKit (NLTK) in Python. For each of this classifier, the accuracy and the F1-measure are computed. Three of these classifiers are then used in our third main part: the experiment. For every tweet of every user, and using the trained classifiers, the sentiment polarity is computed. This allows computing an average sentiment polarity for each user representative of the user's overall sentiment. Then, by comparing the results over the SU and the NSU, our hypothesis is that SU tend to be more positive than NSU. However, due to the classifiers accuracy and the type of the task, the results do not support our hypothesis.

II. DATA

A. Training and testing data

The data is divided in two categories: the training data and the testing data. For the training data, we use the dataset published by Sentiment140 researchers: it is a set of 2,000,000 tweets which are labeled as positive (4) or negative (0). For the testing, we build our own dataset as described in the following sections.

B. Using the Streaming API to collect the exercise trackers' tweets

The first step in the process of building the corpus of tweets is to collect the users and this is done by using the Twitter Streaming API. From October 18th 2013 to October 31st 2013, 211,167 tweets are collected by scrapping the tweets from the Twitter stream and by filtering them with four exercise trackers hashtags: runkeeper, nikeplus, runtastic, and endomondo. To take advantage of the Twitter Streaming API, we use a Python library called tweepy. The collected tweets are stored in a JSON format and only some fields of the tweets are kept:

- The tweet ID
- The user ID
- The publication date
- The user profile language
- The tweet geolocation if it has been posted by the user, null otherwise
- The content of the tweet

Now, we want to collect the unique SU and keep a subset of those to run our experiment but first, we have to remove some aggressive marketing tweets published by the exercise trackers editors. Indeed, the editors tend to promote their application by aggressively tweeting by using the same hashtags as used in our filter. To remove the account from the editors, a first solution was to look at the accounts that have posted the greatest number of tweets manually and remove the editors' ones. Yet, this method does not assure that every editor's account has been removed from the dataset. Surely, at first we might think that there would only be four editor's accounts (one for each hashtag) but if we take the example of the nikeplus hashtag, we observe by doing a simple search on Twitter that Nike has created several accounts such as "Nike Running", "Nike Running Canada", "Nike Running France", etc. and each of this account use our selected hashtag "nikeplus". Consequently, we cannot use our first method to remove editors' accounts. The second method, which has been successfully used, was to inspect automatically posted tweets and discover one or more patterns that are present in every

tweet that are posted after an exercise. We obtain the following list of patterns:

- "just posted"
- "i crushed"
- "just finished"
- "just ran"
- "just completed"
- "was out"

To obtain the set of wondered tweets, we remove every tweet that does not contain one of these patterns. One advantage of this method is that it does not only remove marketing tweets but also tweets that are not linked to a sport activity. For example, we remove a tweet which content is "what's the best app? #runkeeper or #runtastic". Clearly, we want to remove this kind of tweets from our dataset and it is done by applying the second method to get our dataset of tweets.

C. *Collect the unique users and their data from the tweets dataset*

Once the tweets published by the exercise trackers are collected, the next step to build the corpus is to collect the unique users. This is done by a simple Python script that scans the collected tweets and creates a dictionary which key is the user id and which value is the number of posted tweets. Running this script on the 211,167 tweets, we get a list of 81,853 sportive users. To reduce the time passed in building the corpus, we only keep 2,000 SU. Now that the list of SU has been made, we use the following process to create a list of NSU:

- The list of friends is collected for each user amongst the SU.
- For each list of friends, we pick one friend randomly to create the list of NSU.

A user v is the friend of a user u if u follows v and v follows u.

D. *Create the corpus*

Using the Twitter Search API via the tweepy library, we collect the 200 last tweets (at most) for each user from the SU list and the NSU list to obtain a theoretical set of 400,000 tweets. However, not every user has posted 200 tweets, and several tweets are being filtered in order to optimize our experiment: the sentiment analysis. Therefore, we remove every tweet that has been automatically generated by exercise tracker applications. We also remove users that have not published any tweet and users that have more than 1,000 followers or more than 1,000 followees to avoid marketing accounts that have not been removed by our filter over the sports tweets. Finally, we filter the content of the tweets by removing the stopwords and the URLs. In the end, we have the following corpus:

- 20,338 tweets by 1,579 sportive users
- 28,070 tweets by 1,433 non sportive users

To get more details about the number of tweets, we show the distribution of the users given the number of published and filtered tweets for the SU and the NSU on Figure 1. For the SU, we can see that a lot of users only have a few tweets and only a few users have a lot of tweets. This is due to us removing the exercise trackers tweets which greatly reduce the number of tweets for each sportive user. For the NSU, the distribution is different as we can see that an important part of the users either have a few tweets or more than 150 tweets. This explains why we get approximately 8,000 more tweets for the NSU.

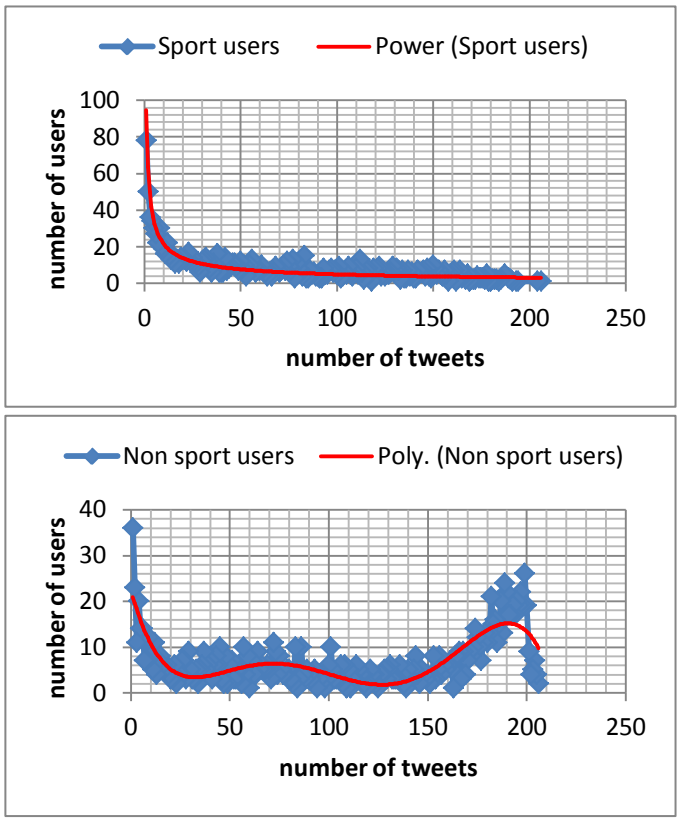

Figure 1: User distributions over the number of tweets for the SU and the NSU

III. METHODS

A. *Build a classifier with NLTK*

Using the large documentation made available by the authors of the Natural Language ToolKit, we build a Naïve Bayes classifier and a MaxEnt classifier. We then made several training run to estimate the training time, the accuracy, and the F1-measure of our classifiers and choose the best ones to build a voting classifier.

B. *Extract the features from the training corpus*

At first, we had two types of classifiers: a Naïve Bayes one and a MaxEnt one. To train these two types of classifiers, we had to extract features from the training dataset. The most

basic feature for text classification is unigrams and that is what we used in our training. Yet, as for the testing dataset, we removed the stopwords and the URLs from the tweets before extracting the features.

So how does the feature extraction work precisely? For each tweet from the Sentiment140 corpus, we extract the unigrams that are not in the stopwords set. Then, we concatenate all these unigrams to create our reference. When we want to classify one tweet, we extract the unigrams and keep only the ones that are known by the reference, we then pass the tweet to the trained classifier to get the classification. This method works but it is slow and likely to overfit because every word from the training dataset is recalled and that is an enormous amount of data to look over when it is done for two million tweets.

The solution that we have found to solve this and to avoid overfitting is to sort the words in the reference by number of appearances and to keep the *n* most frequent words as the new reference. This way, we avoid keeping words that only appear once or twice and that are just noises in the classification task, we also see our overall accuracy improve by removing the words that do not appear often from the reference. In Figure 3, we can see the 15 most frequent unigrams over the all Sentiment140 corpus.

After having implemented this solution, we also add bigrams to the features in order to solve some disambiguation and theoretically improve our overall accuracy. We train the classifiers and evaluate the accuracy and F1-measure with three varying parameters:

- the reference cutoff (i.e. the number of most frequent features to keep)
- the kind of features: only unigrams or unigrams + bigrams
- the amount of training data

The amount of training data has an impact on the training time and we wanted to make several experiments to show that more training data improve the accuracy. In the following figures, the nomenclature is cutoff-{NB,ME}-uni[bi] :

- the first part indicates the value of the cutoff (e.g. 1,000)
- the second part relates to the type of classifier: NB for Naïve Bayes and ME for MaxEnt
- the last part of the name describes what type of features have been used: uni for unigrams only and unibi for unigrams+bigrams.

C. *Evaluate the training resultsand build a voting classifier*

By making the previously defined parameters vary, we build 34 different classifiers with a top accuracy of 59%. We can see on Figure 2 that include bigrams as features dramatically increases the training time but does not improve the accuracy in a significant way. That's why the bigrams features are included in only one classifier: they do not improve the accuracy enough given the training time needed.

This figure also shows that for almost half the time, choosing a cutoff of 3,000 over a cutoff of 5,000 gives the same accuracy when training on all the dataset.

Finally, these experiments on the classifiers highlight the efficiency of the Naïve Bayes classifier over the efficiency of the MaxEnt classifier.

However, we have to note that the accuracy for the best

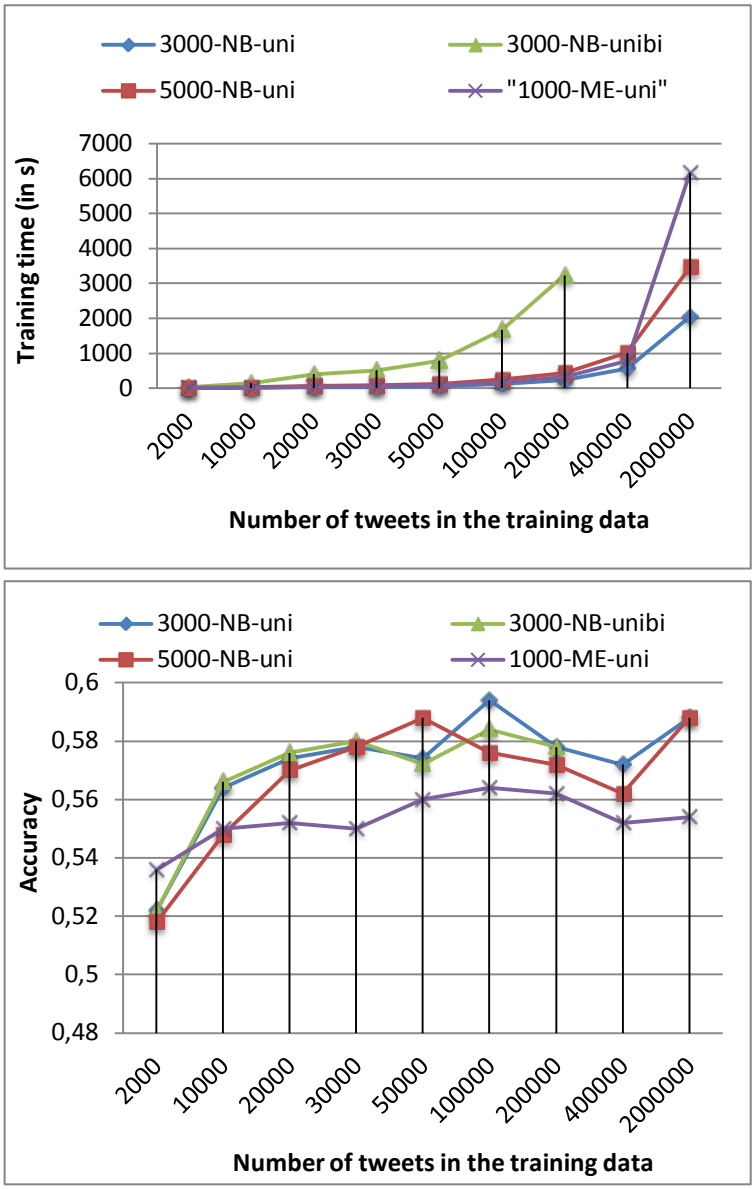

Figure 2: Training time (above) and accuracy (below) in function of the number of tweets in the training set.

Unigrams : not, good, day, get, like, today, work, love, going, got, lol, time, back, one, know

Bigrams : last night, good morning, wish could, feel like, don know, looking forward, looks like, good luck, getting ready, not good

Figure 3: the 15 most frequent unigrams and the 10 most frequent bigrams in the Sentiment140 corpus

classifier is only of 59% which is a poor result for a binary classifier even if the text classification task is not a simple task.

In order to slightly improve the process of classification, we build a vote classifier by taking the most "accurate" classifier for each of the Naïve Bayes case. Thus, we get three classifiers that, when given a tweet, are voting for positive or negative sentiment. Our vote classifier takes the class (positive or negative) that has the most votes and outputs it as a result. All of these operations are implemented in Python using NLTK.

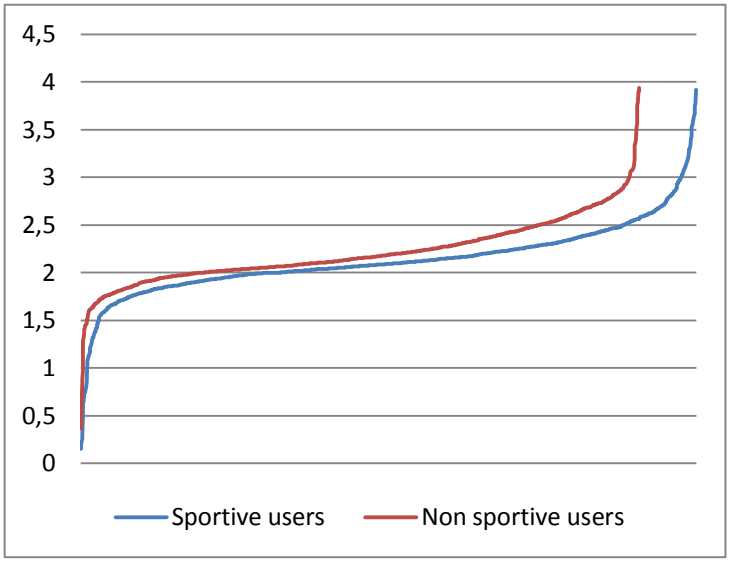

Figure 4: positivity scores for the SU and the NSU

IV. Experiment

Once the vote classifier is trained and the corpus of tweets is built, we can finally run our experiment which is to compute the sentiment classification for each tweet of each sportive and non sportive user. Then, we will be able to compute the average score of the SU and the NSU. To do so, we use a feature of NLTK that allows us to get the probability for an input to be in a given class. As our two classes are copied on the Sentiment140 classes, we have the positive class denoted by 4 and the negative class denoted by 0. Therefore we can compute a "positivity score" by multiplying the positive probability by 4. For example a tweet which have a probability of 0.6 to be in the positive class will get a score of $4 \times 0.6 = 2.8$.

For each user, we run the classifier over his tweets and compute his positivity score. We then sort the users from the less to the most positive one. The results are plot on Figure 4. We can clearly see that both the SU and the NSU follow the same trend with a lot of users with a neutral score ($1.5 \leq score \leq 2.5$) and only few users that are really negative or really positive. However, the NSU seems to be slightly more positive than the SU with an average positivity score of 2.21 against a score of 2.13 for the SU.

Thus, our hypothesis that sportive users would tend to be more positive is not supported by our experiment. These results even contradict our hypothesis as they show that the NSU are slightly more positive than the SU. Nonetheless, they are based on a sentiment classifier with a top accuracy of 59% which does not seem to be enough to conclude anything on the results.

V. Future Work

Therefore, a future work would consist of greatly improving the sentiment classifier to get an accuracy near 80% at least. Once this done, we might be able to conclude something about the results. A more ambitious work would be to actually implement a well-being classifier that would not be binary anymore but would introduce some more dimensions as in the ***Gallup Healthways Well-Being 5***.

Moreover, it would be interesting to detect sportive users in other ways than using the exercise trackers that limit the types of users we collect but by analyzing the stream of each user.

VI. Conclusion

To conclude, this project has generated a lot of work but the results are not as they were expected. However, it was a two-month project and a first project in the machine learning field, so we did not expect to have publishable results on the first try. A lot of time has been wasted in the research of the best library to use the Twitter API, the best toolkit to run machine learning algorithms but it has also allows us to learn a lot about classification tasks and about the machine learning community on the internet. Moreover, it was a motivating project that will be continued as described in the future work during the next semester in the hope to get better experiments and of course better results.

References

[1] Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N Project Report, Stanford* (2009): 1-12.

[2] Go, A., Huang, L., & Bhayani, R. (2009). Twitter sentiment analysis. *Entropy*, *17*.

[3] Roberts, K., Roach, M. A., Johnson, J., Guthrie, J., & Harabagiu, S. M. (2012). EmpaTweet: Annotating and Detecting Emotions on Twitter. In *LREC* (pp. 3806-3813).

[4] Deslandes, A., Moraes, H., Ferreira, C., Veiga, H., Silveira, H., Mouta, R., ... & Laks, J. (2009). Exercise and mental health: many reasons to move. *Neuropsychobiology*, *59*(4), 191-198.

[5] Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

[6] Jacob Perkins. 2010. *Python Text Processing with NLTK 2.0 Cookbook*. Packt Publishing.