Author Profiling: Predicting Age, Gender and Personality of Twitter Users

Daniel Rivera Ruiz
Courant Institute of Mathematical Sciences, New York University
New York 10012, USA
daniel.rivera@nyu.edu

Abstract—

Author profiling is an area of text analytics that tries to uncover characteristics of an author such as age, gender or native language based on the style of written text, i.e. the lexical or grammatical choices that are made by the author. In this paper we will apply author profiling techniques on *tweets*, to try to predict the age, gender and strongest personality traits of Twitter users.

Keywords— age, author profiling, gender, personality, spark, text analytics, twitter

I. Introduction

In recent years, social media and networking platforms have significantly contributed to the decision-making process in various domains. In 2017, an average of 500 million tweets per day were shared on Twitter. Accordingly, significant technical advancements have been made to process and analyze social media data using techniques from different fields such as machine learning, natural language processing and statistics.

Within the field of text analytics, author profiling deals with the task of analyzing a given number of texts to try to uncover various characteristics of the author based on stylistic- and content-based features.

In this paper we apply author profiling techniques in the social media domain. In particular, we try to predict three characteristics of Twitter users: age, gender and predominant personality traits. To achieve this, we project each tweet into a high-dimensional space where we try to embed several characteristics such as content, number of retweets, number of followers, number of hashtags, etc. We use this high-dimensional representation of the tweets to train a fully-connected artificial neural network to perform classification.

The results of our model assign to each tweet one category for every characteristic (age, gender, personality). For the age prediction model, four 10-year ranges were defined as the categories instead of considering each age individually.

Additionally, a list of 5 tweets that are considered to be similar is also provided. The similarity score between tweets is calculated simply as the normalized squared Euclidean distance in the high-dimensional space.

II. Motivation

Author profiling can be used in a number of applications. In marketing, companies are interested in knowing more about people who write positive or negative product reviews. Similarly, a company might want to find out what are the most relevant aspects that characterize its customers.

A quantitative analysis on social media (blogs, Twitter, Facebook, etc.) can provide meaningful insight and results that are directly related to trending applications such as personalized recommender systems and customized advertising.

In the particular case of Twitter, the data available is multimodal, containing text, images, and videos along with contextual and social metadata such as temporal and spatial information, as well as user connectivity and interactions. This user-generated data plays a significant role in making sense of public opinions and reactions to contemporary issues.

III. Related Work

The literature available related to data analytics on Twitter is vast. Over the years, extensive experimentation and analysis for insights have been carried out using Twitter data in various domains such as healthcare, public health, politics, social sciences, and demographics.

One of the main reasons why Twitter has become so popular to perform data analytics is the large amounts of data that are available, in comparison to other sources such as web forums or Reddit.

In [1], Kursuncu, Gaur, Lokala et al. discuss techniques, approaches, and state-of-the-art applications of predictive analysis of Twitter data. Specifically, they present fine-grained analysis involving aspects such as sentiment, emotion, and the use of domain knowledge in the coarse-grained analysis of Twitter data for making decisions and taking actions.

The predictive analysis paradigm for Twitter data discussed in [1] considers prediction as a process based on different levels of granularity. This paradigm contains two levels of analysis: fine-grained and coarse-grained. Fine-grained analysis aims to make tweet-level predictions on domain-independent aspects such as sentiment, topics, and emotions. On the other hand, coarse-grained analysis is used to predict the outcome of a real-world event, by aggregating and combining fine-grained predictions. In essence, low-level signals from tweets, such as sentiment, emotions, volume, topics of interest, location, and time frame, are used to make high-level predictions regarding real-world events and issues.

In addition to introducing the two-level predictive analysis paradigm, this paper also touches on processing and analytic techniques for handling Twitter data and provides details of feature extraction as well as machine learning algorithms. It describes common building blocks that can serve as the foundation for domain-specific predictive applications and discusses the impact of social media on the evolution of real-world events and actions.

In the context of this project, [1] was a valuable source of information mainly because of the fine-grained analysis techniques, the feature extraction process and the machine learning algorithms.

The study presented by Sushree, Ranjan, Mukesh and Santanu in [4] introduces a data processing architecture for massive quantities of real-time data in the context of stock prices prediction. The objective of this work is to develop a system that targets the major requirements for a good real-time data application, i.e. fault-tolerance, extensibility and scalability.

The main hypothesis of the paper is that the extensive flow of relevant information, such as stock prices on financial websites or users opinions on social media, can influence the market rates through leaps and bounds. Therefore, predicting the rates using this information makes it more meaningful since the continuous flow of data reshapes the model prepared from historical data, thus making it more accurate with each iteration.

To achieve this objective, the study uses a distributed architecture that excels at predicting data on the flow, using tools like Apache Flume and the Twitter Streaming API for data ingestion, Apache Spark MLlib and Stanford Core NLP for data processing, and Graph X for visualization. Throughout the stages of the process, Twitter data is analyzed to extract the sentiment of the users for a particular company using various machine learning techniques and the results obtained are stored in HDFS. Using these fault tolerant tools render the implementation highly scalable and robust.

The overall architecture introduced in this paper is fairly similar to the one proposed for the current project, with the main difference being that the original Twitter data for this project is retrieved using the standard search API instead of streaming. However, understanding the data flow and the processing steps of the stock market prediction model provided valuable insight and a good guideline to develop this application.

IV. Application Design

Figure 1 shows the flow diagram for the Author Profiling application using Twitter data. For each characteristic of interest, i.e. age, gender and personality, the main steps of the analytic are the following:

1) Collect and clean data (tweets) using the Twitter API.
2) Project tweets into high-dimensional space (feature selection and word embeddings using word2vec).
3) Split the data into training and validation sets.
4) Train the model (fully-connected ANN) on the training set.
5) Use the validation set to evaluate accuracy and tune hyper-parameters of the model.
6) Use tuned model to make predictions on unseen tweets and visualize results in the UI.

Figure 2 shows the UI for the application. The UI was designed with simplicity in mind: the user only has to introduce the URL of the tweet to be analyzed. In the background the relevant information for the tweet will be retrieved (using the Twitter API) to obtain its high-dimensional projection. The trained models will predict the three labels (age, gender, personality) for the tweet using this representation. Finally, the UI will present these predictions along with graphs showing the levels of confidence for each model. Additionally, the normalized squared Euclidean distance (which is proportional to the cosine similarity) is used to rank the top 5 similar tweets from the training set, which are made available for the user to explore.

V. Datasets

1) Twitter Data [2]

To retrieve tweets for the training and validation sets, the Twitter Standard Search API was used.

The first step of the collecting process was to query tweets with relevant phrases to the categories of the classification problem, e.g. *“I am XX years old”, “I am a boy/girl/man/woman”, “I am feeling happy/sad/excited”,* etc. By doing this, we were retrieving tweets for which the category was unequivocally determined (labeled data).

The next step was to query tweets by UserID, using the IDs from the users of the tweets in the previous step. The intuition behind this is that if a user tweets something like *“I am a girl…”* we can be positive that all tweets coming from the same UserID correspond to a female user. To make sure that the text of the tweet was indeed original content, we filtered out retweets from the training dataset. This technique was not used for the personality tweets, since it is not reasonable to conclude that a user who tweeted about being happy will continue to do so consistently.

The data extracted from each tweet followed the schema shown in Table 1. From this representation, the following transformations were performed:

a) Tokenize the status string.

b) Remove stop words from the tokenized status.

c) Replace each word in the status with its 300-dimensional embedding (unknown words were omitted).

d) Average the embeddings over all the words.

e) Normalize the numerical fields along the whole dataset.

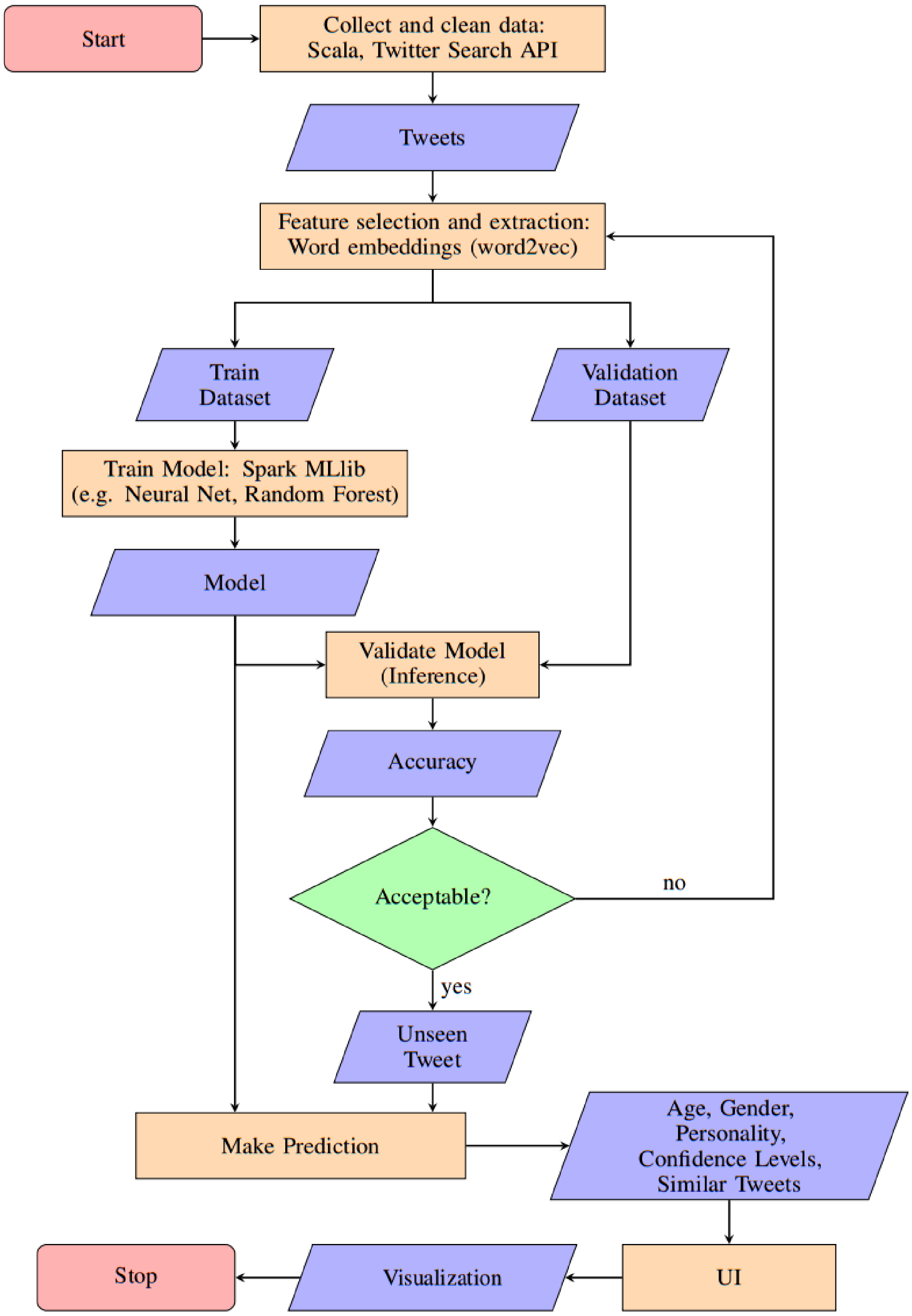
Start
Collect and clean data: Scala, Twitter Search API
Tweets
Feature selection and extraction: Word embeddings (word2vec)
Train Dataset
Validation Dataset
Train Model: Spark MLlib (e.g. Neural Net, Random Forest)
Model
Validate Model (Inference)
Accuracy
Acceptable?
no
yes
Unseen Tweet
Make Prediction
Age, Gender, Personality, Confidence Levels, Similar Tweets
UI
Visualization
Stop

Figure 1 Flow Diagram for the Author Profiling Application

After performing these steps, each tweet had a representation as a floating-point vector in a 309-dimensional space.

Field	Type
Status	String
FavoriteCount	Integer
HashTagsCount	Integer
MediaCount	Integer
MentionsCount	Integer
RetweetCount	Integer
URLCount	Integer
UserFavoriteCount	Integer
UserFollowersCount	Integer
UserStatusCount	Integer

Table 1. Schema of a Tweet

2) Pre-trained word vectors on the Google News dataset [3]

Word2vec is a tool originally developed by Google that provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. The *word2vec* tool takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

Generating the word vectors is a challenging and time-consuming task in itself. Therefore, Google published a pre-trained model that contains 300-dimensional vectors for 3 million words and phrases. The vectors were trained on part of Google News dataset (about 100 billion words).

Given the large size of the training dataset, these "out-of-the-box" embeddings usually yield good results when compared to the more specialized embeddings that can be generated by training a *word2vec* model using a relevant dataset.

For our experiments we opted to simply use the pretrained Google news vectors instead of generating our own model. In the (rare) case where a token from the tweets corpus was not present in the Google news corpus, the token was simply ignored.

VI. Remediation

The actionable insight extracted from our analytics process are the top 5 similar tweets presented to the user. This goes beyond the hard results of the quantitative analysis, which only present the confidence of the three classification models, and provides the user with information that is within the same domain as the input, i.e. Twitter data.

By looking at the most similar tweets, the final users will have a much better idea as to what kind of information the models are using to make decisions and try to use the same information to make (business) decisions themselves.

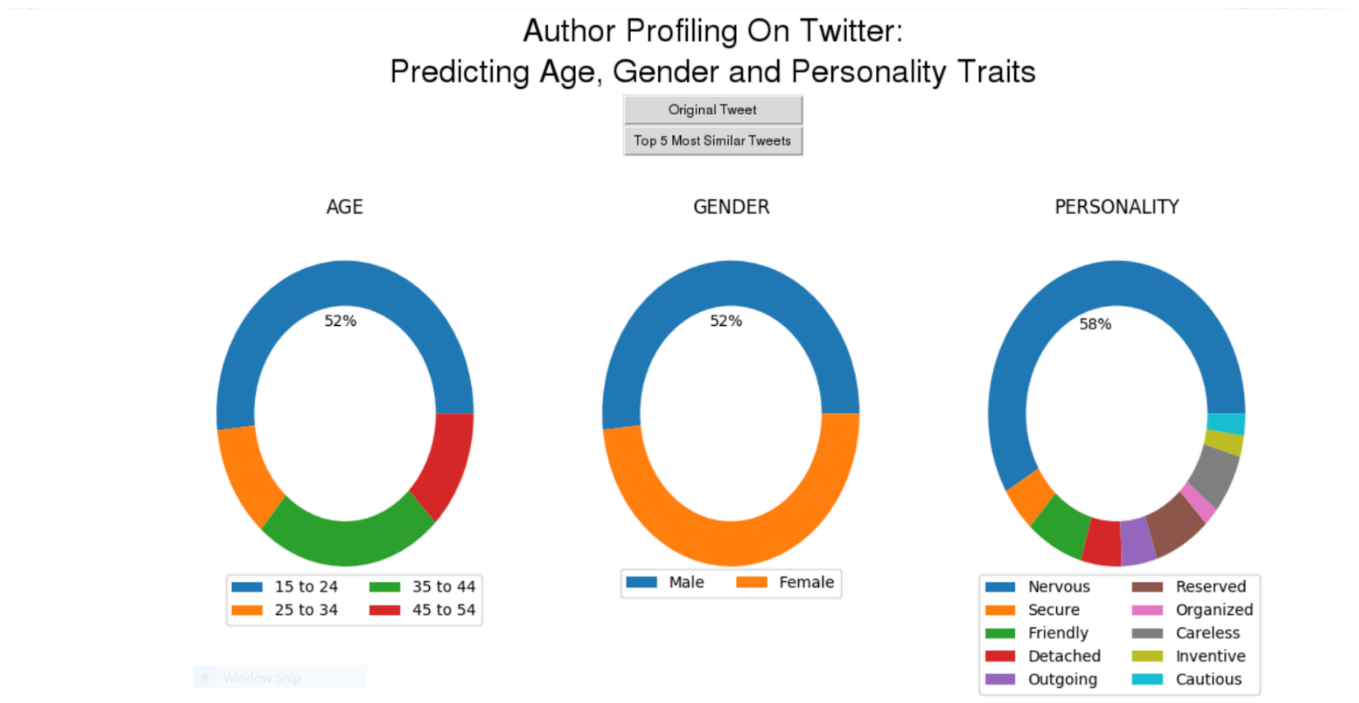

Figure 2. User Interface for the Author Profiling Application.

When the final users of the analytic access the UI, they only need to provide the URL of the tweet of interest as input and a button in the UI will allow them to visualize in a web browser the top 5 similar tweets identified by our algorithm. Under the hood, our application is effectively calculating the similarity of the input tweet against all the tweets used during training, sorting them according to this metric and returning the top 5.

VII. EXPERIMENTS

The data acquisition phase of this project was developed using the Twitter4J API [9] on top of plain Scala. Tweets for the three classification models were collected weekly and stored in JSON format on the CIMS Servers at NYU.

Once the data was ready to be processed, it was transferred to the HPC Dumbo Cluster local file system using Globus [10] and from there to HDFS. The pretrained word embeddings from Google news were downloaded in text format directly into Dumbo and then transferred into HDFS.

With all the required data available in HDFS, we developed the training phase of our application using Apache Spark's Machine Learning Library MLlib [5]. MLlib offers a wide range of popular ML algorithms to choose from, all of which integrate seamlessly within the pipeline of a Spark application. For this project we decided to use a Multilayer Perceptron Classifier, which is basically a fully-connected Artificial Neural Network. The architecture of the network was fairly simple, with only one hidden layer of 1,000 units. The input layer was 309-dimensional (as explained in section V) and the dimension of output layer variated depending on the classification task (4 classes for the age, 2 for the gender and 10 for the personality).

We split the input data collected from Twitter into training and validation sets in order to measure the accuracy of the models on previously unseen data. We used 80% of all the tweets for each task as training data and the rest as validation data. For the three models, we obtained validation accuracies of about 0.75. Finally, we stored the trained models and the training set' high-dimensional representation in HDFS.

In the last stage of the application, we use the trained models from the previous stage to perform inference on the tweet provided by the final user, and the high-dimensional representation of the training set to obtain the top 5 most similar tweets using normalized squared Euclidean distance.

One of the main limitations of our application is the response time of the inference stage. Since the MLlib pre-trained models are stored in a Spark-specific format, they must be loaded into a Spark application in order to be used for inference. Thus, each time the final user wants to analyze a single tweet, the following steps have to take place before they can get the final results of the analysis:

1) Submit a Spark job to the YARN cluster.
2) Load pre-trained models and training set into memory from HDFS.
3) Perform inference and calculate similarity.
4) Write results to local file system.
5) Run the UI using the results obtained in 4).

To improve this, either the Spark application would have to be running continuously (which is highly inefficient and not really feasible in a shared-resources environment such as Dumbo) or the model would have to be exported into a format that is compatible with other tools (e.g. Java or Python). Given the time we think that exploring the second option is really worth the while, since it will provide a response to the user queries much faster with a simple Java/Python application running in the back-end instead of submitting Spark jobs to a YARN cluster every time.

VIII. CONCLUSION

In this paper we developed an Author Profiling system that predicts the age, gender and personality traits of Twitter users. To achieve this, we used information from the users' tweets to capture their writing style and also metadata from their Twitter accounts such as number of followers, number of retweets, number of status updates, etc.

The main target of the application is to provide a prediction (and the level of confidence) for the age, gender and personality trait of a user given one of their tweets as the input. Additionally, it also provides the top 5 most similar tweets in the training set. This information can be very valuable to better characterize Twitter users to target customized advertising campaigns, understand their opinions and interactions, etc.

IX. FUTURE WORK

We believe that generating a bigger, better-curated dataset can help improve the results obtained by the ML algorithms. To achieve this, it would probably be necessary to use the Twitter Premium or Enterprise APIs, which provide much more functionalities than the standard one.

With larger amounts of data, it would also be worth exploring the generation of one single model to predict all three classes (age, gender, personality) at once. It is usually the case that one global model performs better than its partial counterparts. The main challenge, however, would be to come up with a training dataset that is accurately labeled for the three tasks at once.

Finally, with the more advanced features provided by the Premium or Enterprise Twitter APIs, it would be possible to migrate the application architecture from RESTful to Streaming, which would considerably improve the end-to-end flow of information and reduce (or even eliminate) the need of human intervention for data transfers.

ACKNOWLEDGMENT

We would like to thank Twitter for making all the data used for this project available through the Twitter Search API. Also, to professor Suzanne McIntosh, for her invaluable guidance and feedback throughout the semester. Finally, to the HPC team at

New York University for making available to us the Dumbo Cluster, where the entirety of this project was developed and tested.

REFERENCES

1. **Kursuncu, U., Gaur, M., Lokala, N. U. Ga., Thirunarayan, K., et al.** (2019): Predictive Analysis on Twitter: Techniques and Applications. *Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining*, pp. 67-104.
2. **Twitter**. (2019): Search Tweets. Available at https://developer.twitter.com/en/docs/tweets/search/overview.html
3. **Google Code.** (2013): Word2vec. Available at https://code.google.com/archive/p/word2vec/
4. **Sushree D., Ranjan K. B., Mukesh K., Santanu K. R.** (2018): Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction. *Procedia Computer Science*, vol. 132, pp. 956-964.
5. **Apache Spark 2.3.0.** (2019): Machine Learning Library (MLlib) Guide. Available at https://spark.apache.org/docs/2.3.0/ml-guide.html
6. **Scala 2.11.12.** (2019): Scala Library API. Available at https://www.scala-lang.org/api/2.11.12/#package
7. **Karau, H., Konwinski, A., Wendell, P., Zaharia, M.** (2015): *Learning Spark: Lightning-Fast Big Data Analytics*, O'Reilly Media, Inc.
8. **Alexander, A.** (2013): *Scala Cookbook: Recipes for Object-Oriented and Functional Programming*, O'Reilly Media, Inc.
9. **Twitter4J** (2019). An unofficial Java library for the Twitter API. Available at http://twitter4j.org/en/
10. **Globus** (2019). Data transfer with Globus. Available at https://www.globus.org/data-transfer