Towards Predicting Mentions to Verified Twitter Accounts: Building Prediction Models over MongoDB with Keras

Ioanna Kyriazidou[1], Georgios Drakopoulos[2], Andreas Kanavos[1], Christos Makris[1] and Phivos Mylonas[2]

[1]*CEID, University of Patras, Patras, Hellas, Greece*

[2]*Department of Informatics, Ionian University, Kerkyra, Hellas, Greece*

Keywords: Social Analytics, Digital Influence, Digital Trust, Verified Account, Document Databases, MongoDB, Convolutional Network, Logistic Regression, Keras, Pandas, TensorFlow.

Abstract: Digital influence and trust are central research topics in social media analysis with a plethora of applications ranging from social login to geolocation services and community structure discovery. In the evolving and diverse microblogging sphere of Twitter verified accounts reinforce digital influence through trust. These typically correspond either to an organization or to a person of high social status or to netizens who have been consistently proven to be highly influential. This conference paper presents a framework for estimating the probability that the next mention of any account will be to a verified account, an important metric of digital influence. At the heart of this framework lies a convolutional neural network (CNN) implemented in keras over TensorFlow. The training features are extracted from a dataset of tweets regarding the presentation of the Literature Nobel prize to Bob Dylan collected with the Twitter Streaming API and stored in MongoDB. In order to demonstrate the performance of the CNN, the results obtained by applying logistic regression to the same training features are shown in the form of statistical metrics computed from the corresponding contingency matrices, which are obtained using the pandas Python library.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

At the dawn of the big data or 5V era online social media constitute a major driver of research and marketing alike. The main reason it that social media abound with interactions between accounts from a wide array of multimodal options ranging from hashtags to affective states, geolocation information, and live video streams. All these alternatives allow the real time mining of invaluable knowledge regarding numerous topics of social interest. Most importantly, regardless of the level of technological sophistication or the type of data involved, these interaction types ultimately serve as vehicles for communication. And along with communication come trust and digital influence, either explicit or latent.

A major factor towards establishing trust in the digital sphere is determining whether behind a given social network account lies an actual real world entity, whether an individual, a hacker group, a multinational conglomerate, or any other type of formal or informal organization. Among the most recent cases where this factor was questioned was the hunt undertaken in 2017 by the journalists of *Counterpunch* to approach the alleged freelance reporter Alice Donovan(www.counterpunch.org, 2017). Moreover, arguments for establishing the identity of an account owner were put forward during the 2013 public discussions regarding the alleged activities of Twitter eggs, which eventually led to the dropping of the famous egg avatar, a joking reference to the bird logo, in favor of a gender neutral avatar(www.theguadian.com, 2017).

In the aftermath of these discussions Twitter undertook the initiative of introducing a distinction between its own accounts in order to reinforce the credibility of a subset of accounts as well as of the tweets from them. Specifically, Twitter accounts are divided into the following categories:

- Verified accounts are selected by Twitter and correspond to significant real world entities such as governments, sports groups, and academic institutions.
- Unverified accounts are ordinary accounts and constitute the overwhelming majority of the Twitter ecosystem.

Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

The primary contribution of this conference paper is a framework for predicting whether the next mention of a Twitter account, whether verified or not, will be to a verified account. This framework is based on a convolutional neural network (CNN) implemented on TensorFlow using keras as a front end in Python. The mix of structural and functional training features was exctracted from Twitter in JSON format with Twitter4j over Java and stored in MongoDB, which is ideal for natively storing and processing documents following this format. For comparison purposes a logistic classifier was also applied on the same dataset using Python and scikit-learn. The results indicate that the CNN, trained under three different scenaria, outperforms this classifier in terms of a number of statistical metrics derived from the contingency matrix, most prominently accuracy, precision, and type I and II error rates.

The remaining of this work is structured as follows. Section 2 briefly summarizes recent scientific literature regarding social network analysis. The architecture of the Twitter topic sampler is described in section 3. Section 4 presents the fundamentals of the two prediction models used in this work. The relationship between influence, trust, and community structure in the digital sphere as well as the intuition behind the training features are examined in section 5, while in section 6 the results obtained from these models are analyzed. Section 7 recapitulates the main findings and outlines future research directions. Vectors are displayed in lowercase boldface and scalars in lowercase. Finally, the notation of this work is summarized in table 1.

Table 1: Notation of this conference paper.

Symbol	Meaning
$\triangleq$	Definition or equality by definition
$\{s_1,\ldots,s_n\}$	Set with elements $s_1,\ldots,s_n$
$\lvert S\rvert$	Set cardinality
$\Phi(t_k)$	Set of accounts following t_k
$\Psi(t_k)$	Set of accounts followed by t_k
$\mathrm{Var}[f]$	(Deterministic) variance of feature f
$\mathbf{I}[P]$	Indicator function for predicate P

2 PREVIOUS WORK

In the digital sphere influence revolves around the fundamental dynamics of the relationship between two or more accounts comprising a formal or informal group and essentially pertains to why, how, and when the online behavior of a subset of accounts is imitated by the remaining ones in that group (Russell, 2013)(Gilbert and Karahalios, 2009). There is a plethora of tools for representing and assessing in context the significance of these imitation patterns including set theoretic similarity metrics such as the Tversky coefficient (Tversky, 1977), which in contrast to the Tanimoto coefficient is asymmetric in the sense that one set is considered the template and the other assumes the role of the variant with the two roles not being interchangeable. Estimating the cardinality of large sets may not be always easy, thus heuristics based on the HyperLog method have been developed (Drakopoulos et al., 2016b). The algorithmic cornerstone for evaluating influence, digital or otherwise, is the concept of meme, approximately the cultural counterpart of a gene (Blackmore, 2000).

The associated concept of trust is more difficult to model algorithmically, as its has no discernable digital traits. Nonetheless, there have been approaches for inferring trust through natural language processing or matrix factorization techniques combined with the silent but inherent trust transitivity (Jamali and Ester, 2010). The latter was identified as an important factor in software engineering and computer security well before the advent of online social media (Thompson, 1984). An effort to model Web trust with an application to Semantic Web has been presented in (Mislove et al., 2007). Other trust models have been proposed in (Golbeck, 2005), in (Golbeck and Hendler, 2006) which also presents a trust inferrence method, and in (Golbeck et al., 2006) where trust is used as the basis for movie recommendation. An alternative approach to trust would be to rely on the fact that it is ultimately rooted in human emotion. As such, affective computing techniques such as the ones proposed in (Muller, 2004) and in (Pang and Lee, 2008) attempt to discover community structure in social media. For an overview of affective computing see (Picard et al., 2001) or (Picard, 2003).

Given that both influence and trust can be time evolving, it makes sense to model account-account-time trust triplets with third order tensors and apply higher order clustering techniques in order to determine account groups which bond and consolidate or decay over time (Papalexakis and Faloutsos, 2015)(Papalexakis et al., 2014)(Drakopoulos et al., 2019). Tensors have been already applied to network-specific problems, for instance for aligning size with tweet and retweet activity in Twitter (Drakopoulos et al., 2018) and for extending in PubMed the term-document vector query model to a term-keyword-document one (Drakopoulos et al., 2017a).

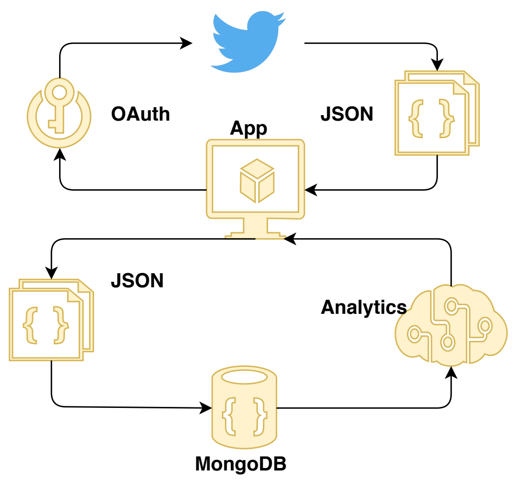

Figure 1: System architecture.

3 ARCHITECTURE

In order to implement the proposed analytics of section 4 a new system written in Java in NetBeans 8.1 has been implemented. Its main components are the Twitter4j library which provides a Twitter API for collecting data such as account names, followers, hashtags, tweets, and retweets, a MongoDB database for managing and querying this information in JSON format, and the prediction models, which have been developed in Python 3.6. Figure 1 displays the architecture for collecting and analyzing Twitter data.

The primary reason for developing our own client is the ability to extend it through more Twitter analytics, in the form of either Pythom modules or Java jars, in order to facilitate more experiments. A fundamental design concept of Twitter is that it is to be integrated in other applictions. To this end, it provides a dashboard for authenticated developers to rapidly create and register applications.

Twitter4j[1] is an open source Java library for harnessing Twitter information. Among its other capabilities it provides integration with OAauth for efficient check of the four Twitter authentication tokens, two for the developer (consumer key, secret key) and two (access token, secret token) for the application itself, as well as with Maven in order to automatically satisfy dependencies. Calls to the appropriate Twitter4j methods have been placed in the source code of our social media crawler. In general, there are two main operational modes for such a crawler, namely account and topic sampling. The latter was selected since we are interested in assessing a functional Twitter feature.

[1]www.twitter4j.org

MongoDB is a mainstay of the NoSQL movement and one of the most popular document databases designed to store JSON formatted documents in a schemaless manner and process among others ad hoc queries, indexing, and real time aggregation. Its architecture is distributed, allowing horizontal and geographical scaling if needed, the former achieved through key shrading. Currently transactions are not implemented, leaving data consistency and replication policy enforcement to the local administrator, leading to the development of a number of third party consistency control tools. MongoDB supports drivers for most of the popular programming languages including Python, C++, and Java.

Finally, JSON is an open Internet standard described in RFC 8259. Originally developed for asynchronous server-client communications, it was widely adopted for formally describing in human readable form documents. Its primary data structure is the associative array, consisting of key-value pairs where the values may belong to different primitive types including strings. Document databases such as MongoDB natively support storing and iteratively or recursively querying JSON structs.

4 PREDICTION MODELS

4.1 Logistic Regression

Logistic regression appears in a number of cases in economics, statistics and engineering and it is a generalization of the least squares regression. Recall that the latter estimates in the presence of noise with a known distribution the functional parameters of an n-th order model

$$\mathbf{p} \stackrel{\triangle}{=} \begin{bmatrix} \vartheta_0 & \vartheta_1 & \ldots & \vartheta_{n-1} \end{bmatrix}^T \tag{1}$$

are based on a set of m observation vectors

$$\mathbf{v}_k \stackrel{\triangle}{=} \begin{bmatrix} 1 & x_{0,k} & x_{1,k} & \ldots & x_{n-1,k} & y_k \end{bmatrix}^T \tag{2}$$

where each vector contains observations of the n input or independent variables $X_0, \ldots, X_{n-1}$ and the corresponding value of the output or dependent variable Y. Thus, the possibly over- or underdetermined linear system of equation (3) is formed

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{m-1} \end{bmatrix} \mathbf{p} + \mathbf{w} \tag{3}$$

where $\mathbf{w}$ is a noise vector of known distribution. Typically the noise and the model parameters are assumed

Table 2: NoSQL database types.

Type	Abstract type	Formal description	Prominent software
document	formatted document	JSON, BSON, XML, YAML	MongoDB, CouchDB, OrientDB
key-value	associative array	JSON, BSON, YAML	Riak, Amazon Dynamo, Redis
column family	wide columns	JSON, BSON, XML, YAML	Apache Cassandra, KeySpace
graph	property graph	JSON-LD, RDF, ontologies	Neo4j, TitanDB, Sparksee

to be uncorrelated. A more detailed formulation of the same system is that of equation (4).

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \end{bmatrix} = \begin{bmatrix} 1 & x_{0,0} & \dots & x_{n-1,0} \\ 1 & x_{0,1} & \dots & x_{n-1,1} \\ \vdots & \ddots & \ddots & \vdots \\ 1 & x_{0,m-1} & \dots & x_{n-1,m-1} \end{bmatrix} \begin{bmatrix} \vartheta_0 \\ \vartheta_1 \\ \vdots \\ \vartheta_{n-1} \end{bmatrix} + \mathbf{w} \tag{4}$$

Binomial logistic regression imposes a logistic transformation to the observations using a single output variable. Moreover, the latter is assumed to be binary, essentially a Bernoulli trial, and logistic regression can compute the probability that the output variable can take either value or, alternatively, it estimates the model parameters such that

$$y_k = \begin{cases} 1, & \mathbf{p}^T \mathbf{v}_k + \eta_k \geq 0 \\ 0, & \mathbf{p}^T \mathbf{v}_k + \eta_k < 0 \end{cases} \tag{5}$$

The logistic probability density function is defined as:

$$\begin{aligned} f_X(x;\mu_0,\sigma_0) &\stackrel{\triangle}{=} \frac{e^{-\frac{x-\mu_0}{\sigma_0}}}{\sigma_0\left(1+e^{-\frac{x-\mu_0}{\sigma_0}}\right)^2} \\ &= \frac{1}{\sigma_0\left(e^{\frac{x-\mu_0}{\sigma_0}}+e^{-\frac{x-\mu_0}{\sigma_0}}\right)^2} \\ &= \frac{1}{4\sigma_0}\operatorname{sech}^2\left(\frac{x-\mu_0}{2\sigma_0}\right) \end{aligned} \tag{6}$$

where the hyperbolic secant is in turn defined as:

$$\operatorname{sech}(\beta_0 x) \stackrel{\triangle}{=} \frac{1}{\cosh(\beta_0 x)} = \frac{2}{e^{\beta_0 x}+e^{-\beta_0 x}} \tag{7}$$

Alternatively, the hyperbolic secant function can be defined in terms of the power series:

$$\operatorname{sech}(\beta_0 x) \stackrel{\triangle}{=} 1 + \sum_{k\equiv 0 \bmod 2}^{+\infty} \frac{E_{k/2}}{k!}(\beta_0 x)^k \tag{8}$$

where E_k is the k-th secant Euler number[2].

[2]OEIS integer sequences A046976 and A046977.

4.2 Convolutional Neural Network

In addition to the logistic classifier, three convolutional neural networks (CNNs) were created using keras acting as the front end of TensorFlow. The four stages of any deep learning model in TensorFlow are:

- Definition. At this stage the architecture of the network including the number of neurons at each layer, the connectivity between each layer, and the number and location of the feedback loops. The latter are important in determining the memory of the classifier.
- Translation. This is achieved with the selection of the loss function and the call of the **compile** method which handles model setup.
- Training. Fitting is done with the **fit** and **evaluate** methods.
- Prediction. The model generates actual predictions based on the **predict** method.

The architecture of the network is defined by the following template, which is used for each new layer of processing neurons:

```
layers = [Dense(3)]
model = Sequential(layers)
model.add(Dense(n, input_dim=m,
  init='uniform', activation='softplus'))
```

The **model.add()** method adds an additional hidden or output layer which can be fully, densely, or sparsely connected with the previous one. The synaptic weights were initialized with a uniform distribution in $[0,1]$. Four activation functions were considered, the sigmoid, the softmax, the rectifier, and the softplus. The sigmoid is defined as:

$$g_s(s;\beta_0) \stackrel{\triangle}{=} \frac{1}{1+e^{-\beta_0 s}} = 2\tanh\left(\frac{\beta_0 s}{2}\right) - 1 \tag{9}$$

The softmax function provides a ranking for the elements of any real valued data vector

$$\mathbf{x} \stackrel{\triangle}{=} \begin{bmatrix} x_0 & x_1 & \dots & x_{n-1} \end{bmatrix}^T \tag{10}$$

by assigning to each individual $\mathbf{x}$ the score:

$$g_m(x_k;\beta_0) \stackrel{\triangle}{=} \frac{e^{-\beta_0 x_k}}{\sum_{j=1}^{n} e^{-\beta_0 x_j}} \tag{11}$$

The rectifier or ramp activation function is defined as:

$$g_r(s;\beta_0) \triangleq \max\{0, |\beta_0|s\} = \begin{cases} |\beta_0|s, & s \geq 0 \\ 0, & s < 0 \end{cases} \quad (12)$$

In many scenarios is used a smoother version of g_r termed the softplus function defined as:

$$g_p(s;\beta_0) \triangleq \ln\left(1 + e^{\beta_0 s}\right) \quad (13)$$

The softplus function has two interesting properties with respect to the network training process. First, it avoids the vanishing gradient problem, which causes the saturation of a number of synaptic weights in large neural networks. Moreover, it is the antiderivative of the sigmoid function:

$$g_p(s;\beta_0) = \frac{1}{\beta_0}\int_{-\infty}^{s} g_s(\tau)d\tau, \quad \beta_0 \neq 0 \quad (14)$$

This implies that a softplus value essentially comprised of a broad spectrum of sigmoid values, which is less prone to instantaneous spikes caused by an isolated training sample. This results to a smoothed trajectory of the weights in the parameter space, rendering the training process more robust.

Eventually, g_p was used in all hidden layers, whereas the logistic activation function was placed in the output layer. The number of layers, including the input and the output layers, was fixed to $[11 : 17 : 11 : 1]$. Initially, the first hidden layer maps the input features in a 50% larger space, which is big enough to offer sufficient discrimination between features which are very close in the original space, but also small enough to avoid overfitting. Then, the transformed features are mapped back to the original space. Finally, a logistic regression is applied to these new features.

The cost function to be minimized was selected to be the binary cross-entropy, which is suitable as its name suggests for binary classification problems, using for training the gradient descent Adam algorithm, which is an extension of the RMSprop and the AdaGrad algorithms by maintaining per-parameter learning rates and using an averaged second moment to update the learning rates. Adam is well defined for large and sparse networks and has low memory requirements. In the source code this choice was declared as:

```
model.compile(optimizer='adam',
  loss='binary_crossentropy',
  metrics=['accuracy'])
```

Following that, the dataset is split into two parts. The training part consists of the variables train_X and train_y , which contain the features and the output variable respectively. Along the same line of reasoning, the test part comprises of the variables test_X and test_y . Once the CNN is trained, its performance is evaluated, and predictions are generated. The training phase consists of J epochs, where in each epoch Q training vectors are driven to the CNN. In the source code:

```
model.fit(train_X, train_y
  nb_epoch=J, batch_size=Q)
model.evaluate(test_X, test_y)
p = model.predict(test_X)
r = [round(s) for s in p]
```

The training of the CNN plays a crucial role in shaping its generalization capability. In general, a golden spot between presenting too few input vectors, in terms of feature variability, or too many distinguishes a properly trained CNN from either a CNN which cannot capture all the dimensions of the feature space or a CNN operating effectively like a dictionary. To this end, the following three scenarios were selected with respect to the parameters J and Q.

- $J = 12500$ and $Q = 1000$ (CNN1 in table 5)
- $J = 1250$ and $Q = 100$ (CNN2 in table 5)
- $J = 150$ and $Q = 100$ (CNN3 in table 5)

Finally, in order to obtain a better understanding of the predictor performance, the contingency matrix is generated using the pandas_ml library.

```
from pandas_ml import contingencyMatrix
cm = contingencyMatrix(test_y, r)
cm.print_stats()
```

5 TRAINING FEATURES

In this section the intuition behind selecting the training features of the prediction model is given. The latter is based on the properties of trust, influence, and community structure. Additionally, the importance of introducing verified accounts is analyzed. The latter is rooted in an axiom common in Twitter, and in any social medium for that matter, stating that:

Axiom 1. *Accounts are Created Equal but Hardly Remain So.*

This can be attributed to a number of factors including topical variety in the form of hashtags, tweeting frequency, response ratio, or elevated status deriving directly from the real world entity or person behind the account. For instance, a Twitter account of an established newspaper is typically considered more reliable and far less prone to fake news posting than that of an individual netizen. Qualitative metrics for assessing digital influence in Twitter have been proposed in (Drakopoulos et al., 2016a).

Establishing digital trust is quite complex as it involves a number of psychological variables besides multimodal interaction. Additionally, it may need to be re-verified over time, as for instance when a two factor authentication is required when a trusted account appears to be active from an unknown location. At any rate:

Axiom 2. *Trust Implies Communication.*

Once trust is established between a group of two or more accounts, it usually takes one out a small number of behavioral forms sharing imitation as a key concept. One option is that an account takes a leading role and the others tend to imitate its actions, perhaps with a varying time lag and some minor alterations. Clearly in this case the leader enjoys more trust than the remaining accounts and the trust relationships form a star with some possible edges connecting non-leading accounts.

Another alternative is the peer group where each account enjoys a comparable amount of trust from the other peer accounts. Again, in this case behavioral correlations tend to appear, although the variance in time lag may be initially larger since each peer is possible to do some fact checking by comparing the behavior of its peers. However, once a critical size of peers has performed the same action, the others will imitate these pioneering peers with a very short time lag and with much smaller variance. Thus, depending on the strength of trust connections, in peer groups imitation might be a two-phase phenomenon. The trust relationships in this scenario tend to form a cycle with many cross-connecting edges, or at the extreme case a clique.

An intermediate case emerges when there is a leading subgroup within the trust group. In that case time lags tend to be somewhat smaller, since imitation is quick both inside the leading subgroup and between the latter and the remaining group. Functionally, the leading subgroup plays approximately the role of the vertex cover as memes tend to be copied quickly from the center to the periphery.

As mentioned earlier, the key in all three cases is imitation with a varying time lag. Thus, delayed correlations are bound to appear in the features of the training set. Therefore, a good classifier is expected to exploit this critical property by having even a memory of some kind. This is a significant difference between the logistic regressor, whose coefficients might contain a limited reflection of these correlations, and the convolutional neural network, whose structure and training process facilitates memory at the expense of course of a more complicated and slow training procedure.

As a sidenote, convolutional networks is not the only class of classifiers with memory. The ordinary feedforward neural networks assimilate features of the training set in their synaptic weights, Kohonen networks or self organizing maps rely on the spatial clustering of their processing units as a form of memory as a result of a Hebbian training process, tensor stack networks depend on the cross training of a forest of neural networks, whereas the models relying on Volterra kernels incorporate memory in the number and lags of the kernel indices as follows:

$$g(x) \approx \sum_{k=1}^{p} \sum_{j=1}^{p_k} h\left[i_{j,1}, \ldots, i_{j,k}\right] \prod_{s=1}^{k} g[n - i_{j,s}] \qquad (15)$$

One more fundamental axiom is:

Axiom 3. *Influence Implies Trust.*

This element is important in its own right, but also allows the indirect determination of trust through influence. This methodology has been employed in social network analysis in order to inferr the select the proper cluster size distribution out of a number of possible ones generated by various community discovery algorithms (Drakopoulos et al., 2017c)(Drakopoulos et al., 2017b).

The data for the prediction models consist of 5000 rows in total, where 1250 of these comprise the training set and the remaining 3750. Each row contains features extracted from a single tweet, where every tweet pertains to presenting the Literature Nobel to Bob Dylan, a very popular topic at the time the dataset was collected.

In order to construct the prediction model, a mix of structural and functional features will be used. Moreover, the feature set has been augmented with semantic information: The eleven top trending hashtags, shown in table 3, for the day the dataset was being collected have been retrieved. This was done on the grouds that they typically carry more semantic information in comparison to an ordinary word in a tweet, in a relationship similar to the one between the index and the ordinary terms in a document (Drakopoulos et al., 2017a). Out of them only #NobelPrize, #BobDylan, #literature, and #Nobel were kept as the frequency of the remaining hashtags was very low.

Thus each row consists of eleven features (f_1 to f_{11}) plus the value of the output indicator (y):

- f_1-f_4: Four individual binary indicators, each corresponding to whether one of the selected hashtags is present in the tweet.
- f_5: The number of followers $\Phi(k)$ of the account.

Table 3: Percentage of appearance of the top trending hashtags.

NobelPrize	**medicine**	**PremioNobel**	**BobDylan**	**AliceMunro**
78.86%	0%	14.53%	89.35%	0%
literature	**malala**	**Nobel**	**peace**	**physics**
56.53%	0%	66.12%	0.04%	0.02%

- f_6: The number of followees $\Psi(k)$ of the account.
- f_7: The number of tweets of the account.
- f_8: The tweet id.
- f_9: The number of mentions in the tweet.
- f_{10}: Is the poster verified?
- f_{11}: Is a verified account mentioned in the tweet?
- y: Is the mention towards a verified account?

Since the tweet id takes large values compared to the boolean indicators, the latter were rescaled by multiplying them by 1000 in order to keep the features at the same range. This contributes to the acceleration and the stability of the training process. Additionally, the original dataset is balanced in terms of value variability, facilitating the easy partitioning to training and test sets as shown in table 4. If a feature f_k is an indicator, then the number of rows which is non-zero $|\mathbf{I}[f_k] = 1|$ is shown, otherwise its deterministic variance $\mathrm{Var}[f_k]$ is shown. Also the output variable y is treated as an indicator feature. From the values of table 4 it follows that the partitioning of the original dataset is correct in the sense of having comparable feature variability in both the training and the test sets.

6 RESULTS

The contingency matrices for the four predictors are shown in table 5. Additionally, in the same table the values of a number of statistical metrics based on the contingency matrix are shown as well as the training time for each predictor. In the case of the logistic regressor, the time for computing the regression coefficients is used. Recall that, **tp**, **tn**, **fp**, and **fn** stand for true positives, true negatives, false positives, and false negatives respectively as is customary in data mining literature.

The most significant statistical metrics which can be directly computed from any contingency matrix include:

- Accuracy (**ac**): Measures how many mentions are properly classified, regardless of whether an account is verified or not.
- Specificity (**sp**): Measures how many of the mentions to unverified accounts are actually discovered.
- Precision (**pr**): Measures how many of the mentions classified as relevant to verified accounts are actually such.
- Sensitivity or recall (**rc**): Measures how many of the metions to verified accounts are actually discovered.
- False positive rate or type I error rate (**t1**): Measures how many of the mentions to unverified accounts are misclassified.
- False negative rate or type II error rate (**t2**): Measures how many of the mentions to verified accounts are misclassified.
- Negative predictive value (**npv**): Measures how many of the mentions classified as relevant to unverified are actually such.
- The Mattthews correlation coefficient or phi coefficient (**mcc**): Measures the agreement between the predicted values and the actual ones. It works even when the number of samples from each category are unevely represented in the dataset.

Also, the latter is defined as:

$$\mathrm{mcc} \triangleq \frac{\mathrm{tp}\,\mathrm{tn} - \mathrm{fp}\,\mathrm{fn}}{\sqrt{(\mathrm{tp}+\mathrm{fn})(\mathrm{tp}+\mathrm{fn})(\mathrm{tn}+\mathrm{fp})(\mathrm{tn}+\mathrm{fn})}} \tag{16}$$

Additional metrics which can be defined in terms of the ones presented here are the F1 (**f1**), the informedness (**if**), and the markedness (**mk**) which are respectively defined as:

$$\begin{aligned} \mathrm{f1} &\triangleq \frac{2}{\frac{1}{\mathrm{rc}} + \frac{1}{\mathrm{pr}}} \\ \mathrm{if} &= \mathrm{sn} + \mathrm{sp} - 1 \\ \mathrm{mk} &= \mathrm{pr} + \mathrm{npv} - 1 \end{aligned} \tag{17}$$

The values of the contingency tables summarized in table 5 can be interpreted as follows. Regarding the CNN training process, the less training rows the network is presented, the more accurate and precise it becomes at the expense of recall. This might imply that CNN can develop a limited generalization capability for a low number of training rows. Moreover, the type I and II error rates increase with the number

Table 4: Dataset partition.

set	$\lvert\mathbf{I}[f_1=1]\rvert$	$\lvert\mathbf{I}[f_2=1]\rvert$	$\lvert\mathbf{I}[f_3=1]\rvert$	$\lvert\mathbf{I}[f_4=1]\rvert$	$\mathrm{Var}[f_5]$	$\mathrm{Var}[f_6]$
train	591 (47.28%)	630 (50.41%)	636 (50.88%)	621 (49.68%)	1216.24	2264.11
test	1904 (50.77%)	2003 (53.41%)	1822 (48.58%)	1852 (49.38%)	1693.44	2253.43
	$\mathrm{Var}[f_7]$	$\mathrm{Var}[f_8]$	$\mathrm{Var}[f_9]$	$\lvert\mathbf{I}[f_{10}=1]\rvert$	$\lvert\mathbf{I}[f_{11}=1]\rvert$	$\lvert\mathbf{I}[y=1]\rvert$
train	64.11	399812	37.21	367 (29.36%)	663 (53.04%)	628 (50.24%)
test	53.43	482703	33.32	1102 (29.38%)	1897 (50.58%)	1925 (51.33%)

Table 5: Contingency matrices, metrics values, and predictor training time (sec).

model	**tp**	**tn**	**fp**	**fn**	**ac**	**sp**	**pr**	**rc (sn)**
logistic	1285	1236	742	487	0.6727	0.6248	0.6639	0.7251
CNN1	1438	1316	535	461	0.7344	0.7109	0.7288	0.7572
CNN2	1400	1361	490	499	0.7363	0.7352	0.7407	0.7372
CNN3	1365	1472	379	534	0.7565	0.7952	0.7826	0.7187
model	**t1**	**t2**	**npv**	**mcc**	**f1**	**if**	**mk**	**time**
logistic	0.3361	0.3752	0.7173	0.3506	0.6932	0.3499	0.3812	17
CNN1	0.2890	0.2427	0.7405	0.4688	0.7427	0.4861	0.4693	731
CNN2	0.2592	0.2627	0.7317	0.4724	0.7389	0.4724	0.4724	362
CNN3	0.2173	0.2812	0.7337	0.5152	0.7493	0.5119	0.5163	242

of training rows. However, even their lowest recorded values might not be considered acceptable for a real world application. We believe that this can be remedied by fine tuning the architecture and the training of the CNN.

In terms of accuracy, precision, type I and II error rates, and of the Matthews correlation every CNN outperforms the standard logistic classfier. Moreover, in two training cases the CNN achieves better recall. This can be attributed to the extended training process of the CNN as well as to the fact that its considerably more synaptic weights have more memory compared to the limited one present in the logistic regressors.

7 CONCLUSIONS AND FUTURE WORK

This conference paper presents the implementation in keras using TensorFlow as a backend for a convolutional neural network (CNN) under three different training scenaria in order to predict whether the next tweet will contain a mention to a verified Twitter account. The dataset is stored as a JSON collection in a MongoDB instance and comprises of 5000 tweets pertaining to the award of the Literature Nobel prize to Bob Dylan. In order to demonstrate the inherent power of CNN, a logistic regression was performed on the same dataset and nine statistical metrics were computed from the four contingency matrices. In terms of accuracy, precision, type I and type II error rates the CNN is superior.

The methodological framework of this work can be extended in a number of ways. Concerning immediate research, a number of combinations of hidden layers and activation functions can be constructed in TensorFlow in order to find the architecture which yields the optimum value for accuracy or for other metrics. Additionally, the deployment of TensorFlow or a similar tool such as theano or torch7 over a GPU or an array of GPUs would accelerate the training process.

Longer term research objectives include techniques for accelerating the training process by using additional constraints for pruning highly correlated synaptic weights, selecting a different objective function which offers increased interpretability within the given Twitter context, and augmenting the training dataset with affective information. The latter is the primary driving force behind the digital activity of netizens and, thus, has a plethora of applications to social media such as evaluating brand loyalty, predicting the outcome of political campaigns, and assessing the digital influence of accounts (Drakopoulos, 2016). As for the synaptic weight constraints, semantic metrics such as Wu-Palmer or Leacock-Chodorow in conjunction with information extracted from the features can be used to control the variance of synaptic weights in neighboring layers. Finally, when the mentions to verified accounts are rare, results from the extreme value theory such as the Fisher-Tippett-Gnedenko theorem can be used to model the probabilistic behavior of mentions.

ACKNOWLEDGEMENTS

The financial support by the European Union and the Greece (Partnership Agreement for the Development Framework 2014-2020) under the Regional Operational Programme Ionian Islands 2014-2020 for the project "TRaditional corfU Music PresErvation through digiTal innovation - TRUMPET" is gratefully acknowledged.

REFERENCES

Blackmore, S. (2000). *The meme machine*. Oxford Universtiy Press.

Drakopoulos, G. (2016). Tensor fusion of social structural and functional analytics over Neo4j. In *IISA*. IEEE.

Drakopoulos, G., Gourgaris, P., and Kanavos, A. (2018). Graph communities in Neo4j: Four algorithms at work. *Evolving Systems*.

Drakopoulos, G., Kanavos, A., Karydis, I., Sioutas, S., and Vrahatis, A. G. (2017a). Tensor-based semantically-aware topic clustering of biomedical documents. *Computation*, 5(3).

Drakopoulos, G., Kanavos, A., Mylonas, P., and Sioutas, S. (2017b). Defining and evaluating Twitter influence metrics: A higher order approach in Neo4j. *SNAM*, 71(1).

Drakopoulos, G., Kanavos, A., and Tsakalidis, A. (2016a). Evaluating Twitter influence ranking with system theory. In *WEBIST*.

Drakopoulos, G., Kanavos, A., and Tsakalidis, K. (2017c). Fuzzy random walkers with second order bounds: An asymmetric analysis. *Algorithms*, 10(2).

Drakopoulos, G., Kontopoulos, S., and Makris, C. (2016b). Eventually consistent cardinality estimation with applications in biodata mining. In *SAC*. ACM.

Drakopoulos, G., Stathopoulou, F., Kanavos, A., Paraskevas, M., Tzimas, G., Mylonas, P., and Iliadis, L. (2019). A genetic algorithm for spatiosocial tensor clustering: Exploiting TensorFlow potential. *Evolving Systems*.

Gilbert, E. and Karahalios, K. (2009). Predicting tie strength with social media. In *SIGCHI conference on human factors in computing systems*, pages 211–220. ACM.

Golbeck, J. and Hendler, J. (2006). Inferring binary trust relationships in Web-based social networks. *TOIT*, 6(4):497–529.

Golbeck, J., Hendler, J., et al. (2006). Filmtrust: Movie recommendations using trust in Web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, pages 282–286.

Golbeck, J. A. (2005). *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland, College Park.

Jamali, M. and Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM.

Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P., and Bhattacharjee, B. (2007). Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM.

Muller, M. (2004). Multiple paradigms in affective computing. *Interacting with Computers*, 16(4):759–768.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Papalexakis, E. E. and Faloutsos, C. (2015). Fast efficient and scalable core consistency diagnostic for the PARAFAC decomposition for big sparse tensors. In *ICASSP*, pages 5441–5445.

Papalexakis, E. E., Pelechrinis, K., and Faloutsos, C. (2014). Spotting misbehaviors in location-based social networks using tensors. In *WWW*, pages 551–552.

Picard, R. W. (2003). Affective computing: Challenges. *International Journal of Human-Computer Studies*, 59(1):55–64.

Picard, R. W., Vyzas, E., and Healey, J. (2001). Toward machine emotional intelligence: Analysis of affective physiological state. *TPAMI*, 23(10):1175–1191.

Russell, M. A. (2013). *Mining the social Web: Analyzing data from Facebook, Twitter, LinkedIn, and other social media sites*. O'Reilly, 2nd edition.

Thompson, K. (1984). Reflections on trusting trust. *Communications of the ACM*, 27(8):761–763.

Tversky, A. (1977). Features of similarity. *Psychological review*, 84(4):327.

www.counterpunch.org (2017). Go ask Alice: The curious case of Alice Donovan.

www.theguadian.com (2017). Twitter drops egg avatar in attempt to break association with Internet trolls.