A Deep Learning Architecture for Audience Interest Prediction of News Topic on Social Media

Ciprian-Octavian Truică*†
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
ciprian.truica@upb.ro

Elena-Simona Apostol*†
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
elena.apostol@upb.ro

Teodor Ștefu†
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
teodor.stefu@cti.pub.ro

Panagiotis Karras
Department Computer Science
Aarhus University
Aarhus, Denmark
panos@cs.au.dk

ABSTRACT

Personalized social media offer communication opportunities that mass media could not afford, yet also raise novel challenges. A prime challenge arising from this shift in digital communication is to detect topics and events of interest. In this paper, we propose and deploy a novel Deep Learning architecture that predicts if a *news topic* becomes viral by analyzing social media diffusion and audience interest in current *news events*. The proposed solution: (i) analyzes news articles, (ii) extracts associated topics and events, (iii) matches the topics and events to filter and extract developing topics, (iv) extracts current events from Twitter and matches them to the filtered news topics, and (v) predicts audience interest in news topics using Twitter likes and retweets. We employ several feature engineering techniques to improve prediction by integrating user metadata into the training set. In our experiments, we correlate two datasets collected over several months in the same time period. The first dataset contains news articles collected from different news venues, while the second one contains tweets regarding the news. The experimental results from our real-world deployment prove that the proposed system achieves high accuracy when integrating *influencers* metadata and the day of the week. Thus, proving that the *news topics* virality prediction is improved under the assumptions that spreaders and the day of the week play a huge role in information diffusion.

KEYWORDS

audience interest prediction, social media, news diffusion, topic modeling, event detection, neural networks

1 INTRODUCTION

The internet age has brought new ways of sharing information that changed the way the general public consumes media content. Physical newspapers moved to a digital form in a few years. With this migration to the virtual world, the number of news sources increased, and consumers gained a wide variety of options from where to choose their daily news using their preferred content. Users have two methods at their disposal to stay up to date with current *news events*:

(1) buy a subscription to news publisher that provides in-app or email news content, or
(2) follow the content of news outlets or *influencers* on social media [30], where *influencers* are users that manage to sway a target audience.

An emerging and growing social media platform is Twitter. Twitter provides users a venue where they can share their thoughts, discuss, and forward various kinds of information [29]. The subjects are diverse, from daily local events to important global issues. The ever-growing number of users around the world tweeting makes Twitter a valuable source for real-time information. Thus, many consumers spend time on their news feeds to catch up with everything around them.

News outlets and publishers also embraced Twitter to update followers by regularly posting tweets that contain short news headlines and including direct links to full articles. Thus, news outlets reach large audiences and increase their reader base. Yet, to predict the audience's interest in a news topic, we should determine the underlying social structures and relationships between Twitter users [14], and utilize the network graph structure modelling the relationships between members of different social groups [18]. Nodes in a group's center are called *influencers* as they have a huge role in spreading the information. Nodes that like or retweet content are known as *spreaders*, as they propagate the information further in the network.

Our proposed solution is motivated by the recent negative impact viral Fake News has on society at large and the current need for systems that can help in stopping the viral spreading of such news. Knowing the veracity, determining the propagation patterns, and predicting the influence of news articles in social media, new strategies for mitigating harmful misinformation can be developed. Thus, the main objective of this paper is to determine if a news topic becomes viral on social media. We tackle this issue by predicting the audience in order to facilitate the development of new network immunization strategies.

The research questions we are trying to answer are:

(Q1) Do the current events presented in mass media also gain traction on social media?
(Q2) Does the diffusion of news articles on social media influence the trending topic of current events?

*Corresponding Author
†These authors contributed equally to the paper

(Q3) Can we predict the audience interest in a news topic by analysing the retweets and likes received by news articles belonging to the topic on social media?

(Q4) Does the network-related metadata improve the audience interest prediction?

To answer *(Q1)* and *(Q2)*, we first extract the *news topics* using Non-Negative Matrix Factorization (NMF) [2] and the *news events* using Mention-Anomaly-Based Event Detection (MABED) [15]. Second, we correlate the *news topics* and *news events* to extract *trending news topics* by employing the cosine similarity. Finally, we extract *Twitter events* also using MABED. For answering *(Q1)*, we analyze the correlation results of *trending news topics* → *Twitter events*. Whereas, for answering *(Q2)*, we analyse the reverse correlation results, i.e., *Twitter events* → *trending news topics*.

To answer *(Q3)*, we consider that the correlated < *trending news topics*, *Twitter events* > pair reveals insights in the audience interest through likes and retweets. Thus, we train two Deep Learning models on the *Twitter events* to predict their likes and retweets on Twitter. The high accuracy of the models proves that we can correctly predict the audience interest in *trending news topics* using social media information and determine if they become viral.

Finally, to answer *(Q4)*, we enhance the training dataset with metadata containing information regarding each tweet's author and its number of followers as well as the day of the week when the tweet was posted online. We base this approach on two assumptions. First, that *influencers* (users with a high number of followers) have a huge role in spreading the information and making news articles topics viral. Second, that user behaviour posting patterns, as well as media consumption, are influenced by the day of the week, as also empirically proven in [3]. We retrain our Deep Learning models and manage to obtain better accuracy results, thus showing that the assumptions stand for our use case.

We use Twitter's graph structure to extract events and to predict the audiences' interest by analyzing the textual and metadata content (e.g., retweets, likes, user followers, etc.) of each post. Thus, we propose and deploy a novel architecture that incorporates topic modeling, event detection, and Deep Learning classification. Our system:

(1) analyzes news articles,
(2) extracts their topics and associated events,
(3) matches these topics and events to filter and extract developing topics,
(4) extracts current events from Twitter and matches them with the filtered news topics, and
(5) predicts audience interest in the news topics using Twitter likes (formally known as favorites) and retweets by enhancing the dataset with metadata, i.e., the tweet author and its number of followers, as well as the day when the tweet was posted.

The novelty of our proposed architectures is threefold:

(1) we correlate and discover new insights between the relation of *trending news topics* and *Twitter events* in both directions, i.e., *trending news topics* → *Twitter events* and *Twitter events* → *trending news topics*;
(2) we manage to create accurate models that predict the audience interest in *trending news topics* on social media and determine if they become viral;
(3) we improve the model's performance by incorporating in our document embeddings and learning models the assumption that *influencers* play an important role in the spreading as well as the day when the tweet was posted, thus making *trending news topics* to become viral.

Experimental results show that all *trending news topics* are correlated to at least one *Twitter event*, whereas the reverse is not true. Furthermore, we obtain better audience interest prediction by enhancing the dataset with metadata.

The rest of this paper is structured as follows. Section 2 presents a survey of state-of-the-art-methods for analyzing news diffusion on social media. In Section 3, we discuss the models and techniques used in our deployed solution. Section 4 presents the architecture of our system and discusses each module in detail. Section 5 showcases and discusses the obtained results. Lastly, in Section 6 we conclude and we present several new directions and improvements for the proposed solution.

2 RELATED WORK

Online media content shapes people's perceptions regarding ongoing social, political, and economical changes around them. In the literature, the relevance of the story selection correlated to a specific audience, done by editors or by algorithms, has been analyzed to find better ways to get news online and explore the relationships that exist between individuals' characteristics and their interests. The results show that the audience's interest can be determined by context-specific characteristics [34]. Furthermore, the prediction of the audience's interest in a *news topic* can help publishers to create recommendation systems for social platforms that leverage tailor-made latent features [5].

One solution for building better recommendation systems for targeting the interested audience proposes to analyse the content of posts to detect bursty topics. The analysis predicts the emergence of current events that are of interest to multiple groups of people who discuss and share the content online [1, 15, 31]. Furthermore, community-based probabilistic algorithms can be used to model the spreading of *news events* on social networks [25]. Therefore in this paper, we use event detection techniques to match *news topics* with *news events* and extract *trending news topics*. Then, we correlate the *trending news topics* with *Twitter events* using the same time frame to determine the spread of current news topics on the social media platform and determine *viral topics*.

The visibility of news on social media also depends on the actions of a diverse set of actors, e.g., the users, their friends, content publishers such as news organizations, advertisers, and algorithms [33]. Thus, we propose a new feature construction method that incorporates metadata into the features of the training set.

Deep Learning architectures have been successfully used for predicting the trend of stocks [19], financial time series [28], marketing [26], etc. Deep Learning architectures have also been used to predict information diffusion in social media. Recurrent Neural Networks (RNN) architecture is a popular deep learning-based model used to model information diffusion, obtaining promising performance. In [36], the authors explore the advantages of using an RNN-based model enhanced with reinforcement learning in order to predict information diffusion in social media. In their work, they tackle diffusion prediction both at the user level (microscopic), i.e., the next influenced user, and at the network level (macroscopic). The

proposed solution, FOREST (reinFOrced REcurrent networks with STructural context), consists of two models: the microscopic and macroscopic cascade models. The microscopic model employs a Gated Recurrent Unit architecture and a structural context extraction algorithm based on neighborhood sampling. This model is fed to the macroscopic model that also applies a reinforcement learning based cascade simulation process. For experiments, they used three datasets, each from different social media platforms. FOREST is compared with other state of the art solutions that employ Long Short-Terms Memory, attention layers, and Convolutional or Recurrent Neural Nentwork architectures. The authors use the Mean-Square Log-Transformed Error (MSLE) metric for comparison. FOREST outperforms all the baseline solutions. In comparison with our work, FOREST does not detect news topics and does not analyze from this point of view the spread of information. Also, their models ignore the timestamp information. A similar solution to [36] that, in contrast, considers the temporal information, but which still does not address audience prediction on particular topics, is proposed in [6]. Their proposed method, called CasCN (Recurrent Cascades Convolutional Networks), is also based on an RNN architecture. CasCN predicts cascades through learning the latent representation of both structural and temporal information.

To the best of our knowledge, there are no current solutions that connect news articles and social media events in order to predict the spread of certain news topics in social media. To address this shortcoming, we analyze both news articles and tweets in order to correlate the events from the two type of sources and employ Deep Learning architectures together with our feature engineered training set for an accurate audience interest prediction on particular news topics.

3 METHODOLOGY

In this section, we discuss the core components of our solution.

3.1 Term Weighting Schemes

In Information Retrieval and Text Mining, a weighting scheme is a statistical measure to evaluate how important a term is to a document in a collection or corpus. Weighting schemes are the basis for many document vectorization techniques, such as the vector space model where each feature is a word (term), and the feature's value is a term weight.

For a corpus of documents $D = \{d_1, d_2, ..., d_n\}$, where $n = ||D||$ is the total number of documents in the dataset, a document d_i is defined as a sequence of terms t_{ij}, $d_i = \{t_{i1}, t_{i2}, ..., t_{im}\}$ where m is the length of the vocabulary. The vocabulary is the set of distinct words that appear in the corpus of documents. The term frequency $TF(t_{ij}, d_i)$ of a term t_{ij} is equal to the number of co-occurrences of that term in a document d_i (Equation (1)).

$$TF(t_{ij}, d_i) = f_{t_{ij}, d_i} \tag{1}$$

The inverse-document frequency $IDF(t_{ij}, D)$ is a statistical measure of the importance of a term in a text document collection (Equation (2)), where t_{ij} is the term, D is the corpus of documents, $n = ||D||$ is the number of documents in the corpus, and n_{ij} is the number of documents where term t_{ij} appears. Terms with a low document frequency add more information than terms with a high frequency. Thus, the more frequently the term appears in the collection, the less informative the term is.

$$IDF(t_{ij}, D) = \log_2 \frac{n}{n_{ij}} \tag{2}$$

Term frequency-inverse document frequency ($TFIDF(t_{ij}, d_i, D)$) is a statistical measure used to determine the importance of a word regarding its frequency in a document relative to the entire corpus. The term importance is proportional to the number of times a word appears in the document, although it is counterbalanced by the frequency of that word in the corpus (Equation (3)).

$$TFIDF(t_{ij}, d_i, D) = TF(t_{ij}, d_i) \cdot IDF(t_{ij}, D) \tag{3}$$

The normalized term frequency-inverse document frequency $TFIDF_N(t_{ij}, d_i, D)$ (Equation (4)) uses the ℓ^2-norm (Equation (5)) to normalize the values of $TFIDF(t_{ij}, d_i, D)$, for each term t_{ij} in each document d_i, in the $[0, 1]$ interval.

$$TFIDF_N(t_{ij}, d_i, D) = \frac{TFIDF(t_{ij}, d_i, D)}{\ell^2(d_i)} \tag{4}$$

$$\ell^2(d_i) = \sqrt{\sum_{t_{ij} \in d_i} \left(TFIDF(t_{ij}, d_i, D)\right)^2} \tag{5}$$

Using the weights, we construct document-term matrices $A \in \mathbb{R}^{n \times m}$ to describe the frequency of terms that occur in the dataset. By considering this representation, rows correspond to documents and terms to columns.

3.2 Topic Modeling

Topic modeling is a statistical unsupervised machine learning method used to extract hidden latent semantic patterns within a corpus of documents. Topic modeling algorithms use either statisical models, i.e., Probabilistic Latent Semantic Indexing (PLSI) [17], generative statistical models, i.e., Latent Dirichlet allocation (LDA) [4], or matrix factorization, e.g., Non-Negative Matrix Factorization (NMF) [2], Latent Semantic Analysis [9] (LSA). Experimental results prove that NMF is the best choice for extracting topics [7].

Non-Negative Matrix Factorization. NMF is an algorithm that factorizes a matrix $A \in \mathbb{R}^{n \times m}$ into two non-negative matrices $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times m}$. In the case of topic modeling, the matrices have the following signification:

(1) A is a document-term matrix constructed using weighted term frequencies for a corpus containing n documents and a vocabulary of size m terms;
(2) W is the document-topic matrix that assigns a document membership to each topic k;
(3) H is the topic-term matrix that assigns to each topic k the importance of a term.

To determine W and H, the objective function $F(W, H)$ must be minimized by respecting the constraint that all the elements of W and H are non-negative. Equation (6) presents the objective function, where $|| \cdot ||_F$ is the Frobenius norm.

$$F(W, H) = ||A - WH||_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - (WH)_{ij})^2 \tag{6}$$

To minimize the objective function (Equation (7)), the values of W and H are updated iteratively (with t the index of the iteration) until they stabilize (Equation (8)).

$$\min_{W \geq 0, H \geq 0} F(W, H) = \min_{W \geq 0, H \geq 0} ||A - WH||_F^2 \tag{7}$$

$$H_{ij}^{t+1} \leftarrow H_{ij}^{t} \frac{((W^t)^T A)_{ij}}{((W^t)^T W^t H^t)_{ij}}$$
$$W_{ij}^{t+1} \leftarrow W_{ij}^{t} \frac{(A(H^{t+1})^T)_{ij}}{(W^t H^{t+1}(H^{t+1})^T)_{ij}} \quad (8)$$

3.3 Information Diffusion

Information diffusion in social media studies the data propagation to find events and forecast their spreading [13]. Event detection is a subdomain of information diffusion that aims to discover real-world events from the social media [38]. We choose Mention-Anomaly-Based Event Detection (MABED) [15] as the *Twitter event* detection algorithm. MABED is a statistical event detection on social media method immune to the topic bias added by the texts unrelated to the event.

Mention-Anomaly-Based Event Detection. MABED is an efficient method for event detection that filters irrelevant content and successfully removes spam messages with no actual intent, posted around certain hours.

To identify a bursty topic and detect an event for a period of time $I = [a; b]$ and a main word t (event label) with MABED, a weight $w_{t'_q}$ is computed for each candidate word t'_q (event keywords) in the time slice i (Equation (9)). The weight is computed using the affine function $\rho_{O_{t,t'_q}}$ (Equation (10)) that corresponds to the first order auto-correlation of the time-series for N_t^i (number of tweets in the time-slice i that contain the main word t) and $N_{t'_q}^i$ (number of tweets in the time-slice i that contain the candidate word t'_q) [12].

$$w_{t'_q} = \frac{\rho_{O_{t,t'_q}} + 1}{2} \quad (9)$$

$$\rho_{O_{t,t'_q}} = \frac{\sum_{i=a+1}^{b} A_{t,t'_q}}{(b-a-1)A_t A_{t'_q}} \quad (10)$$

Where:

(1) $A_{t,t'_q} = (N_t^i - N_t^{i-1})(N_{t'_q}^i - N_{t'_q}^{i-1})$;
(2) $A_t^2 = \frac{\sum_{i=a+1}^{b}(N_t^i - N_t^{i-1})^2}{(b-a-1)}$;
(3) $A_{t'_q}^2 = \frac{\sum_{i=a+1}^{b}(N_{t'_q}^i - N_{t'_q}^{i-1})^2}{(b-a-1)}$.

3.4 Text Representation using Embeddings

Before using machine learning models for classification, prediction, or clustering, the documents must be transformed from textual data to numerical data.

Word Embedding. The Word to Vector model (Word2Vec) is a shallow neural architecture that produces a vector space for a textual dataset using the values of the neural network hidden layer [27]. There are two approaches proposed in the literature:

(1) Continuous Bag-Of-Words model (CBOW) which accounts for the textual dataset vocabulary and represents documents as a set of continuous word-multiplicity pairs. The input to the hidden layer connections is replicated by the number of context words, and the context is preserved through the use of multiple words that target a given word.
(2) Skip-gram model which represents documents as sequences of words with gaps between them. The input to the neural network is the target word, and the output layer is replicated multiple times to accommodate the chosen number of context words. Thus, this model manages to embed in the word representation its linguistic context.

The two models mirror each other while both preserve context. The CBOW model uses multiple neighbouring words to preserve the context for a target word, while the Skip-gram model uses a word to preserve the context for multiple targeted neighbouring words.

Document Embedding.

The Document to Vector (Doc2Vec) model learns continuous distributed vector representations for textual data [23]. The text dimension may vary from phrases and sentences to large documents. The model is similar to the Word2Vec model which maps words into the vector space maintaining the semantic similarities by using a given word's context. There are two approaches in the literature [23]:

(1) The Paragraph Vectors Distributed Memory (PVDM) model [23] extends the CBOW architecture by mapping each document to a vector via an additional document-to-vector matrix and concatenating this vector to the word vectors in order to predict the central word.
(2) The Paragraph Vectors Distributed Bag of Words (PVDBOW) predicts the central word using the same mechanism as PVDM model but does not preserve the word order and ignores the context.

Similarity. Using word or document embeddings together with the cosine similarity [24], the semantic similarity between two words or documents can be computed [20]. This method assumes that two embeddings have a non-zero norm and measures their orientation instead of their magnitude, as in the case of the Euclidean distance. Equation (11) presents the cosine similarity for two p-dimensional vectors x and y.

$$cos(\theta) = \frac{\sum_{i=1}^{p} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{p} x_i^2} \cdot \sqrt{\sum_{i=1}^{p} y_i^2}} \quad (11)$$

3.5 Deep Learning Architectures

Artificial neural networks (ANNs) use processing units to predict an output $y \in \mathbb{R}^{n \times k}$ for an input dataset $X \in \mathbb{R}^{n \times m}$. For the given input containing feature vectors $x_i \in X$, the processing unit will try to predict an output $\hat{y}_i = \delta(\sum_{j=1}^{m}(w_{ij} \cdot x_{ij}) + b)$ using a weight vector w_{ij}, the bias b, and the activation function $\delta(\cdot)$. The activation function has different forms depending on the processing unit (Table 1).

Table 1: Activation functions

Name	Function
Sigmoid	$\delta(z) = \sigma_g(z) = \frac{1}{1+e^{-z}}$
Hyperbolic	$\delta(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
ReLU	$\delta(z) = \max(0, z)$
Softmax	$\delta(z) = \frac{e^z}{\sum_{i=1}^{m} e^{z_i}}$ $(m = dim(z))$

To accurately predict the output y, the ANN models minimize the loss or cross entropy function (Equation (12)) by adjusting the weights after a specified finite number of iterations or when the function stabilizes.

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y})) \quad (12)$$

Stochastic gradient descent (SGD) (Equation (13)) is used to update the weight vector at iteration t using the weights computed during the previous iteration. Equation (14) presents the weigths update function, where η is the global learning rate and $\alpha \in [0; 1]$ is the exponential decay factor.

$$\gamma_t = \nabla_{\boldsymbol{w}^{(t)}} L(\hat{y}, y) \tag{13}$$

$$\triangle \boldsymbol{w}^{(t)} = \alpha \triangle \boldsymbol{w}^{(t-1)} - \eta\gamma_t \tag{14}$$

ADAGRAD is a method used to improve SGD by increasing the learning rate when the $\boldsymbol{w}$ weight vectors are sparse [11]. Equation (15) presents update rule used by ADAGRAD, where $||\gamma||_2$ is the ℓ^2-norm of all previous gradients on a per-dimension basis.

$$\triangle \boldsymbol{w}^{(t)} = -\frac{\eta}{||\gamma||_2}\gamma_t \tag{15}$$

ADAGRAD has two problems :

(1) the continual decay of learning rates throughout training, and
(2) the need for a manually selected global learning rate.

ADADELTA [39] solves these two problems by using the Root Mean Square (*RMS*) to update the weights (Equation (16)).

$$\triangle \boldsymbol{w}^{(t)} = -\frac{RMS[\triangle \boldsymbol{w}]_{t-1}}{RMS[\gamma]_t}\gamma_t \tag{16}$$

Two Deep Learning Architectures used successfully in classification [21] are Multi-Layer Perceptron and Convolutional Neural Network.

Multi-Layer Perceptron. One of the basic processing units is the perceptron, which maps its input x_i to a single binary value $\hat{y}_i \in \{0, 1\}$. The perceptron can generalizes naturally to a multi-class perceptron to predict $\hat{y} = argmax_y f(x, y) \cdot \boldsymbol{w}$ by using a feature representation function $f(x, y)$ that maps each possible input/output pair to a finite-dimensional real-valued feature vector and multiplies it by a weight vector $\boldsymbol{w}$.

The Multi-Layer Perceptron (MLP) is a feed-forward ANN architecture that stacks perceptron units into three fully connected weighted directed layers:

(1) the input layer,
(2) the hidden layer, and
(3) the output layer.

The MLP becomes a Deep Feed-Forward Neural Network architecture by adding multiple hidden layers

Convolutional Neural Network. A Convolutional Neural Network (CNN) is an ANN architecture similar to the MLP, except it contains multiple hidden layers. These hidden layers consist of a series of convolutional layers that apply a filter to the activation function. They include the following types of layers: pooling, fully connected, and normalization layers. The pooling layer is used to reduce the dimensions of the data by combining the outputs of one layer into a single neuron in the next layer.

3.6 Evaluation Methods

Multi-class classification models assign a data point to one and only one non-overlapping class c_i ($i = \overline{1, k}$) [32]. To evaluate the quality of the model we can use the average accuracy (Equation (17)) where:

- TP_i (True Positive) is the number of data points correctly classified with class C_i;
- FN_i (False Negative) is the number of data points incorrectly classified with a class C_j ($j \neq i$);
- FP_i (False Positive) is the number of data points incorrectly classified with class C_i;
- TN_i (True Negative) is the number of data points correctly classified with a class C_j ($j \neq i$).

$$A = \frac{1}{k}\sum_{i=1}^{k}\frac{TP_i + TN_i}{TP_i + FN_i + FP_i + TN_i} \tag{17}$$

4 PROPOSED SOLUTION

Figure 1 presents the architecture of the proposed solution. The architecture is modular and each module is described in the following paragraphs. The code is publicly available on GitHub[1].

4.1 Data Collection and Storage Modules

To collect the News corpus we have used two APIs (Application Programming Interface): News River API[2] and NewAPI[3]. News River API returns the latest 100 news based on a given subject, e.g., politics, brexit, etc. NewAPI is a public and free API for news that contains news from many sources and can be configured to request the latest 100 news from the most popular news publisher, e.g., The New York Time, Reuters, The Washington Times, etc. Although, NewsAPI provides multiple metadata about each news article, e.g., title, description, online location, etc, the content is reduced to the first paragraph. We developed a scrapper to obtain the entire content of the article. To collect tweets, we used the Twitter API to request tweets with specific keywords and usernames. Besides the content of the tweet, we also store likes, retweets, and creation time for each tweet. The datasets are stored in a MongoDB[4] database.

4.2 Preprocessing Modules

Text preprocessing is done differently depending on the task, i.e., Topic Modeling or event detection, and type of corpus, i.e., News or Tweets. Thus, we create three preprocessed corpora:

(1) NewsTM with news articles for topic modeling,
(2) NewsED with news articles for event detection, and
(3) TwitterED with tweets for event detection.

To create the NewsTM corpus, we employ the following perprocessing steps:

(1) extract named entities to treat them as concepts and not as simple terms,
(2) extract lemmas to minimize the vocabulary and store only the base root, and
(3) remove punctuation and remove stop words because they do not add any information gain.

Both NewsED and TwitterED corpora are created using two preprocessing steps:

(1) removal of punctuation, and
(2) tokenization.

We choose this simple preprocessing pipeline to replicate the text preprocessing done originally for the MABED algorithm. The preprocessing pipeline is implemented in SpaCy[5]. The results are stored in the MongoDB database.

[1] https://github.com/cipriantruica/news_diffusion
[2] https://newsriver.io/
[3] https://newsapi.org/
[4] https://www.mongodb.com/
[5] https://spacy.io/

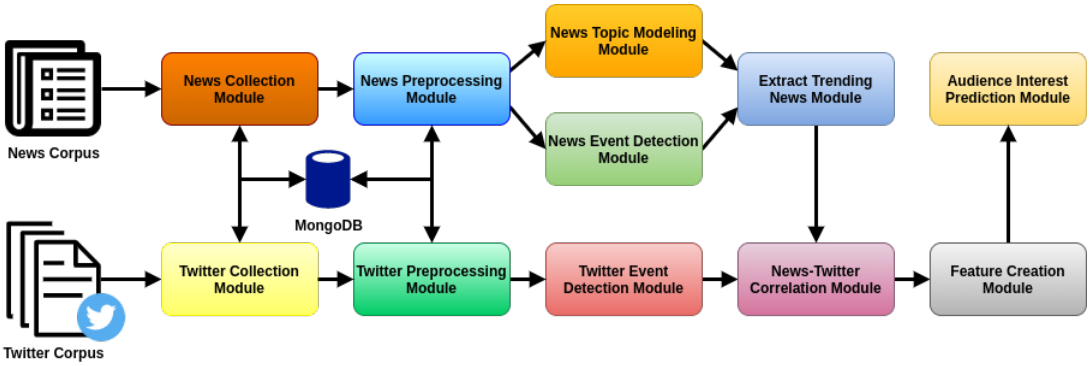

Figure 1: Architecture

4.3 Topic Modeling Module

Topic modeling is used to create an overview of the current and of interest subjects in the media. We use the NMF algorithm to extract the main topics from the NewsTM corpus. A document-term matrix is constructed from the NewsTM corpus after each document is vectorized using the $TFIDF_N$ weight. We use the Scikit-learn[6] implementations for $TFIDF_N$ and NMF.

4.4 Event Detection Modules

For both news articles and tweets, we use the MABED algorithm to detect trending topics. We enhance the NewsED and TwitterED datasets with the creation time of each record. MABED detects events defined by three characteristics:

(1) a set of main words,
(2) a set of related words, and
(3) the period of time when the topic is of interest.

We use the original implementaion of MABED[7].

4.5 Trending News Module

To correlate the news topics to the *news events*, we use the Doc2Vec model to encode the topic keywords (NewsTopic2Vec) and the *news events*' main and related terms (NewsEvent2Vec). Using the cosine similarity (Equation (11)) implemented in SpaCy, we score the match between each topic and each event. We extract the best matches for each topic, and with the help of each matching event, we determine how topics are diffused in mass media.

4.6 Correlation Module

The *news events* are correlated to the *Twitter events* to determine how current information is propagated on social media. As in the case of the *news event*, we use Doc2Vec to encode each *Twitter events*' main and related terms into one vector (TwitterEvent2Vec). We correlate the news and *Twitter events* that appear in the same time interval and extract candidate correlations. The cosine similarity implemented in SpaCy is used to find the best matches from the candidate correlations. Using this method we determine how news is propagated in social media.

[6]https://scikit-learn.org/stable/
[7]https://github.com/AdrienGuille/pyMABED

4.7 Feature Creation Module

Using the *Twitter events* detected by the Correlation Module, we extract the tweets that belong to each event. We consider that a tweet is part of an event if both of the following conditions are true:

(1) it was posted during the event's period of time, and
(2) its textual content contains at least one main word and 20% of the related words.

An event is considered of interest if there are at least 10 records associated to it. We constructed a new sub-dataset using this criteria. We encode each tweet belonging to an event using Word2Vec on the tweet's terms present in the vocabulary containing the main and related terms of that event. We use the Gensim[8] Word2Vec implementation. We then create three custom Doc2Vec embeddings for each tweet by averaging the Word2Vecs as follows:

(1) SW_Doc2Vec: only Word2Vecs found in the pre-trained model are considered in computing the document embedding;
(2) RND_Doc2Vec: random vectors with values in the range $[-1, 1]$ for terms that are not found in the pre-trained model are added before computing the document embedding;
(3) SWM_Doc2Vec: Word2Vecs found in the pre-trained model are multiplied by the word's magnitude in the context of the event before computing the document embedding.

Before creating the datasets for predictions, we also incorporate metadata into the representation of each record. The metadata vector incorporates a one-hot-encoder vector that embeds the author of each tweet, its number of followers, and the day of the week. We added these features because it was proven empirically that *i)* the number of followers influences the propagation of news in social media [16], and *ii)* the media consumption is influenced by the day of the week [3].

We also create a feature that represents the number of followers, during the period of the event, for each user. The likes and retweets classes used for predicting the events interest on social media are constructed similarly to the feature that encodes the user's number of followers. Table 2 presents the encoding for the number (#) of user followers, tweet likes, and retweets, respectively.

[8]https://radimrehurek.com/gensim/

Table 2: Features Encoding

Feature	# < 100	# ∈ [100, 1 000]	# > 1 000
followers	0	1	2
likes	0	1	2
retweets	0	1	2

4.8 Audience Interest Prediction Module

To predict the audience interest in a news topic, we analyze the retweets and likes received by the tweets that discuss the news articles belonging to that topic. We use two Deep Neural Networks architectures for our prediction module:

(1) a Multi Layer Perceptron (Figure 2), and
(2) a Convolutional Deep Learning architecture (Figure 3).

The first architecture uses only perceptron units that can easily generalize a wide variety of problems. The second architecture uses a convolution layer and a max pooling layer. We use accuracy to evaluate the networks' performance and classification's quality.

4.9 Design choices

We decided to fetch the latest tweets and news every 2 hours. Thus, we leave a large enough time window to manage to collect a representative number of new tweets and news articles. The algorithms are executed, from checkpoints or from scratch, after each dataset update, and the models are replaced with the new ones as soon as they finish. Thus, multiple instances of the same algorithm may run at the same time on different datasets. Also, we choose to use NMF instead of LDA [4] as it provides similar results on both small and large length texts in less time [35].

For training the word embeddings and then vectorizing the textual data to extract Doc2Vec, we choose a pretrained word2vec on the Google News corpus[9]. This dataset contains 3 million 300-dimension English word vectors. We made this design choice because this dataset is larger than the datasets we collected and manages to provide better word representations on which to construct our Doc2Vec embeddings. In contrast, the PVDM and PVDBOW models are not good for our study because they will not find good document representations since they can be trained by us only on the collected datasets. Thus, these models do not manage to generalize the document representation.

To alleviate the need to train the neural models each time the datasets are updated, we use checkpoints to continue the training as new data is added in real time. Furthermore, the most demanding networks are trained in less than 7 minutes. Thus, the models can be build again if all data is new.

5 EXPERIMENTAL RESULTS

We apply our method on a real-world data, aiming to eventually predict user interest.

5.1 Datasets

We have collected 261 052 news articles, and 80 569 tweets for a period of 5 months. Both datasets contain records from different news venues and are all written in English. For the Twitter corpus, we also collected statistics for both tweets, i.e., likes and retweets, and users, i.e., number of followers, the friends count, and the retweets count.

[9]https://code.google.com/archive/p/word2vec/

5.2 News Topics

The first set of experiments uses the NewsTM corpus together with NMF algorithm to extract the most relevant 100 topics from all the news articles. This process takes 19.01 minutes. Table 3 presents a subset of 10 *news topics* (NT) extracted for the entire period. These news topics are used to showcase the correlation with the corresponding *Twitter events*.

Table 3: News topics

#NT	Keywords
1	party election vote seat poll voter conservative win european brexit
2	tariff import billion chinese good impose 25 consumer product percent
3	company business market industry customer service growth product year technology
4	trade deal market war global economy talk agreement tension china
5	huawei company google ban smartphone android chinese network security technology
6	iran iranian tehran sanction nuclear drone tension deal gulf tanker
7	israel gaza israeli palestinian hamas rocket militant palestinians jerusalem netanyahu
8	japan abe japanese emperor tokyo naruhito shinzo visit imperial meet
9	impeachment pelosi democrats impeach nancy inquiry speaker house proceeding congress
10	derby horse kentucky race win belmont maximum winner security racing

5.3 News Events

For our experiments, we extract the top 1 000 *news events* from the NewsED corpus, using the MABED algorithm. We use a time frame of 60 minutes. The extraction of *news events* takes 17.08 hours: 177.41 seconds to load the corpus, 1.3 hours to partition the news into time-slices, and 15.73 hours to extract events. Table 4 presents a subset of the *news events* (NE) detected by MABED. These *news events* are some of the *trending news articles* determined during the correlation with the *news topics*.

5.4 Twitter Events

Using MABED, we identified the top 5 000 events on the TwitterED corpus collected for a period of 5 months that have at least 10 tweets associated to it. We use a time window of 30 minutes. The extraction of *Twitter events* takes 11.74 hours: 1.61 seconds to load the corpus, 70.42 seconds to partition the news into time-slices, and 11.72 hours to extract events. Table 5 presents some of the detected *Twitter events* (TE) that we use to exemplify the correlation between *trending news topics* and *Twitter events*.

5.5 Correlation Results

We use cosine similarity to correlate the *news topics* (NT) with *news event* (NE) to extract *trending news topics*. Then, we correlate *trending news topics* with *Twitter events* (TE), i.e., *trending news topics* $\rightarrow$ *Twitter events* . First, we correlate each *news topics* with each *news event* and extract the ones with the highest similarity.

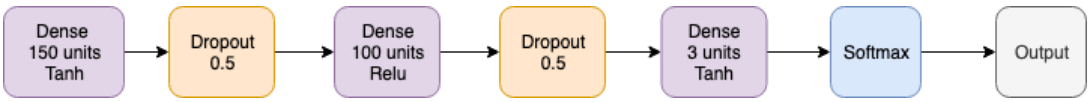

Figure 2: The MLP Network Architecture

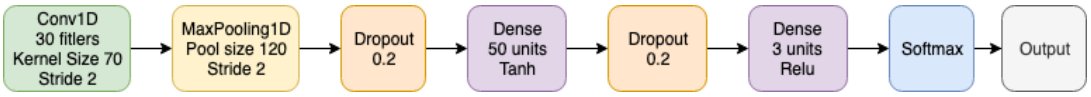

Figure 3: The CNN Network Architecture

Table 4: News events

#NE	Start Date	Start Date	Label	Keywords
1	2019-05-11 03:05:40	2019-05-26 13:05:40	politics	political european eu current election vote campaign voters
2	2019-05-11 03:05:40	2019-05-26 13:05:40	reached	current developing socialnews future 2019 group march company
3	2019-05-11 05:05:40	2019-05-26 13:05:40	plans	current prime group future business company vote european
4	2019-05-11 05:05:40	2019-05-26 13:05:40	comes	current future business part week north company american political
5	2019-05-11 05:05:40	2019-05-26 13:05:40	mobile	huawei current developing socialnews future chinese gopi adusumilli
6	2019-05-05 23:05:40	2019-05-26 13:05:40	threats	iran nuclear washington waters foreign american
7	2019-05-03 19:05:40	2019-05-11 11:05:40	conflict	military gaza israeli killed group hamas islamic political
8	2019-05-21 09:05:40	2019-05-26 13:05:40	japanese	japan abe tokyo north prime huawei tariffs american visit donald
9	2019-05-09 17:05:40	2019-05-22 01:05:40	familiar	white democrats committee administration congress
10	2019-05-02 19:05:40	2019-05-07 19:05:40	bob	derby security win mueller kentucky times

Table 5: Twitter events

#TE	Start Date	End Date	Label	Keywords
1	2019-04-29 20:01:30	2019-07-10 02:01:30	conservative	party theresa brexit leader mps prime minister leadership
2	2019-05-06 02:01:30	2019-06-27 14:01:30	fresh goods	tariffs threaten china trade good escalation import stock
3	2019-05-01 02:01:30	2019-08-09 02:01:30	improving	retail conservative taking actions people life growth online
4	2019-05-08 14:01:30	2019-07-31 08:01:30	war brand	big tax trade us-china markets billion conservative companies
5	2019-05-14 20:01:30	2019-07-12 14:01:30	networks	trump declare national emergency protect computer foreign web
6	2019-04-17 08:01:30	2019-07-25 02:01:30	muslim	aide biden hill bombshell betting handle joe contempt capitol
7	2019-04-19 20:01:30	2019-07-26 08:01:30	impeachment	democrats trump mueller pelosi testimony politically voted
8	2019-04-06 02:01:30	2019-05-22 08:01:30	woods tale	tiger victory roll lawsuit nationals back-to-back grand horse
9	2019-05-09 20:01:30	2019-05-21 02:01:30	senate alabam	passed effectively abortion ban bill committee
10	2019-05-07 08:01:30	2019-08-12 20:01:30	english fans	football manchester club everton fantasy clubs playing

The pairs < *news topics*, *news event* > are the *trending news topics*. We extracted 83 *trending news topics* that have a similarity of over 0.7. Then, for each *trending news topics*, we extract the *Twitter events* in the same time window with the highest Doc2Vec similarity. Given the start date (S_{TT}) of a *trending news topics* (TT), the constraint of the *Twitter events* (TE) start date is $S_{TE} \in [S_{NE}; S_{NE} + 5day]$. We choose this start interval because a *Twitter event* can appear on social media as soon as the news appears in the mass media, but it can also be some delay between the appearance time on the two platforms. The end date is not significant as a *Twitter event* can be prolonged even though the mass media stops releasing new content. We determined a total of 421 < *trending news topics*, *Twitter events* > pairs that respect the time constraint and have a similarity of over 0.65. This process takes 31.2 minutes: 17.41 minutes to extract *trending news topics* and 13.79 minutes to determine the < *trending news topics*, *Twitter events* > pairs.

Table 6 showcases a subset of the correlation between the selected *news topics*, *news events*, and *Twitter events*, where the topic and event numbers can be found in Tables 3, 4, and 5. As we extract 1 000 *news events*, in Table 6 the *news events* also represent *trending news topics*. The presented correlations also include the best and the worst similarities.

Table 6: Correlation between topics and events

#NT	#NE	#TE	Sim NT NE	Sim NE TE
1	1	1	0.87	0.88
2	2	2	0.73	0.79
3	3	3	0.86	0.89
4	4	4	0.78	0.85
5	5	5	0.77	0.78
6	6	6	0.84	0.75
7	7	7	0.90	0.79
8	8	8	0.78	0.77
9	9	9	0.82	0.81
10	10	10	0.77	0.69

The matching between the pair < *news topics*, *news events* > and *Twitter events* shows that the news articles that appear in

Table 7: Unrelated Twitter Events

#TE	Start Date	End Date	Label	Keywords
1	2019-03-08 08:01:30	2019-07-12 14:01:30	cartoon	matt cartoonist telegraph side bobs cartoons
2	2019-04-12 08:01:30	2019-05-03 14:01:30	social media	whatsapp facebook videos mark zuckerberg user
3	2014-10-30 08:01:30	2019-05-21 02:01:30	game of thrones	spoilers season episode missed review sunday
4	2019-08-02 20:01:30	2019-08-14 02:01:30	sleep	coffee news lovers tea studying perfect ashes
5	2019-04-13 14:01:30	2019-07-20 02:01:30	rice	delicious perfectly sandwiches fried dish cheeses

mass media are discussed in detail during their publication on social media. Although not all *news topics* are directly related to a *Twitter events*, e.g., NT #9 discussed Trump's impeachment, and TE #9 discusses the abortion law passed in Alabama, the correlation between *news topics* and *news events* is high. Thus, these topics are considered *trending news topics*. Furthermore, some *trending news topics* match generalized *Twitter events*, e.g., NT #10 is related to the Kentucky derby, where the winning horse named Maximum Security was disqualified, while TE #10 is related to Manchester football clubs, but both talk about sports. Other matches are quite specific, e.g., NT #1 and TE #1, which both are related to European politics and Brexit.

In conclusion, a similarity over 0.77 between *news topics* and *news events* shows that the topics and events are consistent, while a similarity over 0.69 between *news events* and *Twitter events* denotes a generalization tendency of events for *trending news topics*. Furthermore, we found that all the *trending news topics* have correlations with at least one *Twitter event* and that some *trending news topics* are correlated to the same *Twitter events*.

We also analyzed the reverse correlation, i.e., *Twitter events* → *trending news topics*, by applying the same steps as for *trending news topics* → *Twitter events* but in reverse. We observe that the set given by the correlation *Twitter events* → *trending news topics* is the same as the one given by *trending news topics* → *Twitter events*.

However, as Twitter is a social media platform for generic discussion, some events discovered by MABED within the tweets collected are not related to the current *news events*. Thus, we can conclude that some thread discussions present generic topics. They spam for longer periods of time and the keywords are more generic. These events are related to common discussions regarding food, social media in general, television shows, etc. Table 7 presents some of the tweeter events that are not related to any of the news articles topics and do not match any *trending news topics* using our matching condition.

5.6 Audience Interest Prediction

Using the correlation between *trending news topics* and *Twitter events* we want to predict the interest of twitter users in *news topics* on social media using metadata, i.e., the user and its number of followers, and determine if a *news topic* becomes viral. We employ the MLP and CNN architectures to predict the audience interest in the *trending news topic*. The interest is given by both the number of retweets and likes received by the *Twitter Events* correlated to the *trending news topics*. We train each network on a machine with a 3.5GHz quad-core processor and 16GB RAM, using the Keras[10] library with TensorFlow[11] as backend.

Each network is trained until it converges, using an Early Stopping mechanism that checks if there are any changes in the loss function from one epoch to the next. We used the optimizers SGD and ADADELTA and different leaning rates (lr). After hyperparameter tuning and cross validation, we obtain the following configurations which have the best results:

- MLP 1 uses the MLP architecture with the SGD optimizer and a $lr = 0.5$,
- MLP 2 uses the MLP architecture with the ADADELTA optimizer and a $lr = 2$,
- CNN 1 uses the CNN architecture with the SGD optimizer and a $lr = 0.5$, and
- CNN 2 uses the CNN architecture with the ADADELTA optimizer and a $lr = 2$.

The input of the networks contains the document embeddings (Doc2Vec) for each tweet belonging to a *Twitter event* determined by the correlation < *Twitter events*, *trending news topics* >. Thus, as some tweets can belong to multiple events, the size of the Twitter dataset increases. The label for a tweet is determined by the number of likes and retweets, respectively, as presented in Table 2. We create 8 datasets using the custom document embeddings and the metadata vector, as presented in Section 4 as follows:

- A1 uses the SW_Doc2Vec model;
- A2 uses the SW_Doc2Vec model concatenated with the metadata vector;
- B1 uses the RND_Doc2Vec model;
- B2 uses the RND_Doc2Vec model concatenated with the metadata vector;
- C1 uses the SWM_Doc2Vec model;
- C2 uses the SWM_Doc2Vec model concatenated with the metadata vector;
- D1 uses the SW_Doc2Vec model;
- D2 uses the SW_Doc2Vec model concatenated with the metadata vector and the tweet's author number of followers.

We set the document embedding size to 300. The metadata vector has a size 8 which includes an one-hot-encoding vector for the tweets' author, i.e., the *influencer* and the its number of followers, of length 7 and one element for the day of the week. Thus, we include two behaviour factors in our embedding:

(1) a temporal one given by the day which incorporate user behaviour patterns, i.e., the patterns users have to post online;
(2) a global one given by the author metadata which integrates the impact of *influencers* on the virality of *trending news topics*.

For each experiment, the behavior of the network is described by the reduction slope of the loss function and the increasing slope of the accuracy function over the training dataset. Table 8 presents the measured accuracy results over our *validation* sets on predicting *likes*, while Table 9 presents the accuracy for predicting *retweets*. We define accuracy in terms of correct

[10] https://www.keras.io/
[11] https://www.tensorflow.org/

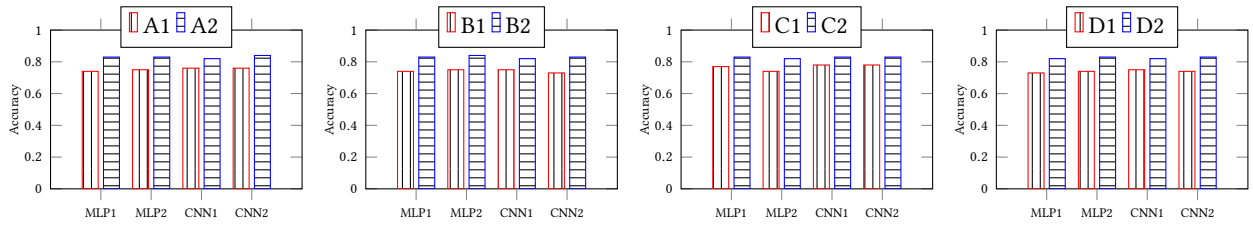

Figure 4: Likes accuracy comparison: without metadata (A1, B1, C1, and D1) vs. with metadata (A2, B2, C2, and D2)

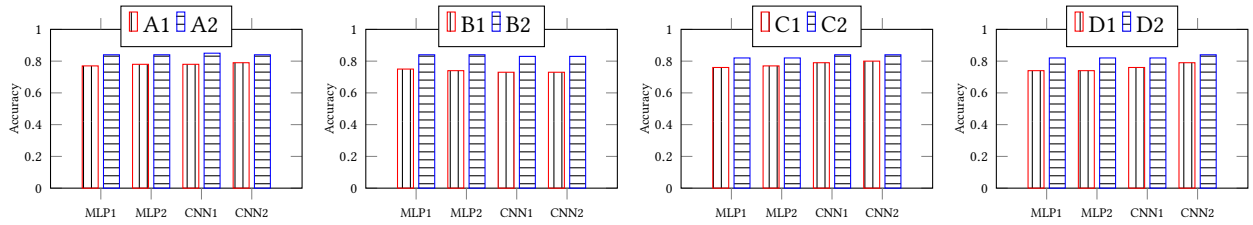

Figure 5: Retweets accuracy comparison: without metadata (A1, B1, C1, and D1) vs. with metadata (A2, B2, C2, and D2)

prediction in one of the 3 encoded classes for likes and retweets. Note that our accuracy results also reflect the error rate, since ErrorRate = 1 − Accuracy. As our results show, we successfully achieve low error rates by virtue of utilizing terms extracted using MABED and NMF; this success is based on the fact that similar topics/events have high similarity scores, while dissimilar ones have low similarity scores.

Table 8: Likes accuracy of correlated results

Dataset	MLP 1	MLP 2	CNN 1	CNN 2
A1	0.74	0.75	0.76	0.76
A2	0.83	0.83	0.82	0.84
B1	0.74	0.75	0.75	0.73
B2	0.83	0.84	0.82	0.83
C1	0.77	0.74	0.78	0.78
C2	0.83	0.82	0.83	0.83
D1	0.73	0.74	0.75	0.74
D2	0.82	0.83	0.82	0.83

Table 9: Retweets accuracy of correlated results

Dataset	MLP 1	MLP 2	CNN 1	CNN 2
A1	0.77	0.78	0.78	0.79
A2	0.84	0.84	0.85	0.84
B1	0.75	0.74	0.73	0.73
B2	0.84	0.84	0.83	0.83
C1	0.76	0.77	0.79	0.80
C2	0.82	0.82	0.84	0.84
D1	0.74	0.74	0.76	0.79
D2	0.82	0.82	0.82	0.84

We observe that both networks reach good results in terms of predicting the audience interest in specific *news topics* on social media, ranging from 0.73 to 0.85 accuracy, and accurately predict whether the *news topic* becomes viral. Furthermore, both architectures achieve similar accuracy scores regardless of the optimizer.

We conclude that these results are highly impacted by the MABED algorithm because by detecting the most trending topics and identifying the event for a tweet afterward, it basically extracts required features to predict a range of likes and retweets.

Interestingly, both architectures reach a prediction score of 0.75 after only a few epochs are finished. After these epochs, the learning process is slow and not very stable, exhibiting fluctuations between 0.78 and 0.90.

The high results are influenced by the metadata used to enhance the training corpus. We observe that the accuracy improves over the baseline with more 0.05 in some cases. This proves that the *influencer* role in spreading the news as well as the behaviour patterns of posting depending on the day have a huge impact on predicting the virality of a *trending news topic*.

The popularity of a person inside a group determines the spread of its messages, thus proving that the *influencers* assumption also holds for determining *trending news topics* on social media. In conclusion, a user that has many followers or tweets with many likes and retweets will maintain the ascending trend for future tweets with high probability. Also, we conclude that using the metadata vector improves the accuracy of prediction for all our experiments (Figures 4 and 5).

5.7 Scalability

To evaluate scalability, we extract 500, 2 500, and 5 000 *Twitter events*. We determine and encode using Doc2Vec the tweets for each event and train the networks. Table 10 presents the runtime evaluation for the networks using a batch size of 5 000 and 500 epochs. Early Stopping mechanism is used to stop the training. All the experiments are performed using the CPU. The training process for one epoch takes on average 1 second and 13/14 milliseconds for the MLP architectures (Figure 6), while the CNN architectures have a linear time increase from 1 second 71 millisecond to 6 seconds 83 milliseconds w.r.t. the number of events and the Doc2Vec size (Figure 7). This extra time is added by the complexity of the convolution layer and the number of kernel filters applied to the input vector. The CNN architectures maintain a much lower number of epochs than the MLP ones, regardless of the number of events or the Doc2Vec size. The difference between optimizers is not obvious at the batch level as

time performance and accuracy remains the same w.r.t. number of events and Doc2Vec size. We observe a difference in the number of batches, as the ADADELTA optimizer, on average, requires more batches until it converges than the SGD optimizer. We note that these results may vary, depending on the hardware used.

Table 10: Runtime evaluation

No. Twitter Events	Doc2Vec Size	Network	No. Epochs	Milliseconds Epoch	Runtime (Seconds)
500	300	MLP1	113	1013	119.51
		MLP2	119	1014	121.87
		CNN1	6	1071	6.67
		CNN2	7	1073	7.75
	308	MLP1	143	1013	154.53
		MLP2	162	1014	177.37
		CNN1	6	1073	7.16
		CNN2	8	1074	8.97
2 500	300	MLP1	316	1013	331.15
		MLP2	363	1014	381.52
		CNN1	6	3078	25.08
		CNN2	7	3079	26.95
	308	MLP1	319	1013	337.27
		MLP2	375	1014	392.24
		CNN1	6	4081	28.09
		CNN2	12	4082	52.69
5 000	300	MLP1	289	1013	334.76
		MLP2	305	1014	348.27
		CNN1	6	5097	31.85
		CNN2	7	5098	37.41
	308	MLP1	328	1013	351.87
		MLP2	368	1014	379.09
		CNN1	6	6081	38.49
		CNN2	14	6083	87.13

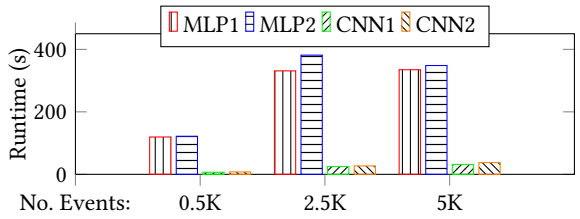

Figure 6: Performance time for 300-dimensions Doc2Vec

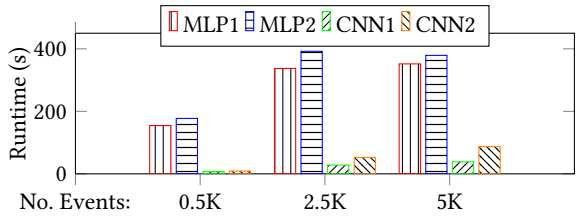

Figure 7: Performance time for 308-dimensions Doc2Vec

5.8 Discussion

Correlation between Trending News Topics and Twitter Events. To determine the *trending news topics*, we correlated the *news topics* with *news events*. Then, we correlated the *trending news topics* with the *Twitter events* that have a high cosine similarity and are in the same time window. We observe that for the same *trending news topics*, we can match multiple *Twitter events* using the imposed constraints. Thus, we can conclude that the events generated by posts on social media about *trending news topics* are intertwined. Therefore, we can conclude that some important *news topics* appear in multiple discussion and that users make connections between them in their discussion. Furthermore, all the *trending news topics* have at least one matching *Twitter event*.

We also applied the reverse correlation, by matching *Twitter events* to *trending news topics*. We discovered that we obtain the same set of pair for *Twitter events* → *trending news topics* as for *trending news topics* → *Twitter events*. During this analysis, we also discovered that multiple *Twitter events* have no correlated *trending news topics*. We attribute these results to the fact that Twitter is a generalized discussion forum. Thus, users do not address in their post only current *news events* but also discuss other events that impact their life but do not appear in mass media.

Audience interest prediction in Trending News Topics. We predict the audience interest in *trending news topics* by constructing two Deep Learning models to determine if they become viral. Thus, a *trending news topic* has a high probability of becoming viral if the posts that belong to the *Twitter event* correlated with it has a high number of likes and retweets.

For our experiments, we embedded the tweets by combining document embeddings with a metadata vector that contains the user's tweets and its number of followers as well as the day of the week. This enhanced tweet embedding feature adds two new dimension within the learning model: a temporal one given by the day and a global one given by the user metadata. The experimental results show that the two models determine with high accuracy if a *trending news topic* becomes viral for both the datasets constructed using only documents embeddings and the ones that also use the metadata vector.

Through the use of the metadata vector, we incorporate in our training set two assumptions. The first assumption is a global one and introduces the concept of *influencers* (users with a high number of followers) in the learning model. The *influencers* have a huge role in spreading the information and making *trending news topics* to become viral. The second assumption is a temporal one which introduces the user behavior based on the day of the week, as behaviour patterns can change from one day to another. Using the assumption, the neural networks learn to determine patterns in the way users post on social media and integrate them in the final prediction model. The experimental results prove that this assumption stands for our use case. Both Deep Learning models achieve an improved accuracy when the metadata vector is concatenated to the document embedding. Thus, we can conclude that the *influencers* metadata manages to create models that better generalize while improving the overall performance.

Fake news mitigation considerations. The spread of misinformation in social media has the effect of polarizing opinions and misleading readers by presenting alleged, imaginary facts about social, economic, and political subjects of interest [16]. Our method manages to predict the virality of news content and the interest the public at large has in different news topics. These findings can prove useful in designing new mitigation algorithms that take into account both textual content and network metadata. Thus, we consider that the presented system manages to determine what topics are important and can be a starting point to develop new strategies for network immunization in the fight against misinformation.

6 CONCLUSIONS

In this work, we introduce and deploy an architecture for predicting whether a *trending news topic* becomes viral on social media. We employ NMF topic modeling to extract *news topics* and MABED event detection to extract *news events* and *Twitter events*. We correlate *news topics* to *news events* by document embedding cosine similarity to extract *trending news topics*, match *trending news topics* to *Twitter events* in the same time window, and thereby extract likes, retweets, and user followers. We propose a new metadata-based document embedding for tweets associated to an event that encodes *influencers* information. Using the new document embeddings, we prepare a training corpus to predict audience interest in a *trending news topic*, using two Deep Learning architectures.

The similarity between *news topics* and *news events* in our data is over 0.77, while that between *trending news topics* and *Twitter events* is over 0.69. Our results show that the *trending news topics* and the *news events* are consistent, while the proposed architectures predict audience interest in a *news topic*, with accuracy over 0.82 in the metadata enhanced dataset.

In the future, we plan to use other matching techniques, e.g., Minimum Cost Flow, to correlate *news topics*, *news events*, and *Twitter events*, and use topic modeling [7] to see if we can extract more coherent topics from news. We also plan to employ word, sentence, and document level transformers (BERT [10], XLNet [37], ALBERT [22], ELECTRA [8]) as embeddings to take advantage of contextual information extracted using these models. Further, our solution can be included in larger solutions for fake news mitigation and misinformation immunization strategies for social media.

This work was done in collaboration with RoNews, a start-up offering news media content, which was interested in detecting article topics and verifying content veracity. Hootsuite and Sprinklr have also presented interest in this solution.

REFERENCES

[1] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. 2008. On-line lda: Adaptive topic models for mining text streams with applications to topic detection and tracking. In *IEEE International Conference on Data Mining*. 3–12.

[2] Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning Topic Models – Going beyond SVD. In *Annual Symposium on Foundations of Computer Science*. 1–10.

[3] Frank Bentley, Katie Quehl, Jordan Wirfs-Brock, and Melissa Bica. 2019. Understanding Online News Behaviors. In *ACM Conference on Human Factors in Computing Systems*. 1–11.

[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

[5] Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2019. Next cashtag prediction on social trading platforms with auxiliary tasks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 525–527.

[6] Xueqin Chen, Fan Zhou, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Fengli Zhang. 2019. Information diffusion prediction via recurrent cascades convolution. In *IEEE International Conference on Data Engineering*. IEEE, 770–781.

[7] Yong Chen, Hui Zhang, Rui Liu, Zhiwen Ye, and Jianying Lin. 2019. Experimental explorations on short text topic mining between LDA and NMF based Schemes. *Knowledge-Based Systems* 163 (2019), 1–13.

[8] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *International Conference on Learning Representations*. 1–17.

[9] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*. 4171–4186.

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12, 61 (2011), 2121–2159.

[12] Orhan Erdem, Elvan Ceyhan, and Yusuf Varli. 2014. A new correlation coefficient for bivariate time-series data. *Physica A: Statistical Mechanics and its Applications* 414 (2014), 274–284.

[13] Ahmad Foroozani and Morteza Ebrahimi. 2019. Anomalous information diffusion in social networks: Twitter and Digg. *Expert Systems with Applications* 134 (2019), 249–266.

[14] Guiyuan Fu, Feier Chen, Jianguo Liu, and Jingti Han. 2019. Analysis of competitive information diffusion in a group-based population over social networks. *Physica A: Statistical Mechanics and its Applications* 525 (2019), 409–419.

[15] Adrien Guille and Cécile Favre. 2014. Mention-anomaly-based event detection and tracking in twitter. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 375–382.

[16] Imane Hafnaoui, Gabriela Nicolescu, and Giovanni Beltrame. 2019. Timing Information Propagation in Interactive Networks. *Scientific Reports* 9, 1 (2019).

[17] Thomas Hofmann. 2000. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization. In *Advances in Neural Information Processing Systems*. 914–920.

[18] Maryam Hosseini-Pozveh, Kamran Zamanifar, and Ahmad Reza Naghsh-Nilchi. 2019. Assessing information diffusion models for influence maximization in signed social networks. *Expert Systems with Applications* 119 (2019), 476–490.

[19] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to Chaotic Whispers. In *ACM International Conference on Web Search and Data Mining*. 261–269.

[20] Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *ACM International on Conference on Information and Knowledge Management*. 1411–1420.

[21] Jaeyoung Kim, Sion Jang, Eunjeong Park, and Sungchul Choi. 2020. Text classification using capsules. *Neurocomputing* 376 (2020), 214–221.

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*. 1–17.

[23] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[24] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225.

[25] Xiaoyan Lu and Boleslaw K. Szymanski. 2018. Scalable Prediction of Global Online Media News Virality. *IEEE Transactions on Computational Social Systems* 5, 3 (2018), 858–870.

[26] Luis Miguel Matos, Paulo Cortez, Rui Mendes, and Antoine Moreau. 2019. Using Deep Learning for Mobile Marketing User Conversion Prediction. In *International Joint Conference on Neural Networks*. 1–8.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*. 1–12.

[28] Ben Moews, J. Michael Herrmann, and Gbenga Ibikunle. 2019. Lagged correlation-based deep learning for directional trend change prediction in financial time series. *Expert Systems with Applications* 120 (2019), 197–206.

[29] Reza Motamedi, Soheil Jamshidi, Reza Rejaie, and Walter Willinger. 2019. Examining the evolution of the Twitter elite network. *Social Network Analysis and Mining* 10, 1 (2019), 1–18.

[30] Emilio Serrano, Carlos A. Iglesias, and Mercedes Garijo. 2015. A Survey of Twitter Rumor Spreading Simulations. In *Computational Collective Intelligence*. 113–122.

[31] David A. Shamma, Lyndon Kennedy, and Elizabeth F. Churchill. 2011. Peaks and Persistence: Modeling the Shape of Microblog Conversations. In *ACM Conference on Computer Supported Cooperative Work*. 355–358.

[32] Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management* 45, 4 (2009), 427–437.

[33] Kjerstin Thorson, Kelley Cotter, Mel Medeiros, and Chankyung Pak. 2019. Algorithmic inference, political interest, and exposure to news and politics on Facebook. *Information, Communication & Society* (2019), 1–18.

[34] Neil Thurman, Judith Moeller, Natali Helberger, and Damian Trilling. 2018. My Friends, Editors, Algorithms, and I. *Digital Journalism* 7, 4 (2018), 447–469.

[35] Ciprian-Octavian Truică, Florin Radulescu, and Alexandru Boicea. 2016. Comparing different term weighting schemas for Topic Modeling. In *IEEE International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. 307–310.

[36] Cheng Yang, Jian Tang, Maosong Sun, Ganqu Cui, and Zhiyuan Liu. 2019. Multi-scale Information Diffusion Prediction with Reinforced Recurrent Networks.. In *IJCAI*. 4033–4039.

[37] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russlan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*. 5753–5763.

[38] Zhenguo Yang, Qing Li, Liu Wenyin, and Jianming Lv. 2019. Shared Multi-view Data Representation for Multi-domain Event Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 5 (2019), 1243–1256.

[39] Matthew D. Zeiler. 2012. Adadelta: an adaptive learning rate method. *CoRR* abs/1212.5701 (2012), 1–6.