applied sciences

MDPI

Article

User Representation Learning for Social Networks: An Empirical Study

Ibrahim Riza Hallac *, Betul Ay and Galip Aydin

Department of Computer Engineering, Firat University, 23119 Elazig, Turkey; betulay@firat.edu.tr (B.A.); gaydin@firat.edu.tr (G.A.)
* Correspondence: irhallac@firat.edu.tr

Abstract: Gathering useful insights from social media data has gained great interest over the recent years. User representation can be a key task in mining publicly available user-generated rich content offered by the social media platforms. The way to automatically create meaningful observations about users of a social network is to obtain real-valued vectors for the users with user embedding representation learning models. In this study, we presented one of the most comprehensive studies in the literature in terms of learning high-quality social media user representations by leveraging state-of-the-art text representation approaches. We proposed a novel doc2vec-based representation method, which can encode both textual and non-textual information of a social media user into a low dimensional vector. In addition, various experiments were performed for investigating the performance of text representation techniques and concepts including word2vec, doc2vec, Glove, NumberBatch, FastText, BERT, ELMO, and TF-IDF. We also shared a new social media dataset comprising data from 500 manually selected Twitter users of five predefined groups. The dataset contains different activity data such as comment, retweet, like, location, as well as the actual tweets composed by the users.

Keywords: distributed representation; user representation; user embedding; social networks

Citation: Hallac, I.R.; Ay, B.; Aydin, G. User Representation Learning for Social Networks: An Empirical Study. *Appl. Sci.* **2021**, *11*, 5489. https://doi.org/10.3390/app11125489

Academic Editor: Carlos A. Iglesias

Received: 28 April 2021
Accepted: 11 June 2021
Published: 13 June 2021

1. Introduction

Objects are expressed by their properties, that is, by the elements and components of which they are composed. One of the topics of artificial intelligence, which has developed with increasing momentum in recent years, is the development of methods to understand the world we live in and to represent the entities in it.

In the past decade, there have been exciting developments in the representation of words and documents in the semantic space. Especially with the advent of deep learning methods, natural language processing approaches that were still under the influence of mechanical methods changed course to mimic human learning abilities. New techniques that can boost almost any Natural Language Processing (NLP) task are constantly emerging. One way of illustrating the power of these methods is to apply analogical reasoning based on the resulting word vectors and examine their performance on capturing semantic and syntactic word relationships [1]. For example, if vec(w) is a function that maps a word "w" to a dense vector, which is often referred to as word embedding, by applying element-wise subtraction and addition, vec("king") − vec("men") + vec("women") operation results in a vector that has the closest cosine similarity with vec("queen"). Similarly, vec("Istanbul") − vec("Turkey") + vec("France") = vec("Paris") indicates the correctness of the word embedding model. This analogy is often used in the evaluation of the word embedding models [2,3].

A similarity measure is used in linear analogical reasoning calculation, which is usually Euclidean Distance or Cosine Similarity. Word vectors can even reflect some instances of bias such as racism and sexism based upon the bias existing in the training

corpus. However, there are other ways to show the power of word vectors. Word vectors are used against other word representation methods in NLP problems such as clustering and classification. Their performance is analyzed with task-oriented evaluations.

It is possible to represent social media users in semantic space, just like words, and to conduct various analogies as a measure of accuracy between these representations. One of the most difficult challenges in social media research is dealing with complex unstructured textual and network data. The use of emerging word embedding and document embedding techniques has been enhancing information retrieval, recommendation systems, security, and many others [4]. State-of-the-art distributed semantic representation techniques can be categorized into three classes [5]: classical word embeddings, character-level features, and contextualized word embeddings. Definitions of the state-of-the-art text representation methods that are used in this study are given as follows:

Pre-Trained Word Embeddings are general purpose word vectors that are learned by training over large text corpora. Using pre-trained word embeddings is considered as a kind of transfer learning as they are used in improving the performance of other text processing tasks such as document classification and clustering, entity recognition, etc. Word embeddings are used as input features in deep learning models for solving NLP problems. Word2vec and Glove are the best-known word-embedding algorithms.

Word2Vec is a neural network-based algorithm for training word embeddings [6]. It learns the semantic similarities between words with a simple fully connected layer. Word2vec is based on the "You shall know a word by the company it keeps" idea of distributional hypothesis, which means that similar words are likely to appear close to each other in a corpus [7]. This results in similar words having closer vectors in the semantic space. Word2vec algorithm introduces two architectures: Skip-gram and CBOW. While Skip-gram method learns the embeddings by predicting the surrounding context words of a given word, CBOW method predicts a target word given the context words.

Global Vectors (also known as GloVe) is a log-bilinear regression model for word representation [8]. This model learns word embeddings using word co-occurrence matrix given a corpus and captures the global corpus statistics. In this aspect, while Word2vec is considered as a predictive model, GloVe is considered as a count-based model that takes into account statistics of word occurrences.

FastText library [9] provides a simple and efficient method for word embeddings in text classification tasks. Unlike word2vec, FastText uses n-grams so that the local word orders are not ignored.

Flair is an NLP framework for text classification and language models [10]. Flair provides a simple interface for different types of word embeddings and language models such as word2vec, GloVe, and Bert.

BERT (Bidirectional Encoder Representations from Transformers) [11] and ELMO (Embeddings from Language Models) [12] have started to transform the word embedding methods by introducing context aware approaches. Prior to these models, the state-of-the-art word representation techniques were context-independent models such as Glove, Word2vec, and FastText, which does not take into account word order in their training phase. Also, once their training phase is completed, they produce one vector for each word and the model is not needed afterwards. However, for the context aware approaches, when they are used in the downstream NLP tasks, their model is needed to produce vectors for words or sentences because the resulting representations vary each time depending on the input. For example, in the sentence "The mayor of Batman plans to sue DC Comics for using his city's name in their Batman Comic book franchise" the word "Batman" has two very different meanings. First meaning is the city name in Turkey and the second is the fictional superhero from the movies like "The Dark Knight". In a static word representation method, the word Batman has only one vector, but in a dynamic representation system its vector is produced depending on the context.

ELMO uses a character-based representation of the input text. Then, it applies character convolutions to these representations to obtain word level embeddings. The main part

of the model is the LSTM (long short-term memory) layers where two language models are employed in opposite directions. The forward (right-to-left) and backward (left-to-right) representations obtained from the LSTM layers are concatenated in the output layer. BERT model, on the other hand, uses a newer neural architecture called transformers [13]. BERT uses sub-word tokens of the input text and constructs words from these tokens. Then, it is pre-trained on large unlabeled data for two unsupervised tasks, which are Next Sentence Prediction and Masked Language Model. After the pre-training is completed, the model is used in downstream NLP tasks.

ConceptNet Numberbatch provides multilingual representations of words built on a combination of semantic knowledge graph and existing pre-trained word embeddings such as Glove and word2vec [14]. It can leverage a human-made knowledge graph by using the retrofitting method for improving the vector values of pre-trained embedding models [15].

Doc2vec model is a neural document embedding technique also known as Paragraph Vector. It is an extension to the word2vec model for the representation of single sentences or documents [16]. In this model, a paragraph identifier is added to the input of the neural network created to learn word vectors, which will be called document id or paragraph id (1-hot encoded vector of the documents). Thus, the neural network is able to take the document id as input and learn the vector representations of the documents of different word lengths. Paragraph vector model presents two different architectures: Distributed Memory (PV-DM) and Distributed Bag-of Words (PV-DBOW). PV-DM works in a similar way to the CBOW word2vec model, while PV-DBOW works in a similar way to the skip-gram word2vec model.

In the framework of this study, the vector space representation of the social media (Twitter) account is interchangeably referred to as user representation, user embedding, or user vector. Our assumption is that the user vector of a person from the fashion world would be closer to another person from the same world compared to a person from the world of politics. Furthermore, we presume the user vector of a democratic politician would be more similar to other democratic users than the representations of republican users and vice versa. Using Twitter as an example, we present user representation models that leverage state-of-the-art text analysis techniques to generate semantic representations of users in vector space that can be used in the task of automatically generating meaningful observations for users of a social network. The effectiveness of the user embeddings was demonstrated through testing on a gold standard dataset that we prepared. In these evaluations, in addition to using different algorithms such as doc2vec, ELMO, and BERT, we explored in detail which type of dataset is preferable.

This study is a continuation of our earlier work on user representation based on distributed document embeddings (user2vec) [17]. In the previous study, a gold standard dataset was created with tweets of manually selected users. User embeddings were obtained by averaging the document embeddings of their tweets. The document embeddings of each tweet were learned from a doc2vec model trained on the dataset. Different pre-processing approaches were compared and it was found that general pre-processing methods (removing the stop words and words that contain non-alphanumeric characters) gave better results. In addition, the effect of the vector size in parameter selection was investigated by training the model up to 2000 iterations for different vector sizes (20, 30, 40, 50, 100, 200, and 300). When training the Doc2vec model, only tweets of the users whose embeddings were to be obtained were used as the corpus. The document vectors were obtained using the tags of the documents (`Gensim: doc2vec_model['document_tag']`).

The contributions of this study are as follows:

(a) A method such as the `Gensim-doc2vec_model['document_tag']` cannot be used in the case of extracting embeddings for new tweets that are not included in the corpus used in the training of the doc2vec model. In this case, after training the model, `infer_vector()` function is used to infer vectors for new text. This generative method is not fully deterministic and subsequent calls for the same text produce

different vectors, however, they would be close to each other in the latent-space. The performance of the models is assessed when no information (tweet, comment, followers, etc.) of the users to be represented are included in the training data. See Appendix A for some basic information on the Gensim Framework.

(b) It is not always feasible to have a gold standard training set consisting of manually selected users. An important question is whether there are alternative resources for training user representation models. Therefore, utilization of available and easily/readily accessible textual resources such as Twitter and Wikipedia are examined. Also, combinations of the datasets are mixed together in different scenarios.

(c) A wide range of information (interests, opinions, sentiment, or topic) can be present in the tweet text. NLP techniques are used in extracting information from tweets, which consist of words, hashtags, mentions, smileys, etc. However, there is so much more to leverage from when representing a Twitter user. Twitter handles of users who engaged with the tweets by Retweeting, Liking, and Replying are collected into a dataset. Another dataset is created with the user's profile information: handle, name, description, and location. Several approaches are proposed for training doc2vec-based user representation models on the aforementioned datasets that embody both textual and non-textual information.

(d) The performance of PV-DM and PV-DBOW algorithms are compared. Also, the effect of the `min_count` parameter (`min_count=1` and `min_count=5`) is investigated for each algorithm. (see Appendix A for the explanations of these parameters)

(e) A traditional method, TF-IDF; pre-trained word embeddings (PWE) such as Glove, FastText; and pre-trained sentence encoders (contextualized embeddings) such as BERT, ELMO have been studied in various aspects in obtaining document vectors.

A detailed comparison is conducted among the proposed methods based on their performance in terms of obtaining embeddings of the test users correctly. In the evaluation of the vector representation accuracy, a user vector is determined as correct if it positions the user in the vector space close to the users of the same group and distant to users of the other groups. Precision, Recall, and F1 scores are presented in the paper.

2. Related Work

The popularity of social networking sites has undeniably increased over the past decade. There are over 4 billion internet users in the world. As of 2021, Twitter, a micro-blogging service, has 330 million active users. In the simple case, where no information about users is encoded, they are represented by a 330 million-dimensional one-hot-encoded vector.

With recent advances in Deep Learning, NLP techniques have been widely applied in social media analysis tasks. Particularly, text representation methods play an important role in solving topic detection, named entity recognition (NER), sentiment analysis, and content classification problems. In [18], an encoder-decoder deep learning model was proposed for identifying occupations in COVID-related tweets. Samad et al. [19] investigated the performance of different tweet vector representation approaches for three-way sentiment classification (positive, negative, and neutral) task over an imbalanced tweet dataset.

Obtaining dense vector representations of social media users has the potential to pave the way for social media prediction and recommendation approaches such as [20–22]. Recent works have shown the effectiveness of leveraging text-based representation methods on distributed semantic representation of things (items, users, etc.) in a variety of applications [23,24]. In a study by Facebook [25], they used embedding tables to transform sparse features of user-item pairs to dense vectors in personalized recommendation tasks.

User representations can be used in various applications such as gender prediction [26] and identifying personality traits [27,28]. Computation of similarity/dissimilarity among objects is an essential data-mining task [29]. One promising line of research considers finding similarities of users in digital platforms [30–32].

A wide range of information exists on social networks that can be incorporated into the algorithms for learning user representations. The capability to keep and process all possible data together is only possible for large companies such as Twitter or Facebook. However, it is possible to carry out extensive analyses with publicly available data. The main profile information (picture, name, description, location), followers, followees, and the recent posts are the first things that are considered when a user's profile is analyzed manually. Then, a lot more details can be obtained such as who likes their posts, comments, and reshares (resharing links, content etc.). There is a need for algorithms on how to take these features into account when creating solutions for analyzing users automatically.

3. Datasets

An important goal of this study is to explore the performance of different types of datasets in the training of user representation models. A great deal of work is undertaken in the preparation of the datasets. In the previous study [17], 100 Twitter users from each of the fields of Economy, Crypto Economy, Politics, Technology, and Fashion were manually selected and the most recent 1000 tweets, which had been composed by these 500 users, were collected using the Twitter REST API. With this method, a gold standard dataset consisting of 500K labeled (user, group) tweets was built.

A separate Twitter profile was created to be used in the process of selecting users for each category. A small number of users were selected as a starting point by examining various articles and blogs on the internet such as "Top Economics Influencers" and "Who to follow on Twitter for technology". We then identified a total of 500 profiles by visiting more than 5000 profiles with the help of the automatic "follow" suggestions made by Twitter. We considered the presence of the following criteria in the manual examination of the profiles:

- Relevance: High degree of relevance to the specified category.
- Popularity: High number of followers (at least a few thousand followers).
- Posting behavior: Being an active Twitter user who tweets occasionally.

Sixty percent of the users in each group are separated and a training dataset is created using the tweets of these randomly selected users. The tweets of the remaining 40% of the users are used as the test dataset. The training dataset is named as 300K-TW.

300K-TW is used as a source and expanded by enriching it with additional information to create two new datasets. Location and description information of the users were also collected with Twitter REST API. We call the username and location dataset as NL. A related study proposed an approach for combining such non-textual discrete features for better user representations [33]. It should be noted that some users do not share their location or description information. A second dataset named as NLD-RLC is created with combining the data from NL with user description and some additional network features. These additional features are lists of users' names that engaged with the tweets in any of the following ways: retweet, like, or comment. The purpose of the NL and NLD-RLC datasets are to enrich the feature pool, therefore, they are not intended to be used solely by themselves. All of the datasets are briefly described in Table 1. A test dataset version of each of the 300K-TW, NL, NLD-RLC datasets also exists. They are obtained in the same fashion described above but are not given separately merely for reasons of simplicity.

Table 1. Datasets.

Name	ID	Type	Source and Description	Size Information
300K-TW	A	Text corpus	Tweets of manually selected Twitter users for five pre-determined groups. This is the text-only dataset used in our previous study.	- 60 users from each category - 1K tweets from each user; 300K tweets in total.
1M-Wiki	B	Text corpus	English Wikipedia dump dated 2019.06.20. It contains 4.672.834 Wikipedia entity pages.	1M randomly selected documents having minimum 50 and maximum 1K characters.
1M-TW	C	Text corpus	News Outlet Tweet IDs [34]. It contains 39,695,156 tweet IDs. (*)	Total of 1M randomly selected tweets.
NL	D	Social network data	Data set consisting of username and location information of each user in 300K-TW dataset.	Information of 240 users.
NLD-RLC	E	Text corpus and Social network data	Description information is added to the NL dataset. This NLD dataset is merged with a list of users who retweeted/liked/commented the tweets (RLC) in 300K-TW.	Information of 300K tweets and 240 users.

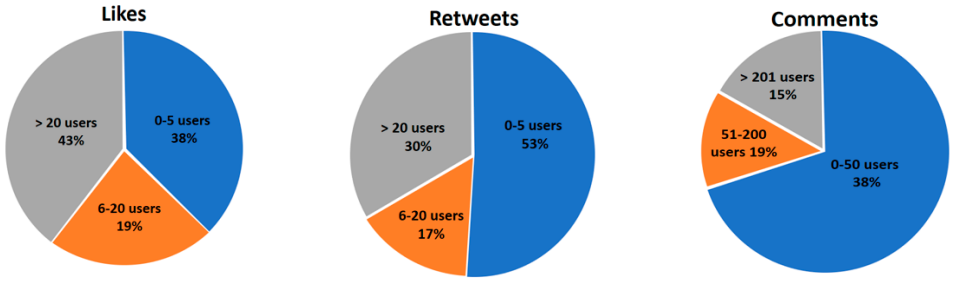
* The original dataset contains only the tweet IDs (dehydrated). Text of the tweets are rehydrated with their IDs using Twarc [35]: a tool for harvesting Tweets. The tweets of this dataset are consisted of tweets that were collected from the Twitter accounts of big media organizations such as newspapers and television channels.

As far as we know, collecting interaction data over semantically grouped social media users is an untested approach. The NLD-RLC dataset contains the usernames of the Twitter accounts that interacted with the tweets in the 300K-TW dataset. The standard Twitter API and the wrappers implemented in different programming languages that built around it such as Tweepy, Twarc, Twitter4J, and Bear, provide ways to collect "retweet", "comment" and "like" activity data of Twitter accounts. In other words, one can get the list of tweets that are retweeted/commented/liked by a user through the APIs (within the bounds of rate-limiting). However, to our knowledge, getting the list of users who retweeted/commented/liked a given tweet is not provided by the free, Standard Twitter API, yet there are some workarounds for this. However, they are mostly insufficient, because the availability of the results is particularly affected by the popularity of the accounts and by the time factor. Limits also occur in the case of manually viewing the lists on a web browser. Twitter shows only up to 100 users. The NLD-RLC dataset was scraped by automating the interactive steps of browsing the lists of each tweet in the 300K-TW dataset. The automation is achieved by writing a script for this task using the Selenium web driver API and the BeautifulSoup scraping library. A PC is configured to run this script 24/7 for 4 months and users' Twitter handles are crawled and saved. Each user has 1000 tweets and it takes about 4–8 h for harvesting the lists for one user.

The details of this dataset are as follows:

- There is not a fixed number of users who retweeted/commented/liked a tweet. The script collects only the handles of the users that are shown at that moment.
- A total of 3,599,588 different Twitter handles were collected. Their distributions for different interactions are shown in Figure 1. For example, it shows that:
    - for 38% of the tweets, there are found to be up to five handles of the users who liked the tweet.
    - for the 17% of the tweets, there are found to be 6–20 handles of users who retweeted the tweet.

- The number of handles for each group are shown in Table 2. For example, it shows that a total of 247,109 handles are collected from the retweet, like and comment lists of the tweets of the Economy group's users.
- The users that appear in the user lists of all five groups (Intersection of handles among groups) $G1 \cap G2 \cap G3 \cap G4 \cap G5 = 1171$.



(**a**) (**b**) (**c**)

Figure 1. Distributions of number of handles for (**a**) like, (**b**) retweet, and (**c**) comment lists.

Table 2. Group descriptions and number of interactions.

Group Id	Group Name	Number of Unique Handles
G1	Economy	247,109
G2	Crypto Economy	340,653
G3	Technology	649,865
G4	Fashion	803,920
G5	Politics	1,278,024

Note: For the convenience, groups are referred with their IDs.

The NLD-RLC is a complex dataset that contains both textual and non-textual features. The interactions with the tweets are represented by the lists of handles that appear in the retweet/comment/like activities. Each list consists of hundreds of thousands of handles, which are distinct values similar to elements in sets. We present some similarity measurements between the lists in order to guide subsequent analyses over the groups. A naive approach is to count the number of elements that are common between each pair of lists. As it can be seen in Table 3, Economy and Politics groups have the most common number of handles in their interaction lists. These are also the groups with the lowest and greatest number of handles. This shows that this method does not provide insight into the similarity of lists. Jaccard [36] and Overlap Coefficient (Szymkiewicz–Simpson coefficient) metrics [37] are commonly used to calculate similarities between units with different numbers of discrete features.

Table 3. Intersection of the lists among the groups.

∩	G1	G2	G3	G4	G5
G1	450,162	57,347	65,247	15,561	**122,840**
G2		439,652	44,633	6782	25,580
G3			817,242	29,839	86,233
G4				887,604	57,264
G5					1,515,083

The Jaccard similarity of groups Gx and Gy is $|Gx \cap Gy|/|Gx \cup Gy|$, which is the size of the intersection of sets Gx and Gy to the size of their union. Since the number of usernames of the two groups and the number of common usernames between them are known, the size of their union is calculated as follows: $|Gx \cup Gy| = |Gx| + |Gy| - |Gx \cap Gy|$.

In overlap coefficient the similarity is calculated with $|Gx \cap Gy|/\min(|Gx|, |Gy|)$. The similarity values for the two metrics are given in Table 4.

Table 4. Jaccard and overlap coefficient similarities of the lists.

Jaccard Similarity	G1	G2	G3	G4	G5	Overlap Coefficient	G1	G2	G3	G4	G5
G1	1	**0.069**	0.054	0.012	0.067	**G1**	1	0.130	0.145	0.035	**0.273**
G2		1	0.037	0.005	0.013	**G2**		1	0.102	0.015	0.058
G3			1	0.018	0.038	**G3**			1	0.037	0.106
G4				1	0.024	**G4**				1	0.065
G5					1	**G5**					1

The NLD dataset is prepared by combining the name, location, and description information of the tweets' authors. The NLD-RLC dataset is prepared by additionally augmenting each data in the NLD dataset with the handles of the users who retweeted, commented, and liked the tweets. The details of the applied data enrichment steps for the "300K-TW + NLD-RLC" datasets are explained in Section 4.2. Similar to [33], features are padded with dummy tokens for preventing non-textual units to be in the same window with the tokens of the tweet text. The list of used dummy tokens is "enrtag", "nametag", "loctag", "desctag", "liketag", "rttag", "commtag". The "enrtag" is used to separate the enrichment meta data from the tweet text. The maximum window size is selected as 5 for the doc2vec models in this study. Therefore, the related dummy token is added five times before and five times after each feature.

4. Methodology

The goal of the study is to evaluate different techniques for training a user embedding model. A, B, C, D, and E are the datasets used in the training of different models. It is also compared how well the models perform under different dataset scenarios. Input of a model can be a user's tweet text, user profile information, user activity data, or a combination of them depending on the proposed approach. The output of the model is a fixed size vector of R^d. M and N represent the number of training and test users respectively and they are assigned to S predefined groups. The output of the model is evaluated by computing the similarities between training and test users. A prediction is considered correct if the user vector is most similar to the averaged group vectors of the training users of the same group.

In all approaches, a d-dimensional $\overrightarrow{T} \in R^d$ vector is generated for each tweet in the data set. This phase is carried out with a different text representation method (*Embedding_Model*) for each experiment. This is the part that differs from experiment to experiment. Then, embedding of users is obtained by element-wise-averaging of all their tweet vectors:

$$\frac{1}{K} \sum_{v=1}^{K} Embedding\_Model(T_v) \quad (1)$$

Here, K represents the number of tweets that a user has. Group embeddings are calculated for each group $\theta_k (k = 1, \ldots, S)$ by element-wise-averaging of training user embeddings $\alpha_i (i = 1, \ldots, M)$ of that group. The correctness of the embeddings of the test users $\beta_j (j = 1, \ldots, N)$ is determined by comparing their distance to the embedding of the group they belong to with embeddings of other groups. Euclidean distance similarity measure is used in the distance evaluation [38]. $\left\{ E^i_\alpha, E^j_\beta, E^k_\theta \in R^d \right\}$ are a set of embeddings where E^i_α shows the embedding of a training set user α_i, E^j_β shows the embedding of a test set user β_j, and E^k_θ shows the average embedding for group θ_k calculated over training users.

$$E^k_\theta = \frac{S}{M} \sum_{i=1}^{M} E^i_\alpha \; where \; \alpha_i \in \theta_k \quad (2)$$

$d_{jk} = \left| E_{\beta}^{j} - E_{\theta}^{k} \right|$ shows the euclidean distance between a test user j's embedding and average embedding of group k; we count the number of correctly predicted user embeddings, i.e., true positives, by the number of true operations on $\left\{ d_{jk} < d_{j\gamma} \right\}$ where $\beta_j \in \theta_k$ and $\beta_j \notin \theta_\gamma$ for $\forall\ j = 1, \ldots, N$. Steps 1-2-3 in Figure 2 show the stages of obtaining text representations and are applied differently for each method. Steps 4-5-6 are applied in the same way in all approaches.

In the implementation of the proposed user representation method, there is, in fact, no need for a dataset built from manually selected users of predefined categories. It is a fully unsupervised approach. That is, it does not consider the group labels in the training phase. However, such a dataset is used to generate group embeddings, and the group embeddings are used in the evaluation of user embeddings. The extensive experiments of the proposed method are tested with applying different text representation techniques on various datasets. These approaches are examined under three main sections:

- Doc2Vec Approach
- Using non-textual features in Doc2Vec (enrichment)
- PWE Approaches

4.1. Doc2Vec Approach

In this approach, tweet vectors are extracted through the Doc2Vec algorithm on a specified corpus. This step is conducted as in our previous study [17].

$$\frac{1}{L} \sum_{c=1}^{L} \log P\left(w_c \middle| T, w_{c-p}, \ldots, w_{c+p}\right) \tag{3}$$

$\overrightarrow{T}$ denotes the vector representation of a tweet, w represents words in the tweet with length L as $\{w_1, w_2, w_3, \ldots w_L\}$, and p denotes the width of the sliding window of words centered on w_c.

Embeddings of training and test users are derived from their tweet vectors. The stages of this approach are briefly shown in Figure 2. Doc2vec models are trained on different datasets by using different parameters. Dimension size parameter is determined as 100 in all of the document models throughout this study. The role of dimension size was investigated in detail in the previous study. For this reason, and to better observe the effects of other parameters, the effect of dimension size is not examined. Seven different experiments are determined according to the type of corpus used in the training of the model. They are shown in Table 5.

Table 5. Experimental setup #1.

Case No.	Dataset	Scenario
1	A	Use of tweets belonging to a limited number of users considering the domains of the users handled in the problem.
2	B	Use of a big, general-purpose text corpus such as Wikipedia, News, etc.
3	C	Use of a large number of crawled tweets such as tweets from a specific time period.
4	A, B	Use of various datasets (A, B, C) that are put together in different combinations.
5	B, C	
6	A, C	
7	A, B, C	

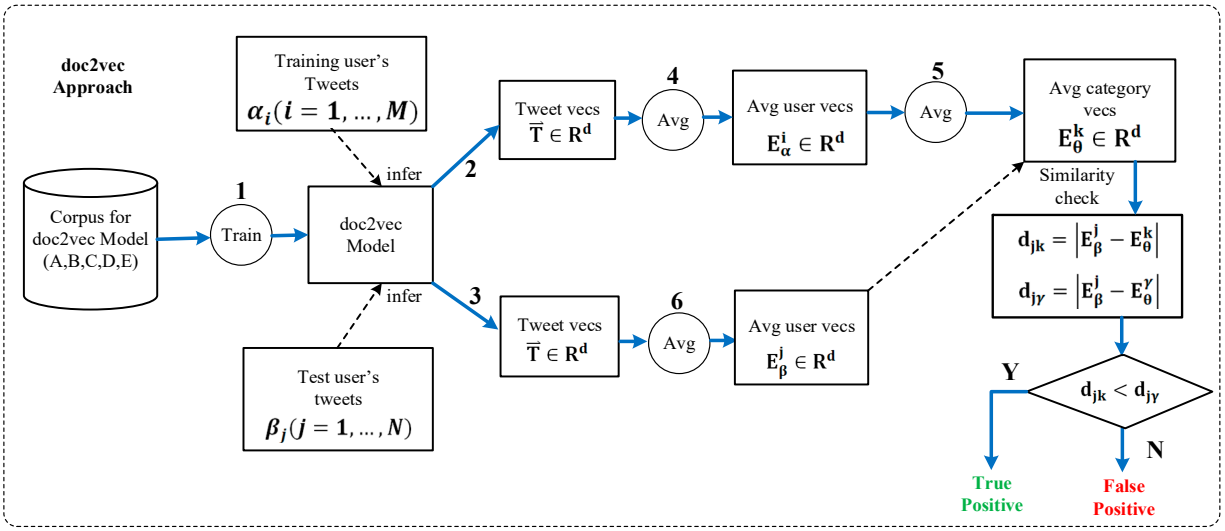


Figure 2. Overview of the doc2vec approach.

Twenty-eight experiments are carried out in total, by examining each of the seven cases in Table 5. over two parameters' two different values. These parameters are "min count" for values 1 and 5; "dm" for values 0 (distributed bag of words algorithm) and 1 (distributed memory algorithm).

4.2. Using Non-Textual Features in Doc2Vec (Enrichment)

Doc2vec is basically an extension of the word2vec word representation method. Document (paragraph) vectors are learned along with word embeddings. In this approach, user representations are extracted from a doc2vec model trained not only on user tweets but also with non-textual data. The tweet text is embodied with various information about the user (name, location, etc.) and the interactions of the tweet (retweet, like, comment). The default features of the doc2vec model are a tag and a list of tokens. The tokens of a tweet are captured as all words except stop words and words with non-alphanumeric characters in the tweet text. In this approach, the tag and token list pairs ("tweet_id", "tkn") of the documents are determined as the tweet ID and the tokens of the tweet. Non-textual features ("usr_name", "loc", "desc"; "Lusr", "Rusr", "Cusr") are added together with the standard features to enrich the learning capability of the model.

D and E datasets were created for the implementation of this approach. Table 6 illustrates an example enrichment operation on a tweet from Dataset A with Dataset E. The concatenation process of the features of a single tweet with an id of "tweet_id_1", composed by the user with the handle "usr_name_1", is explained in this table. The location information for this user is "loc_1", and profile description text is "dec_1". Tokens of the tweet are "tkn_1", . . . "tkn_5". Handles of the users who liked this tweet are "Lusr_1", . . . , "Lusr_6". Users who retweet this tweet are "Rusr_7", . . . , "Rusr_10" and who commented are "Cusr_11" and "Cusr_12".

The primary information about a tweet is the tweet text. Therefore, tokens of the tweet text are placed in the beginning of the Enriched_Data_1. Tokens are combined together for each tweet and they are treated as documents. A dummy token such as "nametag" is added between the tokens in the concatenation. The intuition here is that the model learns the probability of these tokens coexisting in documents. The tokens are not directly put together side by side. Dummy tokens ("loctag", "liketag", etc) are put between for each value of non-textual features in order to avoid order effects among them. They are added five times before and five times after the values because the window size parameter of the doc2vec algorithm is chosen as five. A special dummy token ("enrtag") is also added between the end of tweet tokens and the starting of the non-textual features. A

similar approach for padding "username" and "location" features with dummy tokens is introduced by [33]. Experiments of this approach are listed in Table 7.

Table 6. Enrichment with dummy tokens.

Tweet Data	Sample
Tweet Id	tweet_id_1
User name	usr_name_1
Location	loc_1
Description	desc_1
Tweet-List of tokens	tkn_1, tkn_2, tkn_3, tkn_4, tkn_5
List of likes	Lusr_1, Lusr_2, Lusr_3, Lusr_4, Lusr_5, Lusr_6
List of retweets	Rusr_7, Rusr_8, Rusr_9, Rusr_10
List of comments	Cusr_11, Cusr_12
	Enrichment
Enriched_Data_1	**tkn_1 tkn_2 tkn_3 tkn_4 tkn_5** enrtag enrtag enrtag enrtag enrtag nametag nametag nametag nametag nametag **usr_name_1** nametag nametag nametag nametag nametag loctag loctag loctag loctag loctag **loc_1** loctag loctag loctag loctag loctag desctag desctag desctag desctag desctag **desc_1** desctag desctag desctag desctag desctag liketag liketag liketag liketag liketag **Lusr_1** liketag liketag liketag liketag liketag **Lusr_2** liketag liketag liketag liketag liketag **Lusr_3** liketag liketag liketag liketag liketag **Lusr_4** liketag liketag liketag liketag liketag **Lusr_5** liketag liketag liketag liketag liketag **Lusr_6** liketag liketag liketag liketag liketag rttag rttag rttag rttag rttag **Rusr_7** rttag rttag rttag rttag rttag **Rusr_8** rttag rttag rttag rttag rttag **Rusr_9** rttag rttag rttag rttag rttag **Rusr_10** rttag rttag rttag rttag rttag commtag commtag commtag commtag commtag **Cusr_11** commtag commtag commtag commtag **Cusr_12**
Tagged Document	TaggedDocument ([***Enriched Data_1***], [***tweet_id_1***])

Table 7. Experimental setup #2.

Case No.	Dataset	Scenario
1	A, D	In the [33] study, user information are incorporated with tweets. In order to make a comparison with this study, datasets A and D are used.
2	A, E	Use of tweet, user information, and user activity data together.
3	A, B, E	Use of tweet, non-textual data, and a general-purpose corpus together.
4	A, C, E	
5	A, B, C, E	

4.3. Pre-Trained Word Embedding (PWE) Approaches

Word embeddings have undoubtedly been one of the most significant advances in the last decade, performing better in many NLP problems than traditional word vectors. This approach is based on using PWEs $w \in \mathbb{R}^d$ and obtaining tweet vectors $\vec{T} \in \mathbb{R}^d$ with the average of all words that exist in the tweet. This approach is the least complex and simplest among the proposed approaches in this study. As far as we know, such a document representation method has not been used in the user representation studies in the literature before. We think that a method based on a text representation technique for obtaining user representations should at least provide better results than this approach. Some studies on text representation usually provide results of simple approaches such as averaging word embeddings and TF-IDF for the sake of comparison as a baseline [33,39,40]. PWE experiments are presented for a similar objective. Also, an overall comparison of state-

of-the-art word vector techniques on the proposed method is presented as a contribution to the literature. Table 8 describes the main experimental setup for PWEs approaches. Dataset A is used in all of the experiments in this section.

Table 8. Experimental setup #3.

Case No.	PWE Model	Scenario
1	Glove [8], FastText [41], Google News [42], NumberBatch [14]	Comparison of publicly available PWEs.
2	Glove	Using the same PWE model trained on different corpus with different dimension sizes.
3	Word2Vec (our)	Training a word2vec model on our dataset. Use different parameters (dimension size, minimum count)
4	BERT, ELMO	Use of pre-trained contextualized word representation models
5	BERT, ELMO, Glove	Use of stacked embeddings as presented in [10]
6	TF-IDF	Using a sparse representation technique as another baseline. Investigate TF-IDF for a different number of features.

5. Experimental Results and Discussion

The results of the three different scenarios are presented separately with the same performance metrics. A task-based evaluation methodology is applied to determine the quality of the user vectors. Experiments are conducted for the scenarios (experimental setups) given in Table 5, Table 7, and Table 8 on 500 Twitter users belonging to five (S) groups. The users of each group are randomly split into %60 train and %40 test sets $(M = 300,\ N = 200)$. Evaluations are based on the test users, and their data (profile information, tweet, and activity data) are always excluded from the training phase. In some experiments, training users are only used for finding group embeddings and their twitter data are also excluded from the training of the embedding model.

The description of the computing environment and details on the time times of the experiments are presented in Appendix B.

5.1. Doc2Vec and Doc2Vec Enrichment Experiments

The results of the 28 experimental cases described for the doc2vec approach in Section 4.1 and the 20 experimental cases described for its enrichment approach in Section 4.2 are presented in Tables 9 and 10, respectively. The effects of the datasets and training parameters are evaluated in detail. The "Results" columns show the overall accuracy of each experiment for different doc2vec model parameters.

Table 9. Results of Experimental setup #1.

Case No (See Table 5)	Training Dataset	Results			
		PV-DM (dm = 1)		PV-DBOW (dm = 0)	
		min Count = 1	min Count = 5	min Count = 1	min Count = 5
1	A	0.945	0.960	0.910	0.925
2	B	0.920	0.900	0.900	0.900
3	C	0.940	0.945	0.910	0.915
4	A + B	0.930	0.935	0.915	0.935
5	B + C	0.890	0.955	0.900	0.940
6	A + C	0.970	**0.975**	0.925	0.930
7	A + B + C	0.940	**0.975**	0.915	0.965

Table 10. Results of Experimental setup #2.

Case No (See Table 7)	Training Dataset	Results			
		PV-DM (dm = 1)		PV-DBOW (dm = 0)	
		min Count = 1	min Count = 5	min Count = 1	min Count = 5
1	A + D	0.925	0.910	0.880	0.895
2	A + E	0.940	**0.970**	0.930	0.860
3	A + B + E	0.920	0.950	0.930	0.960
4	A + C + E	0.940	0.960	0.905	0.945
5	A + B + C + E	0.935	0.950	0.890	0.940

In the experiments, the models are saved and their evaluation scores are calculated at the end of each iteration. The given results here are obtained from the most successful model of each experiment within 20 iterations. Table 11 presents the confusion matrices for the best results of these two approaches.

Table 11. Confusion matrices for experimental setups #1 and #2.

Exp. #1	G1	G2	G3	G4	G5	Exp. #2	G1	G2	G3	G4	G5
G1	38	0	1	0	1	G1	38	0	1	0	1
G2	1	38	1	0	0	G2	1	39	0	0	0
G3	0	0	39	0	1	G3	0	0	39	1	0
G4	0	0	0	40	0	G4	0	0	0	40	0
G5	0	0	0	0	40	G5	1	0	0	1	38

The results suggest that training the proposed model with a general-purpose corpus comprising a large variety of tweets (Dataset C) can achieve 94.5% accuracy on the twitter user representation task. Using a fine-grained tweet corpus (Dataset A), which is specifically built for the task, can achieve 96% accuracy. Enriching this dataset with user information and user activity data (Datasets A and E) increases the accuracy to 97.0%. However, using basic user information together with the tweets of the users (Datasets A and D) does not increase the accuracy. However, the best result (97.5% accuracy) is obtained with training the model with the combination of Datasets A and C.

It is demonstrated that the proposed models can slightly improve (1–3%) the user embedding quality by leveraging additional social media corpora and user activity data. The best parameters are found to be "dm = 1" and "min count = 5" for all of the cases except when the model is trained on Datasets B or D. These two cases (Experimental Setup # 1 Case-2 and Experimental Setup # 2 Case-1) can be ignored in parameter selection because they are proved to not be improving the models. This finding about the superiority of pvdm over dbow contradicts the results in [43].

In addition to overall accuracy, precision (P), recall (R), and F1 scores are measured in Tables A2 and A3 (see Appendix C). The maximum F1 scores for each class are highlighted in both tables. Looking at all doc2vec experiments, we see that the best F1 score for no class is achieved when the "min count" parameter is set to 1 and "dm" is set to 0. Moreover, in Table A3, no model achieves the best F1 score when "min count" is 1 even when "dm" is 1. It is also important to note that the DBOW algorithm (dm = 0) is much faster but has lower performance.

The embeddings of the test users, which are extracted by the best model, are reduced to two-dimensional space using t-SNE [44] and plotted in Figure 3. The handles of randomly selected four users for each group are shown in the plot. The larger dots show the average group embeddings of the training users. The dots representing the users are colored according to the groups they belong to. Considering that our main dataset (A) consists of tweets from 2018 and earlier, when examining the actual Twitter profiles of the users, it can be observed that the users positioned close to each other have noticeable similarities in social media and real life. For example, users belonging to politics group (colored orange) are well separated from the other groups. In this group, Donald Trump Jr.

("donaldjtrumpjr") and Mike Pence ("mike_pence"), who are relevant for the Republican Party (United States of America), are close to each other and are relatively far from Boris Johnson and Sadiq Khan, who are particularly relevant for the United Kingdom politics. The positioning of these four Twitter users, who are at the common point of politics, in these proximities in a 2D drawing, demonstrates the success of the proposed method. Similarly, two users of the fashion-themed group, "khloekardashian" and "KimKardashian", are located very close to each other and are known to be genuinely interested in fashion and lifestyle in real life, and are also siblings to each other.

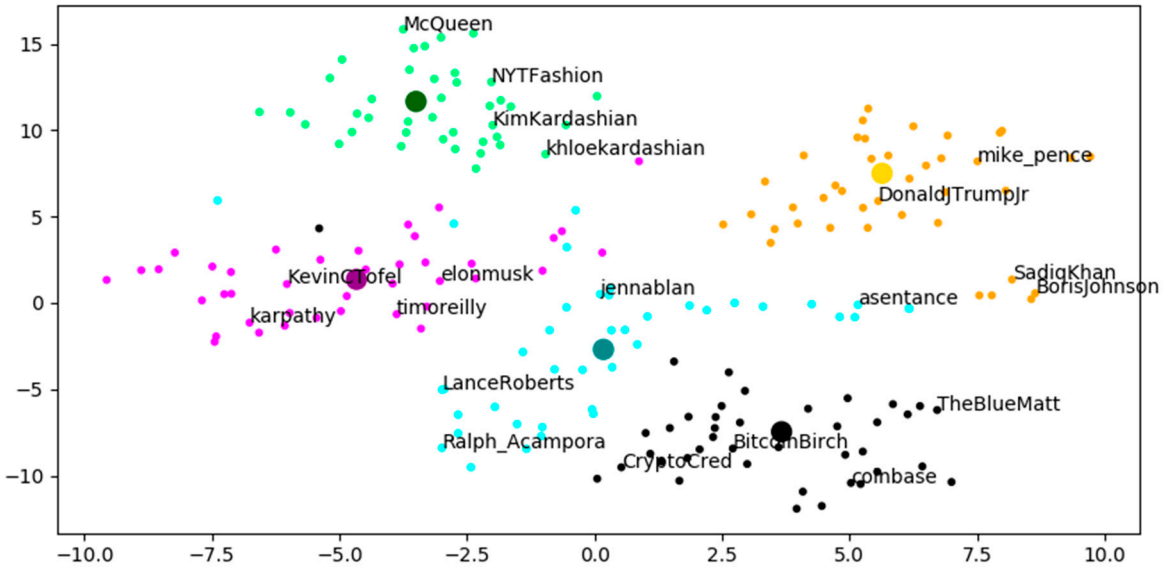


Figure 3. 2d view of the best model (accuracy 97.5%).

It is also worth noting that the evaluation process of the experiments is a very compute-intensive task. Assessing the model's ability to represent users requires obtaining 1K document representations for each users' record. Records consist of one or all of the following social network data in text format: tweet, user-info, and interaction data. Obtaining a vector representation from a trained Gensim doc2vec model is done using the infer_vector() function. Calling this function 500K times takes 4–42 h depending on the following conditions: size of the training corpus, size of the records, and the specifications of the model. This process takes up the longest time when the training corpus is the largest, and the records of the users consist of all the social network data (tweets, user-info, and interaction data). See Appendix B for a list of running times among different experiments.

5.2. Comparison of Different PWE Models

Table 12 compares the use of publicly available PWE models in the user representation task introduced in Section 4.3. In addition, experimental results for the commonly favored Glove model are listed separately in Table 13 (see Table 8 Cases 1–2). Glove embeddings are available in various vector formats in terms of their vector sizes and the corpora on which they were trained. Table 13 shows how these properties affect the performance of the user embeddings. The F1 scores are the weighted average of the F1 scores of all groups.

Table 12. Comparison of the PWE models.

PWE MODEL	Dimension	Accuracy	F1
Fast Text	300	0.810	**0.809**
Number Batch	300	0.810	0.807
Google News	300	0.720	0.713

Table 13. Glove hyper-parameter search.

Corpus Source	Vocab Size	Number of Tokens	Dimension	Accuracy	F1
Wikipedia	400×10^3	6×10^9	100	0.835	0.828
			200	0.815	0.808
			300	0.800	0.797
Twitter	1200×10^3	27×10^9	25	0.800	0.799
			50	0.805	0.799
			100	0.835	0.830
			200	0.840	**0.834**

The performances of the Word2Vec models that were trained on Dataset A from scratch (Table 8 Case-3) with different min count parameters are presented in Table 14. The dimension size is chosen to be 100 for these models. The best word2vec model gives the better accuracy compared with results in Tables 12 and 13.

Table 14. Word2vec experiments.

Min Count	Acc	F1
1	0.790	0.786
3	0.830	0.827
5	0.845	**0.842**

Table 15 shows the performance of the contextualized word embeddings BERT and ELMO. A document pool embedding is created by stacking BERT, ELMO, and Glove embeddings using the Flair NLP framework (see Table 8 Cases 4–5).

Table 15. Contextualized embedding experiments.

Embedding Model	Dimension	Accuracy	F1
Bert	3072	0.815	**0.814**
Elmo	3072	0.770	0.765
Bert-Elmo-Glove (Stacked)	3072 + 3072 + 100	0.815	0.812

Figure 4 depicts the F1 scores obtained in TF-IDF-based user representation experiments (see Table 8 Case-6). The number of maximum features is the size of the vocabulary, which also determines the size of the user embeddings. Thirty experiments are conducted with five different training datasets for six different numbers of features. It can be seen that datasets A and C perform better than the other datasets. The detailed accuracy and F1 scores for these experiments are reported in Table A4 (see Appendix C).

Glove, FastText, Google News, and NumberBatch are publicly available, ready-to-use PWEs. They are trained on different datasets with different algorithms and hyperparameters. Therefore, it is difficult to compare their performance on this task under exactly the same conditions. However, the comparison of the different Glove embeddings shows that dimension size, dataset, and number of tokens have some effect on the accuracy of the user embeddings. It is also shown that GoogleNews embeddings do not perform well for this task. FastText and NumberBatch and Glove give similar accuracy results. The pre-trained Bert model also performs as well these models but it does not outperform them. Considering the high computation and time cost caused by the big dimension sizes of Bert and Elmo, these models were not proved as effective alternatives to traditional PWEs without a fine-tuning process.

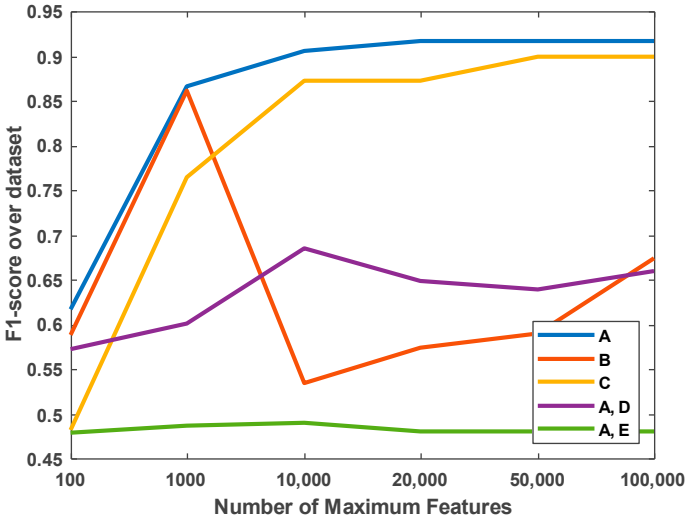


Figure 4. TF-IDF Experiments.

6. Conclusions and Future Work

In this work, we focus on learning real valued vectors for social media users by leveraging the state-of-the-art text representation methods. Important insights were gained that will be useful in developing of real-world applications that require a user representation model. The experiments show how parameters such as the representation method, hyper-parameter selection, and the type of dataset should be adjusted to achieve the best results under certain conditions. We create a one-of-its-kind dataset by harvesting social media data of manually selected 500 Twitter users to demonstrate the performance of the proposed approaches. The extensive experimental results serve as a guide for researchers from industry and academia who are interested in accommodating both textual and non-textual features in user representation tasks. A detailed comparison of the approaches covers a wide range of text representation methods including traditional methods (TFIDF), classical word embeddings (word2vec, Glove, FastText, etc.), doc2vec algorithms (PV-DBOW, PV-DM), and contextualized embeddings (Elmo, Bert). Some important findings of this study that are worth mentioning are:

1. In regards to proposed doc2vec models: A fine-grained tweet corpus like Dataset-A and its enriched versions Datasets D and E are unlikely to be present for most of the cases. Therefore, we recommend using datasets similar to Dataset C. The performance of a model trained on Dataset C can be increased with adding more tweets from users of a specific group if there is an intended focus group. Our doc2vec model can give 97.5% overall accuracy. PV-DM algorithm is used with min count = 5 for the best model.
2. We envision that the attempts for incorporating a wide range of social media activity data such as comment, like, retweet, mention, follow, etc. in user representation learning are at an early stage and there are still many challenges to be solved. Our model that is trained on the combination of dataset A and E can give 97.0% accuracy.
3. Although the publicly available word vectors are very simple to use and can give acceptable accuracies, they fail to perform as well as the doc2vec approaches. In the case of using a PWE, one should consider the corpus it was trained on and the length of the vectors. Our word2vec model trained on Dataset A can give 84.5% accuracy. The accuracy of the Glove model trained on Tweet corpus can reach to 84.0%.
4. Bert and Elmo do not outperform the PWE models. We believe that they would perform better than the given results if they are fine-tuned or trained from scratch. We plan to conduct broader experiments using contextualized embedding methods.

We also note that the following cases can also be investigated within the scope of using PWEs. However, they are not included in this study.

- Download a publicly available PWE and fine-tune it for your specific task.
- Train a language model from scratch using the algorithm of PWE (Glove, FastText, Google News, or NumberBatch).
- A broad investigation of using Contextualized word embeddings in user representation models. In this study, only their pre-trained multilingual versions are used.

The proposed approach has the potential to contribute to many applications within the scope of social media prediction and recommendation. Examples of these applications include the following possible scenarios:

- User similarity: Users who are most similar to a user in a network can be queried through the similarities of their embeddings. With such a system, unknown users can be defined over other similar users automatically without the need for applying manual analyses over their social media data. This approach can be useful in many social media analysis tasks such as trend analysis and reputation analysis.
- Social media sentiment analysis: In classical sentiment analysis, models learn from previously expressed comments with the aim of determining user opinions without considering who their authors are. We predict that sentiment analysis methods will continue to improve with the new techniques that incorporate user information as well as the actual text.
- Fake news detection: Early detection of whether a piece of information spreading on social media is true or fake is an important need in today's online world. Similar to the sentiment analysis example, a system can be designed where a social media post is evaluated not only by its content, but also by the representations of the users who create and support it.

Author Contributions: Conceptualization, I.R.H., B.A. and G.A.; methodology, I.R.H.; software, I.R.H.; validation, I.R.H., B.A. and G.A.; formal analysis, B.A.; resources, I.R.H., B.A. and G.A.; data curation, I.R.H.; writing—original draft preparation, I.R.H.; writing—review and editing, I.R.H. and B.A.; supervision, G.A.; project administration, G.A.; funding acquisition, G.A. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the Turkish Presidency of Defense Industries (SSB) under the project "Deep Learning and Big Data Analysis Platform (DEĞİRMEN)".

Data Availability Statement: The data presented in this study are available at https://github.com/irhallac/TwitterUserDataset (accessed on 10 June 2021).

Acknowledgments: This study was carried out as part of the PhD thesis research of the first author at Firat University, Turkey.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Gensim Framework

Gensim is a framework that offers capabilities for topic modeling and document similarity via an open source Python library. It also provides implementations of word2vec and doc2vec algorithms. In this work, Gensim is used for training the word-embedding and document-embedding models. The following notes and descriptions regarding the framework may be useful for understanding the technicalities of the experiments.

gensim.models.Doc2Vec: Document/paragraph vector model (gensim.models.Word2Vec for word2vec). Some important parameters of the model are as follows: Doc2Vec(vector_size, min_count, dm, epochs=, callbacks, workers=workers)

min count: Minimum number of occurrences for words to be included in the vocabulary. Setting this parameter to 1 (min count = 1) means that the vocabulary will contain all the words that exist in the corpus.

dm: This parameter specifies the doc2vec algorithm to be used in the training: dm = 0 means using the distributed bag of words algorithm (dbow); dm = 1 means using the distributed memory algorithm (pvdm).

callbacks: This allows running routines when epochs start or end. We used this option to evaluate the models at the end of epochs.

workers: This parameter specifies the number of worker threads.

gensim.models.doc2vec.TaggedDocument: An input document in Gensim is called a TaggedDocument. A TaggedDocument object contains words of the document as a list alongside with a tag representing it.

model.docvecs: This object is used to access the vector representation of documents. For example, if the tag of an input document is "tag1", then its representation is returned as follows: model.docvecs['tag1']. This method is never used in this work since test documents are not present in the training set.

Doc2Vec.infer_vector: Infers document representation from a trained model. In this work, this method is used for getting the representations of test documents as well as the training documents.

Appendix B. Computing Platform and Its Performance

Four machines are used for running the experiments. Each machine has 48-core Intel(R) Xeon(R) CPU E5 and 256GB RAM. Two of the machines have E5-2650 processors and the other two have E5-628L. This difference is ignored in the time measurements. Machines are selected randomly depending on their availability. The purpose of presenting the running times of the experiments is to give an idea about how the dataset size and model parameters effect the computing performance.

Gensim framework can train the doc2vec models in parallel with threads. This feature is provided with the "workers" parameter passed to the model on creation. The processors have 48 cores each so they can run 48 threads. Thirty-six is used as the number of workers in the trainings. However, this parallelism decreases the running time of each iteration (iteration-time) during the training of the model; it does not decrease much the total execution time of the experiments. This is because inferring vectors from the trained models constitutes the biggest part (95–99%) of the running time of the experiments and Gensim does not provide a native multithreading mechanism for the infer_vector() function.

Table A1 shows an overall comparison of the experiments in terms of running times. A different dataset is used for the given experiments, but they share the same parameters (dm = 0, min_count = 1). Changing these parameters results in a slight time differences in the iteration-time. The same experiment with dm parameter as 1 instead of 0 increases the iteration-time. In terms of min_count parameter, increasing it decreases the iteration-time. Since it was not possible to make a generalization in the same way for the inferring-time, the details are not given for the other parameters.

Table A1. Average running times.

Dataset	Iteration Running Time (Minutes)	Infer (Minutes)
A	0.5	256
B	10.4	303
C	1.5	282
A + B	11.16	321
A + C	2	286
B + C	12.93	322
A + B + C	13.46	330
A + E	2.33	2152
A + B + E	11.18	2521
A + C + E	5.11	2189
A + B + C + E	13.8	2534

Iteration running time: average running time of one iteration during the training process; Infer: document inferring time for 500K records.

Running an experiment on the dataset "A + B + C + E" for 200 iterations takes up to a year in the given hardware environment (1.77 day per iteration). For this reason, the results

are gathered for 20 iterations. Because it is not feasible to infer 500 documents at each iteration for a longer experiment, different approaches are applied for gathering results from longer iteration numbers. The first approach is to train the model for 200 iterations but evaluate it only every 10 iterations. The second approach is to train the model for 2000 iterations with evaluating the model every 100 iterations. It was observed that training the models for longer iterations does not increase the accuracy of the models. The best versions of the models are achieved within 20 iterations.

Appendix C. Precision, Recall, and F1 Scores of Doc2vec Experiments

Precision (P), Recall (R), and F1 scores are given for each semantic group (G) in Tables A2 and A3. In the first columns, "C" stands for "case number of the respective experimental setups".

Table A2. Detailed scores of Experimental setup #1.

C	G	Results											
		PV-DM (dm = 1)						PV-DBOW (dm = 0)					
		min Count = 1			min Count = 5			min Count = 1			min Count = 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	G1	0.923	0.900	0.911	0.973	0.900	0.935	0.971	0.825	0.892	0.968	0.750	0.845
	G2	1.000	0.975	0.987	1.000	0.975	0.987	0.974	0.925	0.949	0.929	0.975	0.951
	G3	0.946	0.875	0.909	0.949	0.925	0.937	0.971	0.825	0.892	1.000	0.900	0.947
	G4	0.889	1.000	0.941	0.909	1.000	0.952	0.741	1.000	0.851	0.800	1.000	0.889
	G5	0.975	0.975	0.975	0.976	1.000	0.988	0.975	0.975	0.975	0.976	1.000	0.988
2	G1	0.892	0.825	0.857	0.938	0.750	0.833	0.919	0.850	0.883	0.923	0.900	0.911
	G2	0.865	0.800	0.831	0.773	0.850	0.810	0.946	0.875	0.909	0.972	0.875	0.921
	G3	0.886	0.975	0.929	0.864	0.950	0.905	0.907	0.975	0.940	0.929	0.975	0.951
	G4	0.976	1.000	0.988	0.974	0.950	0.962	0.952	1.000	0.976	0.952	1.000	0.976
	G5	0.976	1.000	0.988	0.976	1.000	0.988	0.951	0.975	0.963	0.951	0.975	0.963
3	G1	1.000	0.800	0.889	1.000	0.850	0.919	0.971	0.825	0.892	0.970	0.800	0.877
	G2	0.950	0.950	0.950	0.949	0.925	0.937	0.971	0.850	0.907	0.923	0.900	0.911
	G3	0.905	0.950	0.927	0.884	0.950	0.916	0.809	0.950	0.874	0.844	0.950	0.894
	G4	0.952	1.000	0.976	0.952	1.000	0.976	0.950	0.950	0.950	0.974	0.950	0.962
	G5	0.909	1.000	0.952	0.952	1.000	0.976	0.886	0.975	0.929	0.886	0.975	0.929
4	G1	0.972	0.875	0.921	0.971	0.850	0.907	0.917	0.825	0.868	0.969	0.775	0.861
	G2	1.000	0.875	0.933	0.974	0.925	0.949	0.872	0.850	0.861	0.826	0.950	0.884
	G3	0.780	0.975	0.867	0.800	1.000	0.889	0.884	0.950	0.916	0.950	0.950	0.950
	G4	1.000	0.925	0.961	1.000	0.925	0.961	0.974	0.950	0.962	0.952	1.000	0.976
	G5	0.952	1.000	0.976	0.975	0.975	0.975	0.930	1.000	0.964	1.000	1.000	**1.000**
5	G1	0.906	0.725	0.806	0.972	0.875	0.921	0.889	0.800	0.842	0.971	0.825	0.892
	G2	0.821	0.800	0.810	0.949	0.925	0.937	0.846	0.825	0.835	0.923	0.900	0.911
	G3	0.822	0.925	0.871	0.929	0.975	0.951	0.881	0.925	0.902	0.907	0.975	0.940
	G4	0.930	1.000	0.964	1.000	1.000	**1.000**	0.952	1.000	0.976	0.952	1.000	0.976
	G5	0.976	1.000	0.988	0.930	1.000	0.964	0.927	0.950	0.938	0.952	1.000	0.976
6	G1	0.974	0.950	**0.962**	0.974	0.950	**0.962**	0.970	0.800	0.877	0.971	0.850	0.907
	G2	1.000	0.925	0.961	1.000	0.950	0.974	0.974	0.950	0.962	0.927	0.950	0.938
	G3	0.929	0.975	0.951	0.951	0.975	**0.963**	0.816	1.000	0.899	0.864	0.950	0.905
	G4	0.976	1.000	0.988	1.000	1.000	**1.000**	0.974	0.925	0.949	0.974	0.925	0.949
	G5	0.976	1.000	0.988	0.952	1.000	0.976	0.927	0.950	0.938	0.929	0.975	0.951
7	G1	0.972	0.875	0.921	0.974	0.950	**0.962**	0.914	0.800	0.853	0.973	0.900	0.935
	G2	1.000	0.925	0.961	1.000	0.950	0.974	0.872	0.850	0.861	0.974	0.950	0.962
	G3	0.830	0.975	0.897	0.951	0.975	**0.963**	0.870	1.000	0.930	0.951	0.975	**0.963**
	G4	1.000	0.925	0.961	1.000	1.000	**1.000**	0.974	0.950	0.962	0.976	1.000	0.988
	G5	0.930	1.000	0.964	0.952	1.000	0.976	0.951	0.975	0.963	0.952	1.000	0.976

Table A3. Detailed scores of Experimental setup #2.

C	G	Results											
		PV-DM (dm = 1)						PV-DBOW (dm = 0)					
		min Count = 1			min Count = 5			min Count = 1			min Count = 5		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
1	G1	0.902	0.925	0.914	0.895	0.850	0.872	1.000	0.625	0.769	0.900	0.675	0.771
	G2	0.974	0.950	0.962	0.974	0.925	0.949	1.000	0.900	0.947	0.884	0.950	0.916
	G3	0.939	0.775	0.849	0.939	0.775	0.849	0.841	0.925	0.881	0.854	0.875	0.864
	G4	0.909	1.000	0.952	0.833	1.000	0.909	0.780	0.975	0.867	0.870	1.000	0.930
	G5	0.907	0.975	0.940	0.930	1.000	0.964	0.867	0.975	0.918	0.975	0.975	0.975
2	G1	0.884	0.950	0.916	0.950	0.950	0.950	0.897	0.875	0.886	0.750	0.750	0.750
	G2	1.000	0.950	0.974	1.000	0.975	0.987	1.000	0.950	0.974	1.000	0.950	0.974
	G3	0.974	0.925	0.949	0.975	0.975	0.975	0.826	0.950	0.884	0.776	0.950	0.854
	G4	0.889	1.000	0.941	0.952	1.000	0.976	0.974	0.950	0.962	0.860	0.925	0.892
	G5	0.972	0.875	0.921	0.974	0.950	0.962	0.974	0.925	0.949	0.967	0.725	0.829
3	G1	0.917	0.825	0.868	0.946	0.875	0.909	0.925	0.925	0.925	0.907	0.975	0.940
	G2	1.000	0.950	0.974	1.000	0.950	0.974	0.974	0.950	0.962	1.000	0.975	0.987
	G3	0.947	0.900	0.923	0.974	0.925	0.949	0.900	0.900	0.900	1.000	0.925	0.961
	G4	0.800	1.000	0.889	0.870	1.000	0.930	0.923	0.900	0.911	0.951	0.975	0.963
	G5	0.974	0.925	0.949	0.976	1.000	**0.988**	0.929	0.975	0.951	0.950	0.950	0.950
4	G1	0.927	0.950	0.938	0.951	0.975	**0.963**	0.895	0.850	0.872	0.884	0.950	0.916
	G2	1.000	0.950	0.974	1.000	0.950	0.974	0.975	0.975	0.975	1.000	0.975	**0.987**
	G3	0.974	0.950	0.962	0.973	0.900	0.935	0.800	0.900	0.847	0.927	0.950	0.938
	G4	0.851	1.000	0.920	0.909	1.000	0.952	0.881	0.925	0.902	0.950	0.950	0.950
	G5	0.971	0.850	0.907	0.975	0.975	0.975	1.000	0.875	0.933	0.973	0.900	0.935
5	G1	0.881	0.925	0.902	0.921	0.875	0.897	0.837	0.900	0.867	0.902	0.925	0.914
	G2	1.000	0.950	0.974	1.000	0.950	0.974	0.974	0.925	0.949	1.000	0.975	**0.987**
	G3	0.974	0.925	0.949	0.974	0.950	0.962	0.875	0.875	0.875	0.950	0.950	0.950
	G4	0.870	1.000	0.930	0.889	1.000	0.941	0.895	0.850	0.872	0.902	0.925	0.914
	G5	0.972	0.875	0.921	0.975	0.975	0.975	0.878	0.900	0.889	0.949	0.925	0.937

Table A4. TF-IDF results.

Number of Maximum Features	Accuracy (Acc) and F1 Scores Over Datasets									
	A		B		C		A, D		A, E	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
100	0.615	0.616	0.580	0.588	0.495	0.482	0.595	0.572	0.490	0.478
1000	0.865	0.866	0.830	0.861	0.760	0.764	0.615	0.601	0.500	0.487
10,000	0.905	0.906	0.540	0.534	0.875	0.873	0.695	**0.685**	0.500	**0.489**
20,000	0.915	**0.916**	0.575	0.574	0.875	0.873	0.660	0.648	0.490	0.481
50,000	0.915	**0.916**	0.590	0.590	0.900	**0.900**	0.645	0.639	0.490	0.481
100,000	0.915	**0.916**	0.675	**0.674**	0.900	**0.900**	0.665	0.659	0.490	0.481

References

1. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
2. Koehl, D.; Davis, C.; Nair, U.; Ramachandran, R. Analogy-based Assessment of Domain-specific Word Embeddings. In Proceedings of the 2020 SoutheastCon, Raleigh, NC, USA, 28–29 March 2020; pp. 1–6. [CrossRef]
3. Yang, H.; Sohn, E. Expanding Our Understanding of COVID-19 from Biomedical Literature Using Word Embedding. *Int. J. Environ. Res. Public Health* **2021**, *18*, 3005. [CrossRef] [PubMed]

4. Zhao, J.; van Harmelen, F.; Tang, J.; Han, X.; Wang, Q.; Li, X. *Knowledge Graph and Semantic Computing. Knowledge Computing and Language Understanding: Third China Conference, CCKS 2018, Tianjin, China, August 14–17, 2018, Revised Selected Papers*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 957.
5. Akbik, A.; Blythe, D.; Vollgraf, R. Contextual string embeddings for sequence labeling. In Proceedings of the 27th İnternational Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018.
6. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
7. Firth, J.R. A synopsis of linguistic theory, 1930–1955. In *Studies in Linguistic Analysis*; Longmans: London, UK, 1957.
8. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543.
9. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.
10. Akbik, A.; Bergmann, T.; Blythe, D.; Rasul, K.; Schweter, S.; Vollgraf, R. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, Minneapolis, MN, USA, 2–7 June 2019. [CrossRef]
11. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
12. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep Contextualized Word Representations. *arXiv* **2018**, arXiv:1802.05365.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
14. Speer, R.; Chin, J.; Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In Proceedings of the AAAI Conference on Artificial Intelligence, Palo Alto, CA, USA, 4–9 February 2017.
15. Speer, R.; Lowry-Duda, J. ConceptNet at SemEval-2017 Task 2: Extending Word Embeddings with Multilingual Relational Knowledge. *arXiv* **2018**, arXiv:1704.03560.
16. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014.
17. Hallac, I.R.; Makinist, S.; Ay, B.; Aydin, G. user2Vec: Social Media User Representation Based on Distributed Document Embeddings. In Proceedings of the 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, 21–22 September 2019; pp. 1–5.
18. Carrasco, S.S.; Rosillo, R.C. Word Embeddings, Cosine Similarity and Deep Learning for Identification of Professions & Occupations in Health-related Social Media. In Proceedings of the Sixth Social Media Mining for Health (# SMM4H) Workshop and Shared Task, Mexico City, Mexico, 10 June 2021; pp. 74–76.
19. Samad, M.D.; Khounviengxay, N.D.; Witherow, M.A. Effect of Text Processing Steps on Twitter Sentiment Classification using Word Embedding. *arXiv* **2020**, arXiv:2007.13027.
20. Gallo, F.R.; Simari, G.I.; Martinez, M.V.; Falappa, M.A. Predicting user reactions to Twitter feed content based on personality type and social cues. *Future Gener. Comput. Syst.* **2020**, *110*, 918–930. [CrossRef]
21. Liao, C.H.; Chen, L.X.; Yang, J.C.; Yuan, S.M. A photo post recommendation system based on topic model for improving facebook fan page engagement. *Symmetry* **2020**, *12*, 1105. [CrossRef]
22. Carta, S.; Podda, A.S.; Recupero, D.R.; Saia, R.; Usai, G. Popularity prediction of instagram posts. *Information* **2020**, *11*, 453. [CrossRef]
23. Chen, H.-H. Behavior2Vec: Generating distributed representations of users' behaviors on products for recommender systems. *ACM Trans. Knowl. Discov. Data (TKDD)* **2018**, *12*, 1–20. [CrossRef]
24. Mehrotra, R.; Yilmaz, E. Task embeddings: Learning query embeddings using task context. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 2199–2202.
25. Gupta, U.; Wu, C.J.; Wang, X.; Naumov, M.; Reagen, B.; Brooks, D.; Cottel, B.; Hazelwood, K.; Hempstead, M.; Jia, B.; et al. The architectural implications of facebook's dnn-based personalized recommendation. In Proceedings of the 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, CA, USA, 22–26 February 2020.
26. Chen, L.; Qian, T.; Zhu, P.; You, Z. Learning user embedding representation for gender prediction. In Proceedings of the 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, CA, USA, 6–8 November 2016; pp. 263–269.
27. Lay, A.; Ferwerda, B. Predicting users' personality based on their 'liked'images on instagram. In Proceedings of the 23rd International on Intelligent User Interfaces, Tokyo, Japan, 7–11 March 2018.
28. Mairesse, F.; Walker, M.A.; Mehl, M.R.; Moore, R.K. Using linguistic cues for the automatic recognition of personality in conversation and text. *J. Artif. Intell. Res.* **2007**, *30*, 457–500. [CrossRef]
29. Rajaraman, A.; Ullman, J.D. *Mining of Massive Datasets*; Cambridge University Press: Cambridge, UK, 2011.
30. Adomavicius, G.; Sankaranarayanan, R.; Sen, S.; Tuzhilin, A. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.* **2005**, *23*, 103–145. [CrossRef]

31. Żołna, K.; Romański, B. User modeling using LSTM networks. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
32. Pan, S.; Ding, T. Social media-based user embedding: A literature review. *arXiv* **2019**, arXiv:1907.00725.
33. Xing, L.; Paul, M.J. Incorporating Metadata into Content-Based User Embeddings. In Proceedings of the 3rd Workshop Noisy User-Generated Text, Copenhagen, Denmark, 7 September 2017; pp. 45–49. Available online: http://aclweb.org/anthology/W17-4406 (accessed on 10 June 2021).
34. Littman, J.; Wrubel, L.; Kerchner, D.; Gaber, Y.B. News Outlet Tweet Ids. *Harv. Dataverse* **2017**. [CrossRef]
35. Binkley, P. Twarc-Report README. md. 2015. Available online: https://github.com/DocNow/twarc (accessed on 20 February 2021).
36. Jaccard, P. Nouvelles recherches sur la distribution florale. *Bull. Soc. Vaud. Sci. Nat.* **1908**, *44*, 223–270.
37. Vijaymeena, M.K.; Kavitha, K. A survey on similarity measures in text mining. *Mach. Learn. Appl. An Int. J.* **2016**, *3*, 19–28.
38. Hoff, P.D.; Raftery, A.E.; Handcock, M.S. Latent space approaches to social network analysis. *J. Am. Stat. Assoc.* **2002**. [CrossRef]
39. Dai, A.M.; Olah, C.; Le, Q.V. Document embedding with paragraph vectors. *arXiv* **2015**, arXiv:1507.07998.
40. Benton, A.; Dredze, M. Using Author Embeddings to Improve Tweet Stance Classification. In Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, Brussels, Belgium, 1 November 2018. [CrossRef]
41. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.
42. Google-News Pre-trained Vectors (GoogleNews-Vectors-Negative300.bin.gz). Available online: https://code.google.com/archive/p/word2vec/ (accessed on 1 June 2021).
43. Lau, J.H.; Baldwin, T. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv* **2016**, arXiv:1607.05368.
44. van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.