





## Article

# Topic-Based Document-Level Sentiment Analysis Using Contextual Cues

Ciprian-Octavian Truică <sup>1,\*</sup><sup>†</sup>, Elena-Simona Apostol <sup>1,\*</sup><sup>†</sup>, Maria-Luiza Șerban <sup>1,†</sup> and Adrian Paschke <sup>2</sup>

<sup>1</sup> Computer Science and Engineering Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, RO-060042 Bucharest, Romania; maria\_luiza.serban@upb.ro

<sup>2</sup> Fraunhofer Institute for Open Communication Systems, 10589 Berlin, Germany; adrian.paschke@fokus.fraunhofer.de

\* Correspondence: ciprian.truica@upb.ro (C.-O.T.); elena.apostol@upb.ro (E.-S.A.)

† These authors contributed equally to this work.

**Abstract:** Document-level Sentiment Analysis is a complex task that implies the analysis of large textual content that can incorporate multiple contradictory polarities at the phrase and word levels. Most of the current approaches either represent textual data using pre-trained word embeddings without considering the local context that can be extracted from the dataset, or they detect the overall topic polarity without considering both the local and global context. In this paper, we propose a novel document-topic embedding model, DOCTOPIC2VEC, for document-level polarity detection in large texts by employing general and specific contextual cues obtained through the use of document embeddings (DOC2VEC) and Topic Modeling. In our approach, (1) we use a large dataset with game reviews to create different word embeddings by applying WORD2VEC, FASTTEXT, and GLOVE, (2) we create DOC2VECs enriched with the local context given by the word embeddings for each review, (3) we construct topic embeddings TOPIC2VEC using three Topic Modeling algorithms, i.e., LDA, NMF, and LSI, to enhance the global context of the Sentiment Analysis task, (4) for each document and its dominant topic, we build the new DOCTOPIC2VEC by concatenating the DOC2VEC with the TOPIC2VEC created with the same word embedding. We also design six new Convolutional-based (Bidirectional) Recurrent Deep Neural Network Architectures that show promising results for this task. The proposed DOCTOPIC2VECs are used to benchmark multiple Machine and Deep Learning models, i.e., a Logistic Regression model, used as a baseline, and 18 Deep Neural Networks Architectures. The experimental results show that the new embedding and the new Deep Neural Network Architectures achieve better results than the baseline, i.e., Logistic Regression and DOC2VEC.



**Citation:** Truică, C.-O.; Apostol, E.-S.; Șerban, M.-L.; Paschke, A.

Topic-Based Document-Level Sentiment Analysis Using Contextual Cues. *Mathematics* **2021**, *9*, 2722. <https://doi.org/10.3390/math9212722>

Academic Editor: Stelios Papadakis

Received: 28 September 2021

Accepted: 24 October 2021

Published: 27 October 2021

**Keywords:** document-level Sentiment Analysis; document-topic embeddings; Topic Modeling; Deep Learning Architectures

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Opinion Mining and Sentiment Analysis are related research topics, at the intersection of Machine Learning and Natural Language Processing, that, recently, have been studied intensively [1–6]. The interest in these related topics is due to the wide range of applications where they can be used (e.g., advertising, politics, business, etc.) and the availability of large amounts of textual data. They are generally used to identify opinions and recognize the sentiments expressed, as well as the general polarity of a text, e.g., subjective or objective, positive or negative. The data sources that are mostly used in Opinion and Sentiment Analysis tasks are represented by blogs, posts from social media, comments from movie and product reviews sites or new articles [7]. These can be used to complete different tasks, such as emotion detection and sentiment classification.

Various types of neural networks have been employed to solve more accurately specific Opinion and Sentiment Analysis tasks, e.g., Recurrent Neural Networks (RNNs),

Convolutional Neural Networks (CNNs). RNNs have been proven to offer good results for text analysis tasks [8]. Different types of RNNs, such as GRU (Gated Recurrent Unit) or LSTM (Long Short Term Memory), were developed to overcome the flaws of other feed-forward PERCEPTRON-based neural networks. RNNs can capture information about the input, such as context dependency between words, and share parameters across epochs. CNNs [9,10] are a type of feed-forward networks very popular due to the minimal preprocessing requirement. These types of networks are regarded as more powerful than RNNs. Although CNNs are ideal for image processing and their accuracy is dependent on the initial parameter tuning, they turned out to also bring increased performance in text processing, especially combined with other neural networks.

The main motivation of this paper is to improve the accuracy of document-level Sentiment Analysis (Definition 1) using Deep Learning models which employ contextual cues (Definition 2). Thus, we aim to introduce specific/local (Definition 3) and general/global (Definition 4) contextual cues by employing word embeddings (WORDEMBS) and Topic Modeling in order to improve the accuracy of polarity detection. Thus, to improve the context of Sentiment Analysis, we enhance document embedding (DOC2VEC) using contextual cues through the use of different WORDEMBS, which adds local context by training the embeddings on the documents within a set of documents  $D$ , and Topic Modeling algorithms, which adds global context by extracting topics from the set of documents  $D$ . To add context, we employ DOC2VECS and topic embeddings (TOPIC2VECS) to create a new embedding, i.e., DOCTOPIC2VEC, as a concatenation between a DOC2VEC and a TOPIC2VEC.

**Definition 1** (Document-level Sentiment Analysis). *Document-level Sentiment Analysis is the task used to determine for a document  $d_i$  belonging to a set of documents  $D$  whether its text has a positive, neutral, or negative polarity.*

**Definition 2** (Contextual cues). *The contextual cues consist of the local and global lexical, semantic, and syntactic information of a word  $w_i$  given to a Machine/Deep Learning model to solve a task  $t$ . (Note: We use both local and global context for document-level Sentiment Analysis.)*

**Definition 3** (Local Context). *The local context refers to the local lexical, semantic, and syntactic information of a word  $w_i$  within a document  $d$ . (Note: We extract the local context by training word embeddings for each word  $w_i \in d$ . Thus, the embedding encodes the local context by preserving the word's lexical, semantic, and syntactic similarity as well as its relation with other words within the same document).*

**Definition 4** (Global Context). *The global context refers to the global lexical, semantic, and syntactic information of a word  $w_i$  within a set of documents  $D$ . (Note: We extract the global context by detecting the most important topic for each document  $d_i \in D$ . Thus, documents belonging to the same topic also belong to the same context and the context given by a topic is seen as a global context for the document belonging to this topic).*

A DOC2VEC is constructed as the average of the WORDEMBS for the terms in the document. This embedding manages to preserve the contexts and semantics of words at the document level [11]. WORDEMBS add semantic context by encoding the position for words in a sentence before vectorizing the text. We use five WORDEMB: (1) WORD2VEC CBOW (Continuous Bag-of-Words) model; (2) WORD2VEC SKIP-GRAM model; (3) FASTTEXT CBOW model; (4) FASTTEXT SKIP-GRAM model, and (5) GLOVE model. WORD2VEC captures the context of a word in a document and the relationship with the words surrounding it. Furthermore, this embedding manages to encode the semantic and syntactic similarity of the words within the document. WORD2VEC uses two models to determine the local context: CBOW and SKIP-GRAM. The CBOW model predicts the word's individual context by taking into account the context of all the words within the corpus. The SKIP-GRAM takes a word and determines the words that are in the same context. FASTTEXT extends

Word2Vec by learning embedding vectors for the n-grams that are found within each word. FASTTEXT also uses CBOW and SKIP-GRAM models. GLOVE enhances the local context information of words using global statistics, i.e., word co-occurrence.

We use Topic Modeling to extract the hidden latent semantic patterns and to add a general context to Sentiment Analysis by detecting and grouping document with similar characteristics by the subjects of interest. We employ different Topic Modeling algorithms, i.e., Latent Dirichlet allocation (LDA) [12], Non-Negative Matrix Factorization (NMF) [13], Latent Semantic Indexing (LSI) [14]. We encode these hidden patterns that add a general context to Sentiment Analysis into TOPIC2VECS. TOPIC2VECS are built as the average between the topics top-k terms' relevance and their WORDEMBs. By employing TOPIC2VECS, we manage to encode context-based document grouping and to enhance each document's context by constructing the DOCTOPIC2VEC using the dominant topic as a concatenation between each document's DOC2VEC and TOPIC2VEC. Thus, documents that are similar in meaning and context, including polarity and opinion, will be closer to each other in the vector space than texts which are not necessarily related.

For the experiments, we use a large dataset consisting of game reviews. We create the DOCTOPIC2VECS using the discussed WORDEMBs and Topic Modeling algorithms. Each DOCTOPIC2VEC embedding is used in classification tasks that apply Logistic Regression (LOGREG) and neural networks with LSTM, GRU, Bidirectional, DENSE, and CNN layers. We also design six news Convolutional-based (Bidirectional) Recurrent Deep Neural Network (CNN-(Bi)RNN) Architectures for the task of determining accurate document-level polarity. The results of our benchmark show that the accuracy is improved by about 5% when adding DOC2VEC contextual clues with NMF and LSI Topic Modeling algorithms, compared to the baseline, i.e., DOC2VEC-based LOGREG Sentiment Analysis. Furthermore, the proposed new architectures outperformed the state of the art solution proposed in [3].

The main research questions we want to answer are:

- (Q<sub>1</sub>) Does a Topic Modeling approach improve the overall accuracy of detecting the polarity of textual data?
- (Q<sub>2</sub>) Can local context added by Word Embeddings and global context added by Topic Modeling improve the accuracy of the Sentiment Analysis task?
- (Q<sub>3</sub>) Can a novel CNN-(Bi)RNN architecture prove to be a better model for the Sentiment Analysis task?

Thus, by answering these questions, the main objective of this work is three-fold:

- (O<sub>1</sub>) Analyze the impact of Topic Modeling on the Sentiment Analysis task;
- (O<sub>2</sub>) Construct a novel embedding DOCTOPIC2VEC that encapsulates both local and global context in order to improve the accuracy of detecting the polarity of textual data;
- (O<sub>3</sub>) Build a novel CNN-(Bi)RNN architecture to increase the accuracy of the Sentiment Analysis task.

This paper is structured as follows. In Section 2, we discuss the current advancement in Sentiment Analysis techniques. Section 3 presents the proposed architecture and describes each component module, together with the used algorithms and techniques. In Section 4, we describe the dataset and our set of experiments. Finally, we analyze and interpret the results. Section 5 is drawing the final conclusions and provides several future directions.

## 2. Related Work

Sentiment Analysis approaches can be classified into three categories: Machine Learning, Lexicon-based, and Hybrid [15]. Furthermore, these techniques are divided, based on the granularity level, in word (or aspect), sentence (or short text), and document (or long text) level.

There are not many solutions focusing on context-based Sentiment Analysis models. A context enrichment model for Sentiment Analysis is proposed in [4]. The authors add several processing steps, prior to sentiment classification, in order to augment the dataset with context. One important step discussed here is the prior-polarity identifica-

tion with SentiWordNet. Unfortunately, the authors do not clearly specify what are the advantages of prior-polarity identification, and their model is just conceptual without any real experiments.

Most of the related previous works primarily use either only embeddings as text representation that are incorporated into the Sentiment Analysis model (e.g., [2,3]) or they consider Topic Modeling for determining the opinion by topic, and not to add context to the model (e.g., [16,17]).

In [3], the authors propose a Deep Learning 4CNN-BiLSTM model for document-level Sentiment Analysis. Their model consists of four CNN layers and one BiLSTM layer. For the experiments, they use a relatively small amount of documents, i.e., 2003 articles from French newspapers. They employ two optimizers, *SGD* and *Adam*, and WORD2VEC as WORDEMBs solution. The proposed model is compared with CNN, LSTM, BiLSTM, and CNN-LSTM, and they conclude that it achieves the best accuracy. Although they obtained a high accuracy for the 4CNN-BiLSTM model, the results are not conclusive, as the experiments are performed on a small dataset. In our experiments, we also analyze their model, both the version proposed by them and also by adding Topic Modeling.

Attention mechanisms condition the Sentiment Analysis model to pay attention to the features which contribute the most to the task. The authors of the paper [18] propose a model based on LSTM layers with an attention mechanism. They used different approaches for the attention mechanism, i.e., convolution-based and pooling-based attention mechanism, and the word-vectors used for training, i.e., pre-trained word vectors from Word2Vec and randomly initialized word-vectors. Their model obtained better results than baseline methods on two out of three datasets. Attention-based Bidirectional CNN-RNN Deep Model (ABCDM) [19], another attention-based solution, use independent BiLSTM and GRU layers to extract both past and future contexts and an attention mechanism to put more or less emphasis on different words. To reduce the dimensionality and create new feature representations, the ABCDM model utilizes both convolutional layers and pooling techniques. This model achieves state-of-the-art performance when compared with other Neural Network architectures for the task of Sentiment Analysis on reviews and Twitter datasets.

An improved method for generating WORDEMBs used in Sentiment analysis is proposed in [2]. This method, Improved Word Vectors, uses Part-of-Speech, lexicon-based, and word position techniques together with WORD2VEC or GLOVE models. The performance of the proposed solution is tested using four different Deep Learning models and benchmark sentiment datasets. The results show that when using these embeddings, the accuracy of the model is slightly increased.

One solution that uses Topic Modeling for sentiment detection is presented in [17]. The authors combine shrinkage regression and Topic Modeling for detecting polarity in a Twitter dataset. The proposed model consists of two stages. In the first stage, they detect the polarity of the tweets using two shrinkage regression models. This type of regression adds a penalty in the way the loss function is calculated for models that have too many variables. During the second stage, the relevant topics are identified using LDA. The model estimates the sentiment of each topic using term sentiment scores.

Topic Modeling and WORDEMBs have been used together to analyze the sentiment of topics. However they have never been applied in Sentiment Analysis at the document level, as we propose in this paper. This approach is used for aspect-based topic Sentiment Analysis [20,21]. In this case, Topic Modeling is used for aspect extraction and categorization without considering the global context. In [21], the authors combine domain-trained WORDEMB and Topic Modeling for categorizing aspect-terms from online reviews. Their proposed model uses continuous WORDEMB and LDA algorithm. The model is tested using a small dataset, i.e., the restaurant reviews from the SemEval-2014 dataset consisting of 3841 sentences. One important limitation of their model is that it has a longer convergence time than the standard model and has lower performance than supervised models.

Several recent works also explore pre-trained language models for the Sentiment Analysis task, e.g., BERT [22], RoBERTa [23], ALBERT [24]. In [25], BERT is compared with an LSTM-based architecture and achieves an overall better f-measure. In [26], a RoBERTa Sentiment Analysis model is combined with key entity detection, based on the presumption that people are more prone to observe negative information. This approach improves the accuracy of the Sentiment Analysis task when compared with architectures consisting of BERT or RoBERTa transformers combined with SVM, LR, or NBM.  $DICE_T$  [1] is another transformer-based method for sentiment analysis. The novelty of  $DICE_T$  is that it enhances the data quality by handling noises within contexts. For this, it uses six types of embeddings, i.e., character embeddings, GloVe, Part-of-Speech embeddings, Lexicon embeddings, ELMo [27] and BERT-based embeddings. The concatenated embeddings are fed to a BiLSTM network with attention.  $DICE_T$  has higher performance compared with Sentiment Analysis methods that use the standard one-type of embeddings, e.g., Glove or Word2Vec, or other pre-processing methods, e.g., TFIDF.

### 3. Methodology

Figure 1 presents the proposed architecture for our topic-based Sentiment Analysis using a contextual cues model.

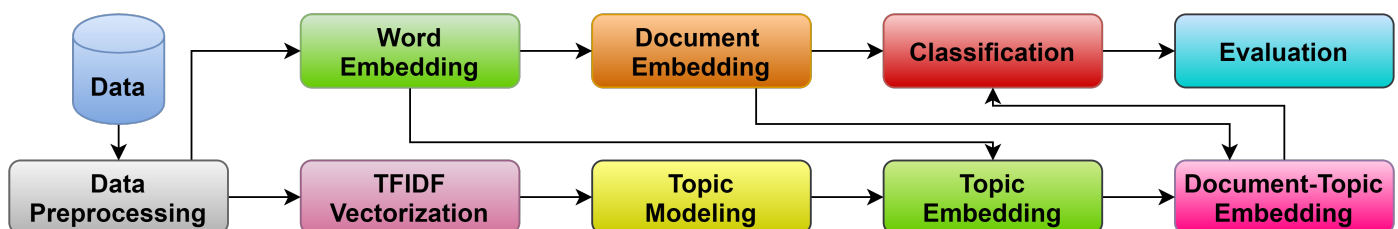


Figure 1. The proposed topic-based Sentiment Analysis using a contextual cues architecture.

The Data Preprocessing module cleans and transforms the textual data to make them suitable for analysis. The Word Embedding and TFIDF Vectorization modules encode the documents' words into vector representations. The Document Embedding module computes a vector for each document based on the Word Embedding. The Topic Modeling uses the TFIDF document vectorization to extract the topics and the most relevant keywords. The Topic Embedding module constructs the vector representation of topics using word embedding. The Document-Topic Embedding module computes the new context enhanced document embeddings using the topic and document embeddings that add both semantic and syntactic context to the vector representation. The classification module uses the new document-topic embeddings to classify documents and extract their polarity. The Evaluation module uses different metrics to determine the accuracy of the classification and determine the quality of the resulting models.

#### 3.1. Data Preprocessing Module

The preprocessing step is important because the text written by people can contain misspelled words, symbols, abbreviations etc. that need to be removed or replaced to facilitate the execution of the subsequently tasks with greater accuracy [28]. The initial text is preprocessed using the following steps:

- (1) The text is cleaned by removing all JavaScript functions, HTML tags, and URL;
- (2) The contractions are expanded;
- (3) The named entities are extracted while the rest of the text is lemmatized;
- (4) The punctuation and stop words excluding negations (i.e., no, not, etc.) are removed;
- (5) The text is transformed to lowercase and then split into tokens;
- (6) The tokens that have a length greater than 3 or are negations are kept. Using this aggressive text preprocessing improves the algorithms' time performance, as the



vocabulary is minimized to the essential tokens without excluding the terms which impact the polarity.

### 3.2. Word Embedding Module

The word embedding models used in this paper are WORD2VEC, FASTTEXT, and GLOVE. Each embedding model (WORDEMB) generates word representations in a vector space. The context of each word within a document is captured when employing these embeddings. Moreover, these models also encode both the relationship and the similarity between words from a semantic and syntactic perspective.

#### 3.2.1. WORD2VEC

WORD2VEC represents a textual dataset as a set of vectors and outputs a vector space [29]. The context similarity of a word within the dataset is determined by measuring the distance between the corresponding vectors in this space. WORD2VEC use either the Continuous Bag-Of-Words (CBOW) or SKIP-GRAM model to create the representation of words.

The CBOW model utilizes the context of a word as input and attempts to predict the word itself. The input layer of the model is represented by the one-hot encoded vectors corresponding to each context words. The average of the vectors from this layer is used to compute the input for the hidden layer. The weighted sum of the inputs, computed by the hidden layer, is sent to the next layer. The hidden layer sends the weighted sum of the inputs to the next layer. Each terms' probability value is computed by the network's last layer and is given as a final result in the form of a vector.

THE SKIP-GRAM model, as opposed to CBOW, starts with the word as input and tries to generate its context. The input layer is the target word vector, while the output layer consists of the vectors with the probability values of the words appearing in the context of a target word. The hidden layer sends the weighted input to the following layer. The SKIP-GRAM model is generally used to discover the semantic similarity between words. Therefore, if two words have a similar context, these words might also have a similar semantic.

#### 3.2.2. FASTTEXT

FASTTEXT is an unsupervised algorithm that uses the CBOW and SKIP-GRAM models for learning word embeddings [30]. This embedding is considered an extension of WORD2VEC as it follows a similar approach [31]. The difference is that the word is not considered the basic unit, but a bag of character n-grams. This facilitates better accuracy and a faster training time compared to WORD2VEC.

#### 3.2.3. GLOVE

GLOVE (Global Vectors) is an unsupervised model applied for learning word embeddings [32]. In comparison to the other models described, i.e., WORD2VEC, FASTTEXT, GLOVE consider both local and global statistics of word–word co-occurrences in the corpus to obtain the vector representations of the words. It uses a term co-occurrence matrix that stores, for each word, the frequency of its appearance in the same context with another word. GLOVE captures the relationship between words by using the ratio of co-occurrence probability. Using the co-occurrence probability ratio, it extracts information from all the word vectors and identifies word analogies or synonyms within the same contexts.

### 3.3. Document Embedding Module

The document embeddings DOC2VEC (Equation (1)) are generated for each document  $d_i$  in the dataset by adding the word embeddings for all the terms  $t$  ( $\text{WORDEMB}(t)$ ) in the document and divide the sum by the number of terms in the document ( $m_i$ ). We build a DOC2VEC for each WORDEMB we previously discussed.

$$\text{DOC2VEC}(d_i) = \frac{\sum_{t \in d_i} \text{WORDEMB}(t)}{m_i} \quad (1)$$

### 3.4. TFIDF Vectorization

The TFIDF (term frequency-inverse document frequency) Vectorization module uses a bag-of-words approach to vectorize the news articles given:

- (1) A textual corpus  $D = \{d_i | i = \overline{1, n}\}$  of size  $n = ||D||$  that contains documents  $d_i$ ;
- (2) A vocabulary  $V = \{t_1, \dots, t_m\}$  of size  $m = ||V||$  that contains the unique words or terms  $t_j$  in the dataset  $D$ .

A document  $d_i \in D (i = \overline{1, n})$  of length  $m_i (\neq m)$  is a multi-set of  $V$ , i.e.,  $d_i = (V, f) = \{t_1^{f(t_1, d_i)}, \dots, t_m^{f(t_m, d_i)}\}$  with  $f(t_j, d_i) \geq 0$  the multiplicity (co-occurrences) function which denotes the number of times  $t_j$  appears in document  $d_i$ . For simplicity, we will denote  $t_j^{f(t_j, d_i)}$  as  $t_{ij}$ , thus,  $d_i = \{t_{i1}, \dots, t_{im}\}$ .

TFIDF (Equation (2)) is defined using:

- (1) Term frequency  $\text{TF}(t_j, d_i)$  (Equation (3)) that computes the co-occurrences  $f(t_j, d_i)$  of a term  $t_j \in V$  in a document  $d_i$ ;
- (2) The inverse-document frequency  $\text{IDF}(t_j, D)$  (Equation (4)) which uses the number document  $n_j$  where a term  $t_j \in V$  appears to penalize frequent terms that bring no information gain;
- (3) The normalization factor  $\ell^2(d_i)$  (Equation (5)) to normalize TFIDF in the range  $[0, 1]$ .

$$\text{TFIDF}(t_j, d_i, D) = \frac{\text{TF}(t_j, d_i) \cdot \text{IDF}(t_j, D)}{\ell^2(d_i)} \quad (2)$$

$$\text{TF}(t_j, d_i) = f(t_j, d_i) \quad (3)$$

$$\text{IDF}(t_j, D) = \log_2 \frac{n}{n_j} \quad (4)$$

$$\ell^2(d_i) = \sqrt{\sum_{j=1}^m (\text{TF}(t_j, d_i) \cdot \text{IDF}(t_j, D))^2} \quad (5)$$

Using the term weights, we can construct a document-term matrix  $A = \{w_{ij} | i = \overline{1, n} \wedge j = \overline{1, m}\}$ , where rows correspond to documents and terms to columns. The cell value  $w_{ij}$  is the weight (e.g., TF, TFIDF, etc.) of term  $t_j$  in document  $d_i$ .

### 3.5. Topic Modeling Module.

This module utilizes statistical unsupervised methods to extract hidden latent semantic patterns within our dataset. We use the following models for this module.

This module utilizes statistical unsupervised Machine Learning methods, i.e., Topic Modeling, to extract hidden latent semantic patterns within our dataset. We use three generative statistical models for this module, i.e., Latent Dirichlet allocation (LDA) [12], Non-Negative Matrix Factorization (NMF) [13], Latent Semantic Indexing (LSI) [14], also known as Latent Semantic Analysis (LSA). The Topic Modeling algorithms use the document-term matrix  $A$  as input.

#### 3.5.1. Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA) is a probabilistic model that groups various terms with similar meaning that represent the same notions [12]. It is one of the most popular Topic Modeling approaches [33]. LDA algorithm relies on the assumption that random mixtures over latent topics can be used to generate documents. In this context, each topic is described by a multinomial distribution over the unique terms in the vocabulary. Thus, we can generate documents using techniques such as Gibbs that samples  $\langle \text{topic}, \text{words} \rangle$  pairs from a random mixture.

For  $k$  topics and a corpus of  $n$  documents  $D = \{d_i | i = \overline{1, n}\}$  where each document  $d_i$  is a sequence of  $m_i$  words  $t_j \in V, j = \overline{1, m}$  modeled as Poisson distributions, i.e.,  $m_i \sim \text{Poisson}(\xi)$  LDA uses the following process:

- (1) Determine a distribution of topics  $\theta_i$  for each document  $d_i$ ;
- (2) Determine a distribution of words  $\varphi_\kappa$  in a topic  $\kappa \in \overline{1, k}$ ;
- (3) For each word  $t_j$  in document  $d_i$ :
  - (a) Determine a topic  $z_{ij}$ ;
  - (b) Determine a word  $t_j^i$ .

The distribution of topics in document  $d_i$  is a Dirichlet distribution over the number of topics  $\theta_i \sim \text{Dirichlet}_k(\alpha)$  where  $\theta_i = \{\theta_{i\kappa} | i = \overline{1, n} \wedge \kappa = \overline{1, k} \wedge \sum_{\kappa=1}^k \theta_{i\kappa} = 0\}$  is a  $k$ -dimensional vector of probabilities,  $\theta_{i\kappa}$  is the probability of topic  $\kappa$  occurring in document  $d_i$ , and  $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_k\}$  is a  $k$ -dimensional vector of positive reals  $\alpha_k > 0$ .

The distribution of words in topic  $\kappa$  is also a Dirichlet distribution over the vocabulary  $\varphi_\kappa \sim \text{Dirichlet}_m(\beta)$  where  $\varphi_\kappa = \{\varphi_{\kappa j} | \kappa = \overline{1, k} \wedge j = \overline{1, m} \wedge \sum_{j=1}^m \varphi_{\kappa j} = 0\}$  is a  $m$ -dimensional vector of probabilities,  $\varphi_{\kappa j}$  is the probability of a word probability of word  $t_j$  occurring in topic  $\kappa$ , and  $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$  is a  $m$ -dimensional vector of positive reals  $\beta_j > 0$ .

For each document  $d_i$  ( $i = \overline{1, n}$ ), we define  $z_{i\kappa}$  described by a set of words  $t_{\kappa j}$  ( $j = \overline{1, m}$ ) of size  $m_i$ . Both  $z_{i\kappa}$  and  $t_{\kappa j}$  are multinomial distributions, i.e.,  $z_{i\kappa} = \text{Multinomial}_k(\theta_i)$  and  $t_{\kappa j} = \text{Multinomial}_m(\varphi_{z_{i\kappa}})$ .

### 3.5.2. Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is a dimensionality reduction paradigm based on linear algebra [34]. Experimental results prove that NMF is the best choice for extracting topics [35]. It is constructed on the premises that a matrix can be created as a product of two non-negative matrices. Thus, NMF factorizes a matrix  $A \in \mathbb{R}^{n \times m}$  into two non-negative matrices  $W \in \mathbb{R}^{n \times k}$  and  $H \in \mathbb{R}^{k \times m}$ . With regard to Topic Modeling, these 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.

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

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

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

$$\begin{aligned} H_{ij}^{\tau+1} &\leftarrow H_{ij}^\tau \frac{((W^\tau)^T A)_{ij}}{((W^\tau)^T W^\tau H^\tau)_{ij}} \\ W_{ij}^{\tau+1} &\leftarrow W_{ij}^\tau \frac{(A (H^{\tau+1})^T)_{ij}}{(W^\tau H^{\tau+1} (H^{\tau+1})^T)_{ij}} \end{aligned} \quad (8)$$



### 3.5.3. Latent Semantic Indexing

Latent Semantic Indexing (LSI) tries to solve the problem of synonyms by identifying terms that statistically appear together. The algorithm's main consideration is that the randomness of word choice within documents hides an underlying latent semantic structure. To determine this latent structure, LSI employs the matrix factorization technique called Singular Value Decomposition (SVD). It identifies syntactical different but semantically similar terms using a structure called hidden "concept" space.

Given the document-term matrix  $A$  with the size  $n \times m$  ( $n$  is number of documents,  $m$  is the number of terms in the vocabulary), LSI uses SVD to interactively factorize  $A$  into a product of three matrices, i.e.,  $A = U\Sigma V^T$ .

- (1)  $U$  is an  $n \times k$  matrix that denotes the document-topics association. The columns of  $U$  are the eigenvectors  $u$  of  $AA^T$ . Thus, these vectors identify the  $k$  non-zero eigenvalues  $\Sigma_L = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k)$  of  $AA^T$ . Moreover,  $u$  are unit orthogonal vectors, i.e.,  $U^T U = I$  and are also called left singular values because they satisfy the condition  $uA = \Sigma_L v$ .
- (2)  $V^T$  is an  $k \times m$  matrix that denotes the topic-keywords association. The columns of  $V$  are the eigenvectors  $v$  of  $A^T A$ . Thus, these vectors identify the  $r$  non-zero eigenvalues  $\Sigma_R = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$  of  $A^T A$ . Moreover,  $v$  are unit orthogonal vectors, i.e.,  $V^T V = I$ , and are also called right singular values because they satisfy the condition  $Av = \Sigma_R v$ .
- (3)  $\Sigma$  is a  $k \times k$  diagonal matrix which has on the diagonal the singular values or eigenvalues  $\sigma_i > 0$ . Thus, this diagonal matrix is defined as  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k)$  where each value is sorted in decreasing order from the one that holds the highest value to the one that represents the smallest one, i.e.,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > 0$ .

### 3.6. Topic Embedding Module

To encode the global context that is hidden in the latent semantic structures defined by the randomness of words, we employ a topic vector embedding TOPIC2VEC that encodes the keyword for the  $k$  topics extracted using one of the Topic Modeling algorithms. TOPIC2VEC takes the weighted average of the word embeddings WORDEMB of each relevant term  $t$  belonging to the topic  $z_i$  ( $i = 1, k$ ) and its probability distribution  $p(t|z_i)$  within the topic  $z_i$ . Equation (9) presents the proposed encoding, where the number of keywords considered for a topic  $z_i$  is  $n_i$ . We build a TOPIC2VEC for each topic model and WORDEMB we previously discussed.

$$\text{TOPIC2VEC}(z_i) = \frac{\sum_{t \in z_i} \text{WORDEMB}(t) \cdot p(t|z_i)}{n_i} \quad (9)$$

### 3.7. Document-Topic Embedding Module

The document with topics embeddings DOCTOPIC2VEC (Equation (10)) are generated by concatenating (operator  $\oplus$ ) the TOPIC2VEC of the most dominant topic of a document with the document's DOC2VEC. We build a DOCTOPIC2VEC for each TOPIC2VEC we previously discussed using the same WORDEMB for both the DOC2VEC and the TOPIC2VEC. By concatenating the DOC2VEC with TOPIC2VEC and obtaining the DOCTOPIC2VEC we manage to encode the local context given by the document embedding (DOC2VEC) with the global context given by the topic embedding (TOPIC2VEC).

$$\text{DOCTOPIC2VEC}(d_i, z_i) = \text{DOC2VEC}(d_i) \oplus \text{TOPIC2VEC}(z_i) \quad (10)$$

### 3.8. Classification Module

For classification, we use the Logistic Regression (LOGREG) algorithm, which serves as a baseline, and multiple Deep Neural Network (DNN) Architectures.

### 3.8.1. Logistic Regression

Logistic Regression (LOGREG) is a classification algorithm successfully used, in many cases, as a baseline for the Sentiment Analysis task to predict the class in which an observation can be categorized [36,37]. The algorithm tries to minimize the error of the estimations made using the log-likelihood and to determine the parameters that produce the best estimations using gradient descent [38]. The log-likelihood functions guarantee that the gradient descent algorithm can converge to the global minimum.

### 3.8.2. Deep Neural Network

Deep Neural Network (DNN) Architectures are used to classify the textual data and extract the polarity at the document level using DOC2VEC and DOCTOPIC2VEC. These architectures are developed using different fully connected or convolutional layers. The neural network units that make up these layers are PERCEPTRON, Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), Bidirectional GRU (BiGRU), and Bidirectional LSTM (BiLSTM). Figure 2 presents the combinations we use between these layers to create 17 DNN Architectures.

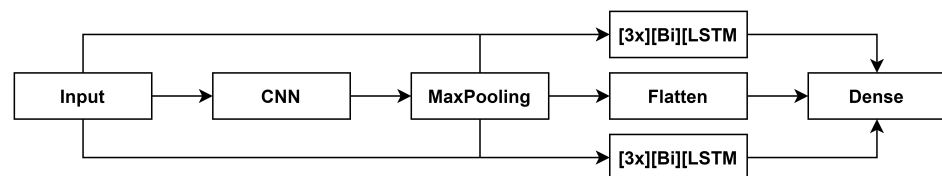


Figure 2. DNN architecture.

A *Perceptron* is a processing unit used to predict the label of an observation  $\hat{y} = \operatorname{argmax}_y f(x, y) \cdot \mathbf{w}$ . The function  $f(x, y)$  is used to map all the possible feature representation  $\langle x, y \rangle$  pairs to a new feature vector  $x$  and multiplies them by a weight vector  $\mathbf{w}$ . The  $x$  vector must fulfill the following conditions: (1) it has a positive number of elements, and (2) the values of its elements are real value numbers.

*GRU* is a recurrent unit that has two gating mechanisms: (1) the update gate, and (2) the reset gate. The update gate is used as both the forget gate and the input gate. The reset gate determines what percentage of the previous hidden state contributes to the candidate state of the new step. Furthermore, the GRU has only one state component, i.e., the hidden state.

*LSTM* is a recurrent unit that uses in its design two components to represent its state: (1) the hidden state is given by a short-term memory component, and (2) the current cell state is achieved by the long-term memory component. The LSTM unit comprise of a gating mechanism with three gates and a memory cell. The gating mechanism has the following gates: (1) the input gate, (2) the forget gate, and (3) the output gate. LSTM controls the gradients' values and avoids the problems of vanishing and exploding gradients by using the forget gate and the properties of the additive functions which compose the cell state gradients.

*Bidirectional RNN* (BiRNN) units allow for the use of information from both the previous and next state to make predictions about the current state. We use both BiGRU and BiLSTM in our models.

*Dense* layers are regular deeply connected neural network layers that contain only PERCEPTRON units.

*CNN* are Deep Neural Networks containing multiple convolution hidden layers that apply a filter to the activation function. After a convolutional layer, it is customary to use a layer that employs a pooling mechanism. The pooling layer reduces the dimensions of the data returned by the convolutional layer. This reduction is achieved by combining the results of the previous layer into a single layer neuron. The output of this single layer neuron is then used as the input of the following layer

Considering these layers, we propose six new CNN-(Bi)RNN architectures: CNN-BiGRU, CNN-3GRU, CNN-3BiGRU, CNN-BiLSTM, CNN-3LSTM, and CNN-3BiLSTM. When multiple recurrent layers are used, they form a stacked architecture.

These architectures are designed as follows:

- (1) The input layer that accepts the DOC2VEC or DOCTOPIC2VEC;
- (2) The CNN layer;
- (3) MAXPOOLING;
- (4) (Stacked) Recurrent layer(s), either BiLSTM or BiGRU;
- (5) DENSE layer containing PERCEPTRON.

Moreover, we implement the DNN Architecture presented in [3]. We use the same configurations for this DNN as presented in the original work. In the experiments, we name this architecture 4CNN-BiLSTM.

### 3.9. Evaluation Module

Evaluation metrics are used to better understand the performance of a model and for fine-tuning the model on a given classification task. In our case, we are solving a multi-class classification problem where we are trying to determine the different polarities of a given text. Thus, we use the weighted accuracy measure for evaluating our models because it takes into account the distribution of classes within the dataset. The weighted accuracy  $\omega A$  (Equation (11)) measures the per-class effectiveness of a classifier by employing the True/False Positive ( $TP_i$  and  $FP_i$ ) and True/False Negative ( $TN_i$  and  $FN_i$ ) rates. Given  $k$  classes  $y_i$  ( $i = 1, k$ ) and a dataset with  $n$  observations where  $n_i$  observations are labeled with class  $y_i$ , then we can compute a weight  $\omega_i$  for each class  $y_i$  using Equation (12).

$$\omega A = \frac{1}{k} \sum_{i=1}^k \omega_i \cdot \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \quad (11)$$

$$\omega_i = \frac{k \cdot n_i}{n} \quad (12)$$

## 4. Experimental Results

### 4.1. Dataset

For the experiments, we used a game reviews dataset containing context textual data posted on the MetaCritic website (<https://www.metacritic.com/>, accessed on 28 September 2021). The original version of this dataset is presented in [39] and improved in [40]. From the dataset, we use only the reviews and the polarity assigned for each review, although the collected raw data also contain other information. The polarity was transformed from the initial string format (i.e., positive, neutral, negative) into integer format (i.e., 2, 1, 0). This dataset contains over 90,500 game reviews with a polarity assigned that can be  $-1$  for negative, 1 for positive, or 0 for neutral. After preprocessing, we had with 90,165 with the following distribution of class: 15,721 negative, 22,433 neutral, and 52,011 positive. Out of the total number of 90,165 comments, 99.31% were in English, while 0.69% were in Spanish. As the number of comments in Spanish is negligible, we kept them to see if and how they impact our analysis. The vocabulary size is 23,016. The reviews contain from 1 to 1217 terms, with an average of 44.02. The reviews with a length between 1 and 50 words are the most common in the dataset, i.e., 66,713. The number of reviews with more than 100 words is 8538.

Experimentally, we have identified that the classification tasks perform better when the training and testing sets keep the proportions of the polarities of the entire dataset. For example, on a LOGREG classification experiment, if the data are split poorly, e.g., mostly positive reviews are used in the training dataset, the accuracy is lower than 55%. If equal proportions are created, based on three-quarters of the initial dataset, the accuracy improves up to 67%, whereas if the dataset is split using the initial proportions, this results in approximately 71% accuracy. Therefore, we conducted the classification experiments

using 80% of the dataset for training and 20% for testing, i.e., 72,132 reviews for training and 18,033 for testing. We preserved the polarity distribution of reviews in the both training and testing subsets. Moreover, we identified that the better the data are cleaned, i.e., as little as possible misspelled or foreign words are left in the dataset, the better the accuracy of the classification tasks is, with an increase of even 10% in accuracy compared to other data normalization methods.

#### 4.2. Word Embedding

To identify the best size for each WORDEMB, we tested various parameters and evaluated the resulting embeddings using a few approaches:

- (1) Computing accuracy by identifying how well the model recognizes analogies; the test is performed using the *questions-words* dataset [41] that contains pairs of analogies from different domains;
- (2) Identifying the cosine similarity between words with positive and negative connotation that appear in the dataset, i.e., (*fun, enjoyable*), (*boring, dull*), etc., and
- (3) Checking the most similar words with a common word in the dataset.

We determined experimentally using a grid search that (1) the best window size is four; (2) the number of epochs used for training is 30; (3) the initial learning rate is  $10^{-2}$ . Table 1 presents the final embedding sizes used for classification determined after evaluation.

**Table 1.** Embedding sizes.

Embedding	Model	Size		
		Embedding	DOC2VEC	DOCTOPIC2VEC
WORD2VEC	CBOW	256	256	512
	Skip-Gram	128	128	256
FASTTEXT	CBOW	256	256	512
	Skip-Gram	128	128	256
GLOVE	N/A	128	128	256

#### 4.3. Document Embeddings

Using the five WORDEMBs, we construct a DOC2VEC for each review as an average of the WORDEMBs for the terms in the document. The size of the DOC2VEC is equal to the size of the WORDEMB used.

#### 4.4. Topic Modeling

We identify 10 topics using the TFIDF document–term matrix as input together with the three Topic Modeling algorithms, i.e., LSA, NMF, and LSI. From each topic, the first 15 most relevant features are used in the algorithm for computing the topic embeddings. The number of documents where a topic is the most relevant is presented in Table 2. Tables 3–5 present the results for LDA, NMF, and LSI, respectively.

Analyzing the results of Table 3, we observe that LDA extracts diverse topics that can be interpreted using the keywords, e.g., Topic 0 is related to racing games, Topic 4 is related to sports games. Furthermore, LDA also manages to determine topics that find hidden latent semantic patterns that describe polarity, e.g., Topic 9 and Topic 7. We also note that LDA manages to detect and group together documents that have words in other languages than English, e.g., Topic 3, being the only algorithm among the three used in our analysis that picked up on the this negligible percent (0.69%) of comments.

As in the case of LDA, NMF (Table 4) manages to determine topics related to different games' genres and polarity. Unlike LDA, NMF manages to discover topics that that group together both polarity and game type, e.g., Topic 0, Topic 1, Topic 2. Furthermore, NMF fails to discover the comments that use a different language to English.

**Table 2.** Number of documents by dominant topic.

LDA		NMF		LSI	
Topic ID	Documents	Topic ID	Documents	Topic ID	Documents
7	83,743	0	28,957	0	81,260
9	5297	3	14,490	2	2037
3	620	1	10,400	1	1664
4	135	2	7356	9	1660
5	108	5	7253	8	1200
0	73	4	6458	7	947
2	64	7	5864	4	788
8	53	8	3643	6	369
1	38	9	2890	5	193
6	34	6	2854	3	47

Finally, LSI (Table 5) manages to determine topics related to the overall game play experience and users' opinion towards this aspect, e.g., Topic 0, Topic 2, Topic 5, Topic 6. Thus, most of the topics detected by LSI contain similar terms, e.g., game, play, to underline some of the polarity, e.g., good, awesome, beautiful, bad, fun, terrible.

**Table 3.** The most relevant features for topics generated with LDA.

Keywords and Impact	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
0	racing 68.54	rule 27.97	dinosaur 41.72	para 102.85	rock 92.42	splinter 44.41	competent 25.88	game 8390.99	sweet 23.92	game 1112.34
1	rally 60.52	guitar 25.26	worm 36.27	con 82.42	football 86.12	cent 42.98	cough 24.28	not 5522.88	piranha 19.7	good 896.02
2	uplay 43.88	medal 19.2	twitch 34.66	mas 63.26	ace 51.38	cell 42.15	bout 15.71	play 3433.44	harry 19.4	great 560.82
3	dirt 40.75	ego 18.57	bastion 31.99	no 62.47	manager 42.61	strike 37.32	pet 15.58	good 2937.24	nan 18.17	play 439.94
4	coaster 34.61	fez 16.11	myst 30.74	las 55.34	soccer 40.53	counter 36.7	nonstop 14.02	great 2137.95	breach 16.36	awesome 342.82
5	blah 33.72	honor 16.07	noir 27.25	son 40.9	innovate 23.84	addict 31.01	demigod 13.24	time 2065.73	potter 16.07	love 300.87
6	processor 27.44	legal 15.57	preview 24.86	ser 39.13	mesa 18.49	spore 27.85	splendid 13.12	fun 2010.35	napoleon 11.27	amazing 273.44
7	terror 26.92	outstanding 15.36	jest 23.08	bien 34.18	sherlock 18.1	annihilation 26.24	remote 11.94	no 1876.78	ruler 10.11	graphic 261.46
8	golf 26.56	wizardry 13.61	van 22.56	bom 24.11	bastard 17.46	outlast 26.19	fluidity 9.61	story 1857.25	refreshingly 9.72	fun 234.68
9	roller 24.31	article 12.29	enthral 19.52	sin 22.63	submit 16.62	halo 24.45	closet 9.57	graphic 1788	orientate 9.71	not 226.23
10	fallen 23.9	wine 11.57	hall 19.46	dos 22.61	doctor 15.63	dogs 23.59	dying 9.37	bad 1623.48	hoi 7.94	cool 149.77
11	theft 22.72	shameless 10.94	working 19.26	hay 21.38	cos 14.38	ing 21.2	nook 9.09	feel 1470.19	sensational 7.24	excellent 144.28
12	bye 20.01	apologize 9.060	tycoon 18.17	mal 21.33	overture 13.55	suck 21.09	bomb 8.8	thing 1359.17	bean 6.32	buy 138.63
13	grand 18.81	sailing 8.43	pour 17.89	excelente 20.64	thumb 13.27	gra 20.33	transformer 8.61	player 1346.69	crapy 5.26	year 127.48
14	car 18.47	tribunal 8.20	braid 17.5	vale 19.7	cheat 11.4	sports 20.13	man 8.57	buy 1328.62	fore 5.08	perfect 123.02



**Table 4.** The most relevant features for topics generated with NMF.

Keywords and Impact	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
0	game 6.64	good 4.91	great 4.94	story 2.05	play 5.24	not 5.98	amazing 4.86	fun 4.33	love 4.46	awesome 4.15
1	time 0.57	game 0.79	game 0.75	character 1.63	hour 0.60	bad 2.01	story 0.53	lot 0.74	game 0.29	graphic 1.10
2	no 0.54	graphic 0.39	graphic 0.38	feel 1.25	time 0.57	buy 1.37	graphic 0.52	friend 0.33	series 0.14	nice 0.24
3	player 0.39	pretty 0.13	story 0.15	level 1.23	year 0.52	money 0.83	game 0.52	hour 0.26	hate 0.12	cool 0.20
4	release 0.38	story 0.12	music 0.14	time 1.07	friend 0.42	people 0.67	simply 0.31	pretty 0.26	fan 0.12	game 0.19
5	people 0.38	series 0.09	atmosphere 0.13	combat 1.06	never 0.28	graphic 0.56	absolutely 0.25	worth 0.24	original 0.11	sound 0.18
6	strategy 0.38	nice 0.09	sound 0.11	no 0.96	player 0.27	review 0.50	episode 0.23	mode 0.22	absolutely 0.10	music 0.15
7	bug 0.35	shooter 0.09	job 0.09	enemy 0.91	free 0.26	worth 0.49	perfect 0.21	price 0.19	buy 0.09	car 0.13
8	year 0.33	year 0.08	fantastic 0.09	not 0.84	day 0.25	bug 0.47	music 0.20	bit 0.18	favorite 0.07	perfect 0.11
9	developer 0.31	sound 0.08	excellent 0.08	system 0.83	stop 0.24	pay 0.45	beautiful 0.19	nice 0.17	expansion 0.07	realistic 0.09
10	review 0.3	racing 0.07	recommend 0.07	puzzle 0.80	enjoy 0.21	thing 0.44	buy 0.16	recommend 0.17	hope 0.07	excellent 0.09
11	fan 0.29	perfect 0.06	action 0.07	interesting 0.78	long 0.19	no 0.44	fantastic 0.15	campaign 0.16	cool 0.07	story 0.08
12	never 0.28	music 0.06	expansion 0.06	world 0.76	game 0.17	waste 0.44	incredible 0.14	level 0.16	story 0.07	totally 0.07
13	work 0.27	world 0.06	lot 0.06	thing 0.73	start 0.16	work 0.38	sound 0.13	simple 0.15	perfect 0.07	racing 0.06
14	enjoy 0.26	atmosphere 0.05	puzzle 0.06	bit 0.73	single 0.16	release 0.38	atmosphere 0.13	challenge 0.15	fall 0.06	effect 0.06

#### 4.5. Topic to Vector

For each topic determined by an algorithm, we build a TOPIC2VEC as the weighted average of the WORDEMB and the importance of each relevant word that describes the topic. Thus, the size of the TOPIC2VEC is same as the size of the used WORDEMB.

#### 4.6. Document-Topic to Vector

A DOCTOPIC2VEC is created by concatenating the DOC2VEC with the TOPIC2VEC of the dominant topic for a document. The same WORDEMB is used when constructing the DOC2VEC and TOPIC2VEC embeddings that are concatenated for building the DOCTOPIC2VEC embedding. Thus, the size of the DOCTOPIC2VEC is twice the size of the used WORDEMB.

#### 4.7. Classification Algorithms

The classification experiments with LOGREG are computed using both DOC2VEC and DOCTOPIC2VEC. For this model to achieve a stronger regularization, we set the inverse regularization parameter  $C$  to  $10^{-5}$ .

Using the GRU units, we built multiple models:

- (1) One with a single GRU Layer;
- (2) One with three GRU layers (3GRU);
- (3) One with a single BiGRU Layer, and
- (4) One with three BiGRU layers (3BiGRU).

**Table 5.** The most relevant features for topics generated with LSI.

Keywords and Impact	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9
0	game 0.58	good 0.77	great 0.79	game 0.53	play 0.57	play 0.39	love 0.47	fun 0.74	love 0.72	awesome 0.64
1	not 0.34	game 0.23	game 0.19	play 0.34	fun 0.41	not 0.36	not 0.35	not 0.24	awesome 0.19	graphic 0.44
2	good 0.24	great 0.15	love 0.11	love 0.11	time 0.12	great 0.33	play 0.3	graphic 0.12	player 0.15	bad 0.27
3	play 0.24	graphic 0.05	fun 0.09	buy 0.08	hour 0.12	buy 0.16	story 0.27	awesome 0.11	good 0.1	amazing 0.19
4	great 0.16	awesome 0.05	awesome 0.08	year 0.06	level 0.11	bad 0.13	amazing 0.17	lot 0.09	map 0.1	no 0.15
5	time 0.13	play 0.05	amazing 0.07	friend 0.03	lot 0.1	good 0.12	awesome 0.16	nice 0.08	no 0.09	nice 0.1
6	fun 0.12	amazing 0.04	graphic 0.03	amazing 0.03	player 0.09	fun 0.1	graphic 0.13	bad 0.08	buy 0.09	sound 0.09
7	graphic 0.11	love 0.03	fantastic 0.03	release 0.03	love 0.09	money 0.09	episode 0.1	game 0.07	expansion 0.08	car 0.07
8	story 0.11	year 0.01	music 0.03	steam 0.03	friend 0.08	pay 0.06	bad 0.09	puzzle 0.07	single 0.07	control 0.07
9	no 0.1	racing 0.01	puzzle 0.03	money 0.03	story 0.08	people 0.06	character 0.07	worth 0.06	fun 0.07	year 0.05
10	bad 0.1	excellent 0.01	story 0.03	fun 0.03	character 0.07	player 0.05	buy 0.06	pretty 0.06	unit 0.07	bug 0.05
11	feel 0.08	perfect 0.01	recommend 0.02	never 0.03	awesome 0.07	free 0.05	puzzle 0.05	buy 0.05	bug 0.07	port 0.05
12	buy 0.08	strategy 0.01	excellent 0.02	awesome 0.03	amazing 0.06	bug 0.05	adventure 0.04	car 0.04	release 0.07	version 0.05
13	thing 0.08	music 0.01	atmosphere 0.02	pay 0.03	mode 0.05	friend 0.05	music 0.04	cool 0.04	campaign 0.07	terrible 0.05
14	love 0.08	adventure 0.01	beautiful 0.02	free 0.03	enjoy 0.05	year 0.05	atmosphere 0.03	price 0.04	lot 0.06	play 0.05

All these models have a final DENSE Layer used for the final classification. Each GRU layer is initialized with 128 units and a dropout of 0.2. The activation for the update gate is the *sigmoid* function (Equation (13)) and for the reset gate the *hyperbolic tangent* function (Equation (14)). The sigmoid function is defined in  $(0, 1)$  and is used for models that utilize the probability of a variable. The hyperbolic tangent function is defined on the  $[-1, 1]$  interval and it is mainly used to better differentiate between the strongly negative values and 0. The DENSE output layer is initialized using the *softmax* activation function (Equation (15)) and with three as the dimension, corresponding with the number of possible values for the polarity. For multiclass classification, the softmax function is a generalized logistic activation function used to normalize the output of a network  $x = (x_1, x_2, \dots, x_K)$  to a probability distribution over predicted output classes  $i = \overline{1, K}$ . In our case, we set  $K = 3$ , as we are predicting the positive, negative, or neutral polarity.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (13)$$

$$\text{tanh}(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (14)$$

$$\text{softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (15)$$

Using the LSTM units, we built multiple models to mirror the GRU architectures:

- (1) One with a single LSTM Layer;
- (2) One with three LSTM layers (3LSTM);
- (3) One with a single BiLSTM Layer, and
- (4) One with three BiLSTM layers (3BiLSTM).

We keep the same initialization parameters for the LSTM and DENSE layer as for the GRU architectures. The activation for the input, output, and forget gates is the *sigmoid* function, while for the hidden state and the cell input activation vector is the *hyperbolic tangent* function. The LSTM models use the same loss function and optimizer parameter.

As CNN architectures proved to be an asset for text classification [10], we build a CNN Sentiment Analysis architecture with three layers: CNN, MAXPOOLING, and DENSE. We initialize the filters to 64 and the kernel size to half the size of the input vector, i.e., DOC2VEC or DOCTOPIC2VEC. We also add the CNN and MAXPOOLING layers on top of the four GRU and four LSTM architectures to determine if convolutions on top of recurrent layers improve the classification as in [9]. Moreover, we implement the Deep Learning Architecture presented in [3] using the same configuration. In the experiments, we name this architecture 4CNN-BiLSTM.

For all the Deep Neural Network Architectures, we utilize a batch size of 5000 to accurately estimate the gradient error in the detriment of the drawback known as slowing the convergence of the learning process. The loss is computed using *categorical cross entropy* and the applied optimizer is *Adam*. Each network is trained with a maximum of 200 epochs, using an automated stopping mechanism that stops the execution if the accuracy is not improved during 20 successive epochs.

#### 4.8. Implementation

The entire pipeline is implemented in *Python3.7*. For named entity recognition and lemmatization we used the *en\_core\_web\_sm* from the *SpaCy* [42] package. We use the *gensim* [43] and *python-glove* [44] packages for the WORDEMBS and the *scikit-learn* [45] package for the TFIDF vectorization, LOGREG classifier, and Topic Modeling algorithms. All the DNN Architectures are implemented in *Keras* [46] with *TensorFlow* [47] as the tensor backend engine. The experiments are run on an NVIDIA® DGX Station™. The code is freely available online on GitHub at <https://github.com/cipriantruica/DocTopic2Vec>.

#### 4.9. Results

Tables 6–10 present the average accuracy obtained after 10 distinct training experiments. As a baseline for the embeddings, we use the DOC2VEC, while, for classification, we use LOGREG. We utilize Stratified Cross-Validation for splitting the dataset into 80–20% training–testing sets with random seeding, i.e., 72,132 reviews for training and 18,033 for testing. Furthermore, we identified that the better the data are cleaned, i.e., as little as possible misspelled or foreign words left in the dataset, the better the accuracy of the classification tasks is, with an increase of even 10% in accuracy compared to other data normalization methods.

The proposed DOCTOPIC2VEC improves significantly the detection of polarity at document-level, for both NMF and LSI, over the simple implementation of DOC2VEC with over 5%. In the case of the LDA, we observe a decrease in accuracy. When using the WORD2VEC CBOW model to construct DOCTOPIC2VEC (Table 6), we obtain the best results and the overall best accuracy (i.e., 0.7718) for all our experiments with the GRU architecture and LSI topic model. For the WORD2VEC SKIP-GRAM (Table 7), the CNN-BiGRU architecture with the DOCTOPIC2VEC for NMF obtain the best results. The CNN-BiGRU architecture also achieves the best results when building the DOCTOPIC2VEC using FASTTEXT and LSI (Tables 8 and 9). When using GLOVE and the LSI topic model

to construct the DOCTOPIC2VEC (Table 10), the best results are obtained with the novel CNN-3BiGRU architecture.

**Table 6.** Experiments using the WORD2VEC CBOW embeddings of size 256. (Note: Bold marks the best accuracy obtained for the combination of Algorithm and Doc2Vec/DocTopic2Vec).

Algorithm	Doc2VEC	DOCTOPIC2VEC		
		LDA	NMF	LSI
LOGREG	0.7259	0.6827	0.7614	0.7610
GRU	0.7429	0.5770	0.7663	<b>0.7718</b>
BiGRU	0.7440	0.5768	0.7681	0.7713
3GRU	0.7464	0.5769	0.7708	0.7713
3BiGRU	0.7469	0.5769	0.7700	0.7668
LSTM	0.7407	0.5771	0.7686	0.7715
BiLSTM	0.7457	0.5769	0.7678	0.7686
3LSTM	0.7459	0.5770	<b>0.7713</b>	0.7689
3BiLSTM	<b>0.7491</b>	0.5768	0.7685	0.7674
CNN	0.7230	0.3111	0.7515	0.7583
CNN-GRU	0.7400	0.5768	0.7613	0.7589
CNN-BiGRU	0.7455	0.7636	0.7643	0.7708
CNN-3GRU	0.7415	0.5768	0.7603	0.7631
CNN-3BiGRU	0.7463	0.5768	0.7596	0.7615
CNN-LSTM	0.7423	0.5769	0.7635	0.7679
CNN-BiLSTM	0.7449	<b>0.7647</b>	0.7653	0.7662
CNN-3LST	0.7401	0.5768	0.7616	0.7633
CNN-3BiLSTM	0.7411	0.7608	0.7603	0.7618
4CNN-BiLSTM [3]	0.7244	0.7463	0.7425	0.7534

**Table 7.** Experiments using the WORD2VEC SKIP-GRAM embeddings of size 128. (Note: Bold marks the best accuracy obtained for the combination of Algorithm and Doc2Vec/DocTopic2Vec).

Algorithm	Doc2VEC	DOCTOPIC2VEC		
		LDA	NMF	LSI
LOGREG	0.7215	0.5772	0.7533	0.7542
GRU	0.7334	0.5768	0.7536	0.7532
BiGRU	0.7359	0.5767	0.7553	0.7563
3GRU	0.7378	0.5768	0.7563	0.7579
3BiGRU	0.7390	0.5767	0.7557	0.7608
LSTM	0.7322	0.5768	0.7530	0.7531
BiLSTM	0.7375	0.5767	0.7557	0.7559
3LSTM	0.7355	0.5768	0.7540	0.7563
3BiLSTM	0.7401	0.5768	0.7584	0.7593
CNN	0.7230	0.5768	0.7507	0.7560
CNN-GRU	0.7387	0.5768	0.7606	0.7543
CNN-BiGRU	0.7445	0.7574	<b>0.7655</b>	0.7628
CNN-3GRU	0.7392	0.5768	0.7569	0.7589
CNN-3BiGRU	<b>0.7456</b>	0.7573	0.7648	0.7639
CNN-LSTM	0.7405	0.5768	0.7599	0.7599
CNN-BiLSTM	0.7430	0.7586	0.7617	0.7594
CNN-3LSTM	0.7306	0.5768	0.7622	0.7586
CNN-3BiLSTM	0.7384	0.7578	0.7630	<b>0.7640</b>
4CNN-BiLSTM [3]	0.7332	<b>0.7592</b>	0.7559	0.7573

**Table 8.** Experiments using the FASTTEXT CBOW embeddings of size 256. (Note: Bold marks the best accuracy obtained for the combination of Algorithm and Doc2Vec/DocTopic2Vec).

Algorithm	Doc2Vec	DocTopic2Vec		
		LDA	NMF	LSI
LOGREG	0.7201	0.6699	0.7518	0.7462
GRU	0.7412	0.5772	0.7547	0.7553
BiGRU	0.7399	0.5772	0.7586	0.7573
3GRU	0.7434	0.5771	0.7591	0.7558
3BiGRU	<b>0.7450</b>	0.5772	0.7575	0.7521
LSTM	0.7399	0.5772	0.7550	0.7565
BiLSTM	0.7405	0.5772	<b>0.7599</b>	0.7553
3LSTM	0.7416	0.5768	0.7584	0.7562
3BiLSTM	0.7441	0.5772	0.7591	0.7568
CNN	0.7206	0.2279	0.7496	0.7479
CNN-GRU	0.7343	0.5768	0.7555	0.7584
CNN-BiGRU	0.7440	<b>0.7539</b>	0.7547	<b>0.7620</b>
CNN-3GRU	0.7347	0.5768	0.7523	0.7578
CNN-3BiGRU	0.7409	0.7528	0.7556	0.7571
CNN-LSTM	0.7376	0.5768	0.7521	0.7522
CNN-BiLSTM	0.7421	0.7504	0.7554	0.7528
CNN-3LSTM	0.7376	0.5768	0.7536	0.7511
CNN-3BiLSTM	0.7366	0.7536	0.7566	0.7582
4CNN-BiLSTM [3]	0.7206	0.7430	0.7431	0.7394

**Table 9.** Experiments using the FASTTEXT SKIP-GRAM embeddings of size 128. (Note: Bold marks the best accuracy obtained for the combination of Algorithm and Doc2Vec/DocTopic2Vec).

Algorithm	Doc2Vec	DocTopic2Vec		
		LDA	NMF	LSI
LOGREG	0.7227	0.5771	0.7514	0.7522
GRU	0.7329	0.5770	0.7583	0.7535
BiGRU	0.7359	0.5769	0.7625	0.7579
3GRU	0.7377	0.5768	0.7613	0.7561
3BiGRU	0.7389	0.5768	0.7639	0.7608
LSTM	0.7331	0.5768	0.7616	0.7515
BiLSTM	0.7349	0.5769	0.7613	0.7566
3LSTM	0.7355	0.5771	0.7609	0.7567
3BiLSTM	0.7400	0.5768	0.7637	0.7599
CNN	0.7240	0.2274	0.7551	0.7563
CNN-GRU	0.7441	0.5768	0.7612	0.7631
CNN-BiGRU	0.7467	0.7612	<b>0.7653</b>	<b>0.7662</b>
CNN-3GRU	0.7392	0.5768	0.7612	0.7650
CNN-3BiGRU	<b>0.7452</b>	0.7620	0.7644	0.7634
CNN-LSTM	0.7418	0.5768	0.7603	0.7612
CNN-BiLSTM	0.7426	0.7610	0.7617	0.7636
CNN-3LSTM	0.7295	0.5768	0.7620	0.7624
CNN-3BiLSTM	0.7428	<b>0.7625</b>	0.7632	0.7608
4CNN-BiLSTM [3]	0.7395	0.7597	0.7581	0.7546



**Table 10.** Experiments using the GLOVE embeddings of size 128. (Note: Bold marks the best accuracy obtained for the combination of Algorithm and Doc2Vec/DocTopic2Vec)

Algorithm	Doc2Vec	DOCTOPIC2VEC		
		LDA	NMF	LSI
LOGREG	0.7089	0.6587	0.7408	0.7432
GRU	0.7097	0.5769	0.7440	0.7461
BiGRU	0.7174	0.5768	0.7457	0.7457
3GRU	0.7255	0.5768	0.7450	0.7480
3BiGRU	0.7301	0.5768	0.7475	0.7525
LSTM	0.7156	0.5768	0.7415	0.7472
BiLSTM	0.7199	0.5768	0.7454	0.7501
3LSTM	0.7264	0.5767	0.7458	0.7481
3BiLSTM	<b>0.7307</b>	0.5767	0.7479	0.7532
CNN	0.7032	0.5745	0.7447	0.7426
CNN-GRU	0.7280	0.5768	0.7504	0.7424
CNN-BiGRU	0.7330	0.7420	<b>0.7552</b>	0.7541
CNN-3GRU	0.7248	0.5768	0.7450	0.7447
CNN-3BiGRU	0.7299	0.5768	0.7548	<b>0.7560</b>
CNN-LSTM	0.7242	0.5768	0.7496	0.7506
CNN-BiLSTM	0.7293	<b>0.7428</b>	0.7543	0.7520
CNN-3LSTM	0.7256	0.5768	0.7489	0.7424
CNN-3BiLSTM	0.7238	0.5768	0.7517	0.7474
4CNN-BiLSTM [3]	0.7190	0.7423	0.7509	0.7453

The experimental results show that the polarity detection accuracy is improved if the Topic Modeling algorithms meet at least one of the following two conditions:

- (1) The document to dominant topic distribution is balanced and manages to group context-related documents together;
- (2) The importance of the terms that belong to the topic have a small value range in order to enhance the document vectorization with the context-dependent terms.

Thus, depending on the used Topic Modeling algorithm, the overall performance of the proposed model changes.

LSI manages to meet the first condition needed to improve the accuracy of the polarity detection task. The importance of the relevant keywords for a topic detected with LSI has values  $\leq 1$ . These values influence the TOPIC2VEC values (Table 5). Thus, the final DOCTOPIC2VEC's values remain balanced for the entire encoding, and the context extracted through Topic Modeling in conjunction with the distribution of document to dominant topic improves the classification task (Table 2).

NMF satisfies both conditions needed to improve the accuracy of the Sentiment Analysis task. For NMF, the importance of the relevant keywords is not normalized and has values in the range  $[0, 6.64]$ , but the majority of the values are still  $\leq 1$  (Table 4). When building the TOPIC2VEC for NMF, some dimensions are going to have higher values which add more importance to the context-related words. During the training of the model, the higher values introduce bias to these dimensions in the classification task and manage to influence the accuracy of detecting the document-level polarity by better grouping documents together. Moreover, the more balanced distribution of documents to the dominant topic obtained by the NMF (Table 2) also influences the context-based grouping of documents.

LSA does not meet any of the two conditions needed for an improved polarity detection model; thus, the accuracy decreases. When using LDA to build the DOCTOPIC2VEC, LOGREG and RNN results are influenced by the importance of a word to a topic and the distribution of the document to the dominant topic. The relevant words for some topics have high importance (Table 3). Thus, the TOPIC2VEC values are larger than the DOC2VEC

values. Because the distributions of document to dominant topic is not balanced (Table 2), the TOPIC2VEC with the highest values is assigned to the majority of documents. When concatenating the DOCTOPIC2VEC, the second half of the embedding and the imbalanced distribution of the document to dominant topic influences the classification task and the results are similar to flipping a coin.

For the CNN with bidirectional RNNs models, we obtain better results for DOCTOPIC2VEC constructed with LDA, as the Deep Neural Network uses convolutions to select values. Therefore, the impact of the second half of the TOPIC2VEC values, as well as the imbalanced document grouping, are minimized, and for some tests, we obtain better results.

We observe that, on average, we obtain better results when using bidirectional models for both the fully connected (e.g., BiGRU, 3BiGRU, etc.) and convolutional architectures (e.g., CNN-BiGRU, CNN-3BiGRU, etc.). We note that, on average, the proposed new architectures perform better for this task. Furthermore, our models outperform the state of the art 4CNN-BiLSTM architecture. Stacking multiple layers of RNNs (e.g., 3GRU, 3BiGRU, 3GRU, etc.) with or without using a CNN brings very little improvement in accuracy over the architectures with a single layer. In case they are better, they only bring a  $\sim 1\%$  improvement. The same observations can be deduced for the architectures that stacks multiple CNNs, i.e., 4CNN-BiLSTM.

As a final remark, we compare our results with the results obtained on the same dataset in [48] and in [40]. Our proposed Deep Neural Network architectures outperform with  $\sim 10\%$  the Transformer-based models in [40] that obtained an accuracy of only 0.67.

## 5. Conclusions

In this paper, we propose DOCTOPIC2VEC, a novel embedding that incorporates contextual cues through the use of Topic Modeling. We use a dataset with game reviews to learn different WORD2VEC models, i.e., WORD2VEC, FASTTEXT, and GLOVE. Applying the different WORD2VEC, we create DOC2VECs for each review and TOPIC2VECs for each topic extracted by LDA, NMF, and LSI. A DOCTOPIC2VEC is constructed for each review as the concatenation of its DOC2VEC with the TOPIC2VEC for its dominant topic. Both DOC2VEC and TOPIC2VEC use the same WORD2VEC when are concatenated into the DOCTOPIC2VEC. To prove the efficiency of the new proposed DOCTOPIC2VEC in the task of Document-Level Sentiment Analysis, we implement different Deep Neural Network (DNN) Architectures using combinations of fully connected (i.e., GRU, LSTM, BiGRU, BiLSTM, DENSE) and convolutional (CNN) layers. Furthermore, we propose six novel Convolutional-based Recurrent DNN Architectures that outperform the state of the art 4CNN-BiLSTM architecture [3].

The experimental results show an improvement in accuracy in determining the document-level polarity of  $\sim 5\%$  when employing the new proposed context-enhanced DOCTOPIC2VEC for the NMF- and LSI-based topic embeddings over the baseline, i.e., DOC2VEC with LOGREG. These embeddings manage to improve the classification by:

- (1) Grouping context-related documents together through the document to dominant topic distribution;
- (2) Enhancing the document vectorization with the importance of context-dependent terms that belong to the topic.

Furthermore, we observe that if the Topic Modeling algorithm does not meet these requirements, the polarity detection accuracy drops significantly, as in the case of LDA. Finally, we want to note that our proposed CNN-(Bi)RNN architectures outperform the best performing state-of-the-art model with  $\sim 10\%$  applied on the same dataset in [40].

By combining Topic Modeling with the Sentiment Analysis task and by obtaining better results, we manage to answer ( $Q_1$ ) and to fulfill objective ( $O_1$ ). We answer ( $Q_2$ ) by adding local and global context through the novel DOCTOPIC2VEC embedding and improving the accuracy of detecting the polarity of textual data, thus achieving objective ( $O_2$ ). By introducing novel CNN-(Bi)RNN Deep Learning Architectures that improve the

accuracy of the Sentiment Analysis task, we answer our final research question ( $Q_3$ ) and complete objective ( $O_3$ ).

As future work, we aim to test other embeddings, e.g., MITTENS [49] which learns domain-specific representations, MOE [50] which manages word misspellings, BERT [22] which considers the word's occurrence and position when computing its context. Furthermore, we plan to explore how the WORDEMBs used in this paper could be used with other neural networks, such as Hierarchical Attention Networks or Deep Belief Networks.

**Author Contributions:** Conceptualization, C.-O.T., E.-S.A. and M.-L.S.; methodology, C.-O.T., E.-S.A. and M.-L.S.; software, C.-O.T., E.-S.A. and M.-L.S.; validation, C.-O.T., E.-S.A. and M.-L.S.; formal analysis, C.-O.T., E.-S.A., M.-L.S. and A.P.; investigation, C.-O.T., E.-S.A., M.-L.S. and A.P.; resources, C.-O.T. and E.-S.A.; data curation, E.-S.A. and M.-L.S.; writing—original draft preparation, C.-O.T., E.-S.A., M.-L.S. and A.P.; writing—review and editing, C.-O.T., E.-S.A. and A.P.; visualization, C.-O.T. and E.-S.A.; supervision, C.-O.T. and E.-S.A.; project administration, C.-O.T., E.-S.A. and A.P.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Acknowledgments:** The research presented in this paper was supported in part by the German Federal Ministry of Education and Research (BMBF) through the project QURATOR (Grant No. 03WKDA1F) and PANQURA (Grant No. 03COV03F), and the German Academic Exchange Service (DAAD) through the projects “Deep-Learning Anomaly Detection for Human and Automated Users Behavior” (Grant No. 91809358) and “AWAKEN: content-Aware and netWork-Aware faKE News mitigation” (Grant No. 91809005).

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

## Abbreviations

The following abbreviations are used in this manuscript:

WORDEMB	Word Embedding
WORD2VEC	Word to vector
GLOVE	Global Vectors
DOC2VEC	Document to Vector
CBOW	Continuous Bag-of-Words
TOPIC2VEC	Topic to Vector
DOCTOPIC2VEC	Document-Topic to Vector
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
BiRNN	Bidirectional Recurrent Neural Networks
GRU	Gated Recurrent Unit
BiGRU	Bidirectional Gated Recurrent Unit
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Networks
LDA	Latent Dirichlet Allocation
LSI	Latent Semantic Indexing
NMF	Non-Negative Matrix Factorization
LOGREG	Logistic Regression
TF	Term Frequency
IDF	Inverse Document Frequency
TFIDF	Term Frequency–Inverse Document Frequency
ABCDM	Attention-based Bidirectional CNN-RNN Deep Model

TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative

## References

- Naseem, U.; Razzak, I.; Musial, K.; Imran, M. Transformer based Deep Intelligent Contextual Embedding for Twitter sentiment analysis. *Future Gener. Comput. Syst.* **2020**, *113*, 58–69. [\[CrossRef\]](#)
- Rezaeinia, S.M.; Rahmani, R.; Ghodsi, A.; Veisi, H. Sentiment analysis based on improved pre-trained word embeddings. *Expert Syst. Appl.* **2019**, *117*, 139–147. [\[CrossRef\]](#)
- Rhanoui, M.; Mikram, M.; Yousfi, S.; Barzali, S. A CNN-BiLSTM Model for Document-Level Sentiment Analysis. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 832–847. [\[CrossRef\]](#)
- Yusof, N.N.; Mohamed, A.; Abdul-Rahman, S. Context Enrichment Model Based Framework for Sentiment Analysis. In *International Conference on Soft Computing in Data Science*; Springer: Singapore, 2019; pp. 325–335. [\[CrossRef\]](#)
- Yadav, A.; Vishwakarma, D.K. Sentiment analysis using deep learning architectures: A review. *Artif. Intell. Rev.* **2020**, *53*, 4335–4385. [\[CrossRef\]](#)
- Vijayaragavan, P.; Ponnusamy, R.; Aramudhan, M. An optimal support vector machine based classification model for sentimental analysis of online product reviews. *Future Gener. Comput. Syst.* **2020**, *111*, 234–240. [\[CrossRef\]](#)
- Nemes, L.; Kiss, A. Social media sentiment analysis based on COVID-19. *J. Inf. Telecommun.* **2020**, *5*, 1–15. [\[CrossRef\]](#)
- Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010; pp. 1045–1048.
- Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1746–1751. [\[CrossRef\]](#)
- Wang, R.; Li, Z.; Cao, J.; Chen, T.; Wang, L. Convolutional Recurrent Neural Networks for Text Classification. In Proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary, 14–19 July 2019; pp. 1–6. [\[CrossRef\]](#)
- 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; pp. 1188–1196.
- Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
- Arora, S.; Ge, R.; Moitra, A. Learning Topic Models—Going beyond SVD. In Proceedings of the Annual Symposium on Foundations of Computer Science, New Brunswick, NJ, USA, 20–23 October 2012; pp. 1–10. [\[CrossRef\]](#)
- Deerwester, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K.; Harshman, R. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.* **1990**, *41*, 391–407. [\[CrossRef\]](#)
- D’Andrea, A.; Ferri, F.; Grifoni, P.; Guzzo, T. Approaches, tools and applications for sentiment analysis implementation. *Int. J. Comput. Appl.* **2015**, *125*, 26–33. [\[CrossRef\]](#)
- Aziz, M.N.; Firmanto, A.; Fajrin, A.M.; Ginardi, R.H. Sentiment Analysis and Topic Modelling for Identification of Government Service Satisfaction. In Proceedings of the International Conference on Information Technology, Computer, and Electrical Engineering, Semarang, Indonesia, 27–28 September 2018; pp. 125–130. [\[CrossRef\]](#)
- Yoon, H.G.; Kim, H.; Kim, C.O.; Song, M. Opinion polarity detection in Twitter data combining shrinkage regression and topic modeling. *J. Inf.* **2016**, *10*, 634–644. [\[CrossRef\]](#)
- Usama, M.; Ahmad, B.; Song, E.; Hossain, M.S.; Alrashoud, M.; Muhammad, G. Attention-based sentiment analysis using convolutional and recurrent neural network. *Future Gener. Comput. Syst.* **2020**, *113*, 571–578. [\[CrossRef\]](#)
- Basiri, M.E.; Nemati, S.; Abdar, M.; Cambria, E.; Acharya, U.R. ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis. *Future Gener. Comput. Syst.* **2021**, *115*, 279–294. [\[CrossRef\]](#)
- García-Pablos, A.; Cuadros, M.; Rigau, G. W2VLDA: Almost unsupervised system for Aspect Based Sentiment Analysis. *Expert Syst. Appl.* **2018**, *91*, 127–137. [\[CrossRef\]](#)
- Al-Janabi, O.M.; Malim, N.H.A.H.; Cheah, Y.N. Aspect Categorization Using Domain-Trained Word Embedding and Topic Modelling. In *Advances in Electronics Engineering*; Springer: Singapore, 2020; pp. 191–198. [\[CrossRef\]](#)
- Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186. [\[CrossRef\]](#)
- Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:cs.CL/1907.11692.
- Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of the International Conference on Learning Representations, Virtual Conference, 26 April–1 May 2020; pp. 1–17.
- Biswas, E.; Karabulut, M.E.; Pollock, L.; Vijay-Shanker, K. Achieving Reliable Sentiment Analysis in the Software Engineering Domain using BERT. In Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, Australia, 28 September–2 October 2020; pp. 162–173. [\[CrossRef\]](#)

26. Zhao, L.; Li, L.; Zheng, X.; Zhang, J. A BERT based Sentiment Analysis and Key Entity Detection Approach for Online Financial Texts. In Proceedings of the 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Dalian, China, 5–7 May 2021; pp. 1233–1238. [\[CrossRef\]](#)
27. Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep Contextualized Word Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; Volume 1, pp. 2227–2237. [\[CrossRef\]](#)
28. Alasadi, S.A.; Bhaya, W.S. Review of data preprocessing techniques in data mining. *J. Eng. Appl. Sci.* **2017**, *12*, 4102–4107. [\[CrossRef\]](#)
29. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013; pp. 1–12.
30. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [a\\_00051. \[CrossRef\]](#)
31. Mikolov, T.; Grave, E.; Bojanowski, P.; Puhresch, C.; Joulin, A. Advances in Pre-Training Distributed Word Representations. In Proceedings of the International Conference on Language Resources and Evaluation, Miyazaki, Japan, 7–12 May 2018; pp. 52–55.
32. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [\[CrossRef\]](#)
33. Niu, L.; Dai, X.; Zhang, J.; Chen, J. Topic2Vec: Learning distributed representations of topics. In Proceedings of the International Conference on Asian Language Processing, Suzhou, China, 24–25 October 2015; pp. 193–196. [\[CrossRef\]](#)
34. Wang, Y.X.; Zhang, Y.J. Nonnegative matrix factorization: A comprehensive review. *IEEE Trans. Knowl. Data Eng.* **2012**, *25*, 1336–1353. [\[CrossRef\]](#)
35. Chen, Y.; Zhang, H.; Liu, R.; Ye, Z.; Lin, J. Experimental explorations on short text topic mining between LDA and NMF based Schemes. *Knowl.-Based Syst.* **2019**, *163*, 1–13. [\[CrossRef\]](#)
36. Petrescu, A.; Truica, C.O.; Apostol, E.S. Sentiment Analysis of Events in Social Media. In Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 5–7 September 2019; pp. 143–149. [\[CrossRef\]](#)
37. Mitroi, M.; Truică, C.O.; Apostol, E.S.; Florea, A.M. Sentiment Analysis using Topic-Document Embeddings. In Proceedings of the 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 3–5 September 2020; pp. 75–82. [\[CrossRef\]](#)
38. Yi, D.; Ji, S.; Bu, S. An Enhanced Optimization Scheme Based on Gradient Descent Methods for Machine Learning. *Symmetry* **2019**, *11*, 942. [\[CrossRef\]](#)
39. Secui, A.; Sirbu, M.D.; Dascalu, M.; Crossley, S.; Ruseti, S.; Trausan-Matu, S. Expressing Sentiments in Game Reviews. In Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2016), Varna, Bulgaria, 7–10 September 2016; pp. 352–355. [35. \[CrossRef\]](#)
40. Ruseti, S.; Sirbu, M.D.; Calin, M.A.; Dascalu, M.; Trausan-Matu, S.; Militaru, G. Comprehensive Exploration of Game Reviews Extraction and Opinion Mining Using NLP Techniques. In *Advances in Intelligent Systems and Computing*; Springer: Singapore, 2019; pp. 323–331. [27. \[CrossRef\]](#)
41. Linzen, T. Issues in evaluating semantic spaces using word analogies. In Proceedings of the Workshop on Evaluating Vector-Space Representations for NLP, Berlin, Germany, 7–12 August 2016; pp. 13–18. [\[CrossRef\]](#)
42. Honnibal, M.; Montani, I. spaCy 3: Industrial-Strength Natural Language Processing. 2020. Available online: <https://spacy.io/> (accessed on 13 October 2021).
43. Řehůřek, R.; Sojka, P. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 22 May 2010; pp. 45–50.
44. Kula, M. glove-python. 2020. Available online: <https://github.com/maciejkula/glove-python> (accessed on 13 October 2021).
45. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
46. Chollet, F. Keras. 2015. Available online: <https://keras.io> (accessed on 8 October 2021).
47. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: <https://www.tensorflow.org/> (accessed on 13 October 2021).
48. Sirbu, D.; Secui, A.; Dascalu, M.; Crossley, S.A.; Ruseti, S.; Trausan-Matu, S. Extracting Gamers’ Opinions from Reviews. In Proceedings of the 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 24–27 September 2016; doi:10.1109/SYNASC.2016.044. [\[CrossRef\]](#)
49. Dingwall, N.; Potts, C. Mittens: An Extension of GloVe for Learning Domain-Specialized Representations. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, New Orleans, LA, USA, 1–6 June 2018; pp. 212–217. [\[CrossRef\]](#)
50. Piktus, A.; Edizel, N.B.; Bojanowski, P.; Grave, E.; Ferreira, R.; Silvestri, F. Misspelling Oblivious Word Embeddings. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, Minneapolis, MN, USA, 2–7 June 2019; pp. 3226–3234. [\[CrossRef\]](#)