Deep Learning Adaptation with Word Embeddings for Sentiment Analysis on Online Course Reviews

Danilo Dessí[1⋆][0000−0003−3843−3285], Mauro Dragoni[2][0000−0003−0380−6571], Gianni Fenu[1][0000−0003−4668−2476], Mirko Marras[1][0000−0003−1989−6057], and Diego Reforgiato Recupero[1][0000−0001−8646−6183]

[1] Department of Mathematics and Computer Science, University of Cagliari, Via Ospedale 72, 09124, Cagliari (Italy)
{danilo_dessi, fenu, mirko.marras, diego.reforgiato}@unica.it
[2] Fondazione Bruno Kessler, Trento, (Italy)
dragoni@fbk.eu

Abstract. Online educational platforms are enabling learners to consume a great variety of content and share opinions on their learning experience. The analysis of the sentiment behind such a collective intelligence represents a key element for supporting both instructors and learning institutions on shaping the offered educational experience. Combining Word Embedding representations and Deep Learning architectures has made possible to design Sentiment Analysis systems able to accurately measure the text polarity on several contexts. However, the application of such representations and architectures on educational data still appears limited. Therefore, considering the over-sensitiveness of the emerging models to the context where the training data is collected, conducting adaptation processes that target the e-learning context becomes crucial to unlock the full potential of a model. In this chapter, we describe a Deep Learning approach that, starting from Word Embedding representations, measures the sentiment polarity of textual reviews posted by learners after attending online courses. Then, with a set of experiments, we demonstrate how Word Embeddings trained on smaller e-learning-specific resources are more effective with respect to those trained on bigger general-purpose resources. Moreover, we show the benefits achieved by combining Word Embeddings representations with Deep Learning architectures instead of common Machine Learning models. We expect that this chapter will help e-learning stakeholders to get a clear view and shape the future research on this field.

Keywords: Big Data · Deep Learning · E-learning · Online Education · Sentiment Analysis · Word Embedding · Domain Adaptation.

⋆ Corresponding author email: danilo_dessi@unica.it Phone number: +39 070 675 8756

1 Introduction

The advent of Social Web has enabled the development and the sharing of experiences among people around the world. Individuals use online social platforms to express opinions about products and/or services in a wide range of domains, influencing the point of view and the behavior of their peers. Such user-generated data, which generally come in form of text (e.g., reviews, tweets, wikis, blogs), is often characterized by a positive or negative polarity according to the satisfaction of people who write the content. Understanding individual's satisfaction is a key element for businesses, policy makers, organizations and social institutions to hear and act on the voice of people. The automatic investigation performed on top of person's opinions in order to detect subjective information usually relies on Sentiment Analysis (SA). Techniques and systems in this field aim to identify, extract and classify emotions and sentiments by combining Natural Language Processing (NLP), Text Mining and Computational Linguistics [18]. Exploiting this artificial intelligence makes possible to replace or complement common practices (e.g., focus groups or surveys) for uncovering opinions, but still presents challenges because of large sources and context dependencies of data.

SA approaches can be classified in supervised and unsupervised. Supervised approaches require a training dataset annotated with numerical values left by users or inferred from the content in the text (e.g., emoticons), and leverage it to build a classifier which predicts a sentiment score for unseen data. Common supervised pipelines first require the extraction of features from the input text, such as terms frequencies, parts of speech, emotional words. Such features are then fed into an algorithm which characterizes each input text with a positive or negative polarity, i.e. sentiment detection. On the other hand, unsupervised approaches rely on lexicons associated with sentiment scores in order to model the sentiment polarity of a given text. While both types of approaches have been shown feasible on the sentiment detection task, they tend to suffer from oversensitiveness to the context where the training data is collected. This results in lower performance when applied to other contexts. Hence, understanding how SA techniques work in emerging areas, such as online education, becomes crucial.

Online educational platforms deployed at large scale, such as Coursera[3] and Udemy[4], are earning more and more attention as social spaces where students can discover and consume a great variety of contents about many topics, and share opinions regarding their educational experience [12]. Such collective intelligence might be useful for various stakeholders, including peers who are planning to attend a given course, instructors who are interested in improving their teaching practices and increasing students' satisfaction, and providers who can get benefits from the feedback left by users to refine tools and services in the platform itself. With this in mind, these platforms can be envisioned as dedicated social media where discussions are limited to specific topics concerning course content quality, teachers' skills, and so on [7]. SA approaches on students'

[3] https://www.coursera.org/
[4] https://www.udemy.com/

opinions have recently started to receive the attention of the involved stakeholders [19] and their design and development is still an open challenge.

The most prominent SA solutions leverage Word Embeddings, i.e., distributed representations that model words properties in vectors of real numbers which capture syntactic features and semantic word relationships. These resources has been shown useful in NLP tasks, like Part-Of-Speech (POS) tagging [26] and Word Analogy [29], and they have been also exploited for SA [23, 20, 15, 25, 41]. The generation of Word Embeddings is based on distributional co-occurrences of adjacent words able to model words meanings that are not visible from their surface. This exploits the fact that words with a similar meaning tend to be connected by a given relation. For instance, the verbs *utilize* and *use*, which are synonym although syntactically different, present similar sets of co-occurring words and can be considered similar, while a third verb, such as *play*, has different co-occurrences and should be considered different from both *utilize* and *use*. The literature acknowledges that Word Embeddings generated from general-purpose datasets, independently from any specific domain, under-perform the ones built on top of texts coming from the target context of the SA task [17]. This happens because context-trained Word Embeddings may capture specific patterns from the target context, while generic-trained ones might have learned patterns acting as noise for the target context. Hence, training Word Embeddings which better fit the e-learning context is crucial to achieve high SA performance.

Deep Learning (DL) has emerged as subclass of the Machine Learning (ML) area, where various neural network approaches are combined together for pattern classification and regression tasks. It usually employs multiple layers able to learn complex data representation, increasingly higher level features, and to correctly classify or measure properties held by data. The success of DL is due to the new advances in the ML field as well as to the ever more increasing computational abilities of computers through the use of Graphical Processing Units (GPUs) [10]. DL has shown improvements in addressing SA tasks [2, 23, 20]. Using DL models powered by Word Embeddings trained on the e-learning context can enable improving the effectiveness of the dedicated SA systems.

In this chapter, we first discuss the state-of-the-art literature on DL and Word Embeddings for SA (Section 2). We provide a high-level description of the existing Word Embeddings generation algorithms (Section 3) and common DL layers and networks (Section 4). Then, we propose a DL model, which is an extension of our preliminary work in [11], trained on Word Embedding representations coming from the e-learning context and able to predict a sentiment score for reviews posted by learners (Section 5). We also report its experimental evaluation on a large-scale dataset of online course reviews (Section 6). We show how Word Embeddings trained on smaller context-specific textual resources are more effective with respect to those trained on bigger general-purpose resources. Moreover, we highlight the benefits derived from the combination of Word Embeddings and DL instead of common ML approaches. Finally, conclusions, open challenges and future directions are discussed (Section 7). Code and models

accompanying this chapter would support data scientists and practitioners in developing next-generation sentiment-aware e-learning platforms[5].

2 State of the Art

In this section, we discuss the literature on DL and SA with a particular focus on the E-learning domain. The reader notices that we discuss separately the use of DL methods and Word Embeddings although they might be adopted together.

2.1 Sentiment Analysis in E-learning Systems

E-learning domain has recently gained the attention of SA in order to get knowledge from new dynamics that E-learning platforms allow to. By leveraging students' emotions, it might be possible contributing to increase the students' motivation and improve learning processes. More specifically, the study of sentiments in a E-learning platform can contribute to learning and teaching evaluation, investigate how technology can influence students' learning process, evaluate the use of E-learning tools, and improve learning content recommendations [36].

The measurement of sentiments and emotions in such platforms should be as less invasive as possible for not disturbing the learning process and not influencing the overall opinion. The adoption of textual reviews left by students' for analyzing sentiments and emotions is one of the less invasive techniques, because data collection and analysis are completely transparent for students. In literature, various scenarios where SA was used to study learning aspects using textual reviews can be found. For instance, one work embraces the use of Sentiment Analysis to mine students' interactions in collaborative tools, guaranteeing the students' communication privacy in their opinions [9]. Another relevant area that exploited SA was the teachers' assessment. For example, the authors in [19] adopted a Support Vector Machine to evaluate the teachers' performance using 1040 comments of systems engineering students as a dataset. The evaluation of the teaching-learning process was also object of study by means of SA in [8]. Its authors adopted comments posted by both students and teachers. Similarly, the authors in [37] studied text sentiment to build an adaptive learning environment with improved recommendations. For example, they described how to choose a learning activity for a student based on his/her goals and emotional profile.

The study of SA within the E-learning domain is still an open research area. With this paper, we aim to make a contribution in this direction by designing an effective DL model for improving sentiment detection in students' reviews.

2.2 Deep Learning for Sentiment Analysis

ML has been extensively adopted for SA tasks, using different types of algorithm and fitting various types of extracted features. For example, authors in [40] used

[5] Please find code and models at https://github.com/mirkomarras/dl-sentiment-coco.

Maximum Entropy (ME) and Naive Bayes (NB) algorithms, adopting syntactical and semantic patterns extracted from words on Twitter. Their method relies on the concept of contextual semantic i.e. considering word co-occurrences in order to extract the meaning of words [47]. In the evaluation on nine Twitter datasets they obtained better performance when the ML algorithms were trained with their method both at tweet and entity level. More recently, authors in [46] applied NB, ME, Stochastic Gradient Descent(SGD), and Support Vector Machine (SVM) algorithms to classify movies reviews in a binary classification problem (i.e. positive or negative evaluation of reviews). They showed that the use of a n-gram model to represent reviews with the above algorithms obtains higher levels of accuracy when the value n was small. Moreover, they showed that combining uni-gram, bi-gram, and tri-gram features enabled to enhance the accuracy of the method against the use of a single representation at once. ML methods rely on lexical syntactical features representations which are derived from text, not considering semantic relationships that can occur between words. Hence, in spite of feature engineering advancements, there has been a growth of techniques to infer semantics as DL that has emerged as an effective paradigm to automatically learn continuously information from text. The first DL approaches were studied at the begin of years 1990, but, due to high computational costs, they lost interest among scientific communities [49]. However, in the last years, more and more powerful computers and a huge availability of big data, DL approaches became state-of-the-art solutions across various domains.

SA domain also experienced the influence of the wide spread of DL approaches. For example, in [15] a Convolutional Neural Network (CNN) composed by two layers was designed to capture features from character to sentence level. An ensemble DL method was proposed by [1], where various sentiment classifiers trained on different sets of features were combined. They performed experiments on six different datasets coming from Twitter and movies reviews. With their experiments, they improved the state-of-art against DL baselines. Another approach to combine various classifiers with DL ones was also proposed by authors in [32], where a SVM classifier was mounted on top of a 7-layer CNN in order to complement the characteristics of each other and obtain a more advanced classifier. With their variation they were able to obtain more than 3% of improvement compared to a classic DL model. To learn continuous representations of words for SA a combination of CNN with a Long-Short Term Memory (LSTM) was exploited by authors in [45]. They were able to assign fixed-length vectors to sentences of varying lengths, showing how DL approaches outperform common ML algorithms. Although the use of DL models have showed amazing improvements in many domains, they have not been deeply studied for applications in E-learning, which can have benefits for exploring users' opinions.

2.3 Word Embeddings for Sentiment Analysis

Word Embeddings have been successfully used in various domains, ranging from behavioural targeting [4] to SA. Within the latter, they have been widely employed for improving accuracy of baselines methods not using Word Embed-

dings. As traditional Word Embeddings methods do not usually take into account words distributions for a specific task, resulting representations might lose important information for a given task. In the context of SA, authors in [24] incorporated prior knowledge at both word and document level with the aim to investigate how contextual sentiment was influenced by each word. On the same direction, other researchers [43] employed sentiment of text for the generation of words embeddings. In particular, they joined context semantics and sentiment characteristics so that in the embedding model neighboring words have both a similar meaning and sentiment. The rationale behind that depends on the fact that many words with a similar context are usually mapped on similar vector representations even if they have an opposite sentiment polarity (e.g. *bad* and *good*). Similarly, authors in [50] augmented sentiment information into semantic word representations and extended Continuous Skip-gram model (Skip-gram), coming up with two sentiment word embedding models. The learned sentiment Word Embeddings were able to correctly represent sentiment and semantics. Furthermore, authors in [27] presented a model that uses a mix of unsupervised and supervised techniques to learn word vector representations, including semantic term-document features. The model showed performances higher than several ML methods adopted for sentiment detection. Focusing on Twitter sentiment classification, authors in [44] trained sentiment-sensitive words embeddings through the adoption of three neural networks designed to detect the sentiment polarity of texts. Their methods encoded sentiment information in the continuous representation of words, and experiments on a benchmark Twitter classification dataset in SemEval 2013 showed that it outperformed the competitors. Last but not least, authors in [39] described a procedure with Word Embeddings for the estimation of levels of negativity in a sample of 56,000 plenary speeches from the Austrian parliament. They found out that the different levels of negativity shown by speakers in different roles from government or opposition parties agree with expected patterns indicated by common sense hypotheses. Their results showed that the Word Embeddings approach offers a lot of potential for SA and automated text analysis in the social sciences.

Several challenges have been created to solve SA polarity detection task and several resulting winning systems employed Word Embeddings within their core. For example, the Semantic Sentiment Analysis challenge [33, 35, 16, 34], held within the ESWC conference, reached its fifth edition[6][6]. The 2018 edition included a polarity detection task where participants were asked to train their systems by using a combination of Word Embeddings already generated by the organizers. The aim was to both validate the quality of their systems (precision/recall analysis) and detect which combination of embeddings worked better. SemEval is a workshop on semantic evaluation that takes place each year and includes a set of tasks in NLP and Semantic Web (e.g., SA polarity detection). One participant of the SemEval-2018 edition targeted the task of irony detection in Twitter [48]. It employed a simple neural network architecture of Multilayer Perceptron with various types of input features, including

[6] http://www.maurodragoni.com/research/opinionmining/events/challenge-2018/

lexical, syntactic, semantic and polarity features. The proposed system used 300-dimensional pre-trained Word Embeddings from GloVe [31] to compute a tweet embedding as the average of the embeddings of words in the tweet. By applying latent semantic indexing and extracting tweet representation through the Brown clustering algorithm, it achieved high performance in both subtasks of binary and multi-class irony detection in tweets. It ranked third using the accuracy metric and fifth using the F_1 metric. Kaggle[7] is the world's largest community of data scientists and offers ML competitions, a public data platform, and a cloud-based workbench for data science. It hosts several challenges, and some were related to SA. For instance, the *Sentiment Analysis on Movie Reviews* challenge[8] asked participants to label the movie reviews collected in the Rotten Tomatoes dataset [30] on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. One recent challenge, namely *Bag of Words Meets Bags of Popcorn*[9], looked for DL models combined with Word Embeddings for polarity detection of movie reviews collected by authors in [27].

3 Word Embedding Representations for Text Mining

Before using words in a model, they should be encoded as numbers. For instance, a function can be used to map words to integers or to one-hot encode words. When applying such an encoding to words, sparse vectors of high dimensionality are commonly obtained. On large data sets, this could cause performance issues. Additionally, such encoding functions do not take into account the semantics of the words. On the other hand, Word Embeddings are dense vectors with lower dimensionality, and the semantic relationships between words are reflected in the distance and direction of the vectors. Each word is positioned into a multi-dimensional space whose dimensions can be empirically chosen. The vector values for a word represent its position in this embedding space. Synonyms are found close to each other while words with opposite meanings have a large distance between them. In this section, we introduce the most representative and recent word embedding generator algorithms, highlighting their pros and cons.

3.1 Word2Vec

The *Word2Vec* word embedding generator [28] aims to detect the meaning and semantic relations between words by exploiting the co-occurrence of words in documents belonging to a given corpus. The core idea is to capture the context of words, using ML approaches such as Recurrent or Deep Neural Networks. In order to eliminate noise, *Word2Vec* operates on a corpus of sentences by constructing a vocabulary based on the words that appear in the corpus more often than a user-defined threshold. Then, it trains either the Continuous Bag-Of-Words (CBOW) or the Skip-gram algorithm on the input documents to learn

[7] https://www.kaggle.com/
[8] https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
[9] https://www.kaggle.com/c/word2vec-nlp-tutorial

the word vector representations. In this chapter, we will consider the Skip-gram algorithm since it works well with small amount of training data, which is often the case, and represents well even rare words or phrases.

3.2 GloVe

The *GloVe* [31] word embedding generator is a unsupervised learning algorithm developed by Stanford. It creates word embeddings by aggregating global word-word co-occurrence matrices from a corpus. The resulting embeddings show interesting linear substructures of the word in the vector space. More precisely, the algorithm consists of collecting word co-occurrence statistics in a form of word co-occurrence matrix. Each element of this matrix represents how often the word i appears in context of word j. The corpus is scanned in the following manner: for each term, it looks for context terms within some area defined by a *windowsize* before the term and a *windowsize* after the term, and it gives less weight for more distant words.

3.3 FastText

The *FastText* [22] word embedding generator is an algorithm for learning word representations. It differs form the previous ones in the sense that word vectors as the ones learned in *Word2Vec* treat every single word as the smallest unit whose vector representation is to be found, while *FastText* assumes a word to be formed by n-grams of character. This new representation of a word is helpful to find the vector representation for rare words. Since rare words could still be broken into character n-grams, they could share these n-grams with the common words. This can help to manage vector representations for words not present in the dictionary since they can also be broken down into character n-grams. Character n-grams embeddings tend to perform superior to *Word2Vec* and *GloVe* on smaller datasets [3].

3.4 Intel

Intel proposes to improve the data structures in *Word2Vec* through the use of mini-batching and negative sample sharing, allowing to solve the neural word embedding generation problem using matrix multiply operations [21]. They explored different techniques to distribute *Word2Vec* computation across nodes in a cluster, and demonstrate strong scalability. Their algorithm is suitable for modern multi-core/many-core architectures and allows scaling up the computation near linearly across cores and nodes, and processing millions of words per second. *Intel* embeddings generally differ from *Word2Vec* embeddings since the number of updates of the model is different across these two implementations, and the convergence is not equal for the same the number of epochs.

4 Deep Learning Components for Text Mining

DL has recently emerged as a new area within ML research. It embraces information processing methods consisting of a sequence of complex non-linear models. Each model forms a layer that independently processes data. The output of a layer is fed as an input to the subsequent layer in the sequence until the final output is obtained. In this section, we provide a high-level overview of the most popular layers and networks used or combined for mining texts and, thus, useful for conducting SA.

4.1 Feed-forward Neural Network (FNN)

Feed-forward Neural Networks (FNN) were one of the first and simplest components applied to learn from data using DL paradigms. One or more levels of nodes, often called perceptrons, are randomly joined by weighted connections in a many-to-many fashion. These networks were historically thought in order to simulate a biological model where nodes are neurons and links between them represent synapses. For this reason, they are also called Multi-Layer Perceptron (MLP) networks. On the basis of the input values fed into the network, nodes of a certain level can be activated and their signal is broadcasted to the subsequent level. In order to activate nodes of a subsequent level, the signal generated at a given level is weighted and must be greater than a given threshold. Weights are generally initialized with random values and adjusted during training in order to minimize a predefined objective function. This family of networks has been proved to be useful for pattern classification, but less suitable for labelling sequences since it does not take into account the sequence of input data. A simple three-layer Feed-forward Neural Network is showed in Figure 1.

The sample FNN as it is accepts three-dimensional inputs and returns two-dimensional outputs. Each node of a given level is connected to nodes of the subsequent layer. The input data is fed into the network by means of Layer 1, which acts as Input Layer, and then sent to the first hidden layer, i.e., Layer 2. The output of this layer is finally propagated to Layer 3, which represents the Output Layer. The action to move data from a layer to another by activating or not the corresponding nodes is generally called *forward pass* of the network.

4.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) are tailored for processing data as a sequence. In contrast to FNNs which commonly pass the input data directly from input to output nodes, RNNs have cyclic or recurrent connections among nodes of distinct levels. This makes possible to model the output of the network by taking into account the entire history of the received input data. Recurrent connections connect past data with the one that is currently being processed, simulating a state memory. The forward pass is similar to the one in FNNs. The unique difference is that the activation of a node depends on both the current input and the previous status of the hidden layers. The text is fed into the network by

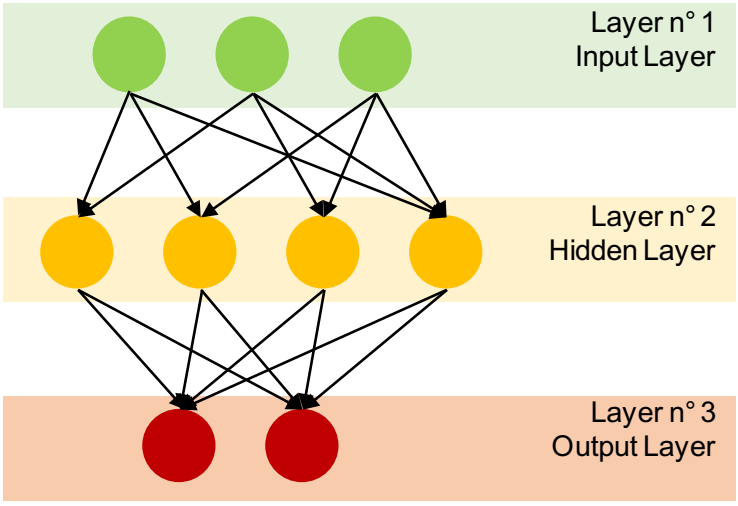

Fig. 1. An example of a Feed-forward Network composed by three layers.

means of vector representations that are recurrently processed. This means that RNNs view a sample text as an ordered sequence of word identifiers, differently from common FNNs working on hand-crafted inputs, e.g., Bag of Words (BOW).

In a wide range of applications, data can present patterns from the past to the future and vice versa. For instance, for classifying the sections of a given story, it could be useful to have access to both future and past sections. However, the future content of a text is ignored by common FNNs and RNNs, as they work sequentially. Bidirectional RNNs (BiRNNs) let the network, at a given point in time, to take information from both earlier and later data in the sequence, going beyond the exposed limitations. The idea behind this kind of networks consists of presenting the training data forwards and backwards by two hidden RNNs which are then combined into a common output layer. This strategy makes possible to find patterns that can be learned from both past and future history of data.

4.3 Long Short-Term Memory (LSTM) Network

Long Short-Term Memory (LSTM) networks are an RNN extension designed to work on sequence problems and that has achieved state-of-the-art results on challenging prediction tasks. LSTM networks employ recurrent connections and add memory blocks in their recurrent hidden layers. These memory blocks save the current temporal state during training and make possible to learn temporal observations hidden in the input data. The fact of using connections as a memory implies that the output of a LSTM network depends on the entire history of the training data, not only on the current input sample. Moreover, using memory blocks allows to relate the current data that is being processed with the data

processed long before, solving the problem experienced by common RNNs. For this reason, LSTM networks have had a positive impact on sequence prediction tasks. As stated for RNN, a bidirectional layer using two hidden LSTMs can be leveraged to process data both forward and backward.

4.4 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) typically perform filtering operations on the input nodes of a layer, abstracting and selecting only meaningful input features. When CNNs are trained, the weights of links between nodes acting as filter are defined. Such networks have been historically applied in the computer vision field and, hence, are not directly applicable on texts as they are. To overcome this limitation, a text must be converted into a vector representation, and the convolutional filters are applied on this representation. A convolutional filter is composed by a kernel that slides on the vector representation and repeats the same function on each element until all the vectors have been covered. In text mining, CNNs can be useful to detect the words characterizing a classes.

4.5 Normalization Layer (NL)

During training, the output of a given layer is affected by parameters and processes used in previous layers. Small changes in the parameters set for a layer can hence be propagated as the network becomes deeper, resulting in large changes in the final output, i.e., the output of the last layer. Considering that the parameters are continuously changed to better fit the prediction task and that the data distribution changes across levels, the variations within the parameters can negatively influence the training, making it computationally expensive. To shape input data with a standard distribution and improve training performance, some Normalization Layers (NL) can be introduced. One of the most common normalization layers is represented by *Batch Normalization*. This layer makes possible to reduce the dependence of the optimization parameters from the input values, avoiding over-fitting and making the training process more stable.

4.6 Attention Layer (AL)

Attention Layers (ALs) are often adopted before the last fully-connected layers of a model. Attention mechanisms in neural networks serve to orient perception as well as memory access. Attention layers filters the perceptions that can be stored in memory, and filters them again on a second pass when they need to be retrieved from memory. Neural networks can allocate attention, and they can learn how to do so, by adjusting the weights they assign to various inputs. This makes possible to solve traditional limits in various NLP tasks. For instance, traditional word vectors presume that a words meaning is relatively stable across sentences. However, there could be massive differences in meaning for a single word: e.g. lit (an adjective that describes something burning) and lit (an abbreviation for literature); or get (a verb for obtaining) and get (an animals offspring).

ALs can capture the shades of meaning for a given word that only emerge due to its situation in a passage and its inter-relations with other words. Moreover, ALs learn how to relate an input sequence to the output of the model in order to pay selective attention on more relevant input features. For example, in order to reflect the relation between inputs and outputs, an Attention layer may compute an arithmetic mean of results of various layers according to a certain relevance.

4.7 Other Layers

There are also other layers that can be leveraged in order to fine-tune the performance of a model. The most representative ones are described below.
Embedding Layer. An Embedding layer turns positive integers (indexes) into dense vectors of fixed size chosen from a pre-initialized matrix. For an integer-encoded text, the dense vector corresponding to those integers are selected.
Noise Layer. A Noise layer is usually employed to avoid model over-fitting. It consists in modifying a fraction of input of layers, adding and subtracting some values following a predefined distribution (e.g., Gaussian).
Dropout Layer. A Dropout layer may be seen as a particular type of a noise layer. It assigns the value 0 to a randomly chosen fraction of its input data. The name *Dropout* comes from the action to dropping some units of the input.
Dense Layer. A Dense layer is a densely-connected layer that is used to map large unit inputs in a few unit results. For example, it may be used to define the number of classes that a model returns, mapping hundred and thousand nodes in a few number of classes.

5 Our Sentiment Predictor for E-Learning Reviews

This section serves as guide on designing sentiment prediction models for educational reviews, and presents practical information on how to implement such systems. The main components of the proposed solution (Figure 2) and the benefits of the designing choices for each of these components will be described. This helps readers have a broader view of difficulties and solutions behind sentiment prediction models, and make appropriate decisions during their design.

5.1 Review Splitting

The *review splitting* step serves to define the various input dataset splits while developing the sentiment prediction model. Firstly, we consider the input dataset as a set D of N reviews organized as follows:

$$D = \{(text_1, score_1), ..., (text_N, score_N)\} \tag{1}$$

where $text_i$ is a textual review and $score_i$ is an integer rating belonging to the set $C = \{score_1, ..., score_M\}$.

During this step, we thus need to split input data D in three subsets, each for a specific phase of the development:

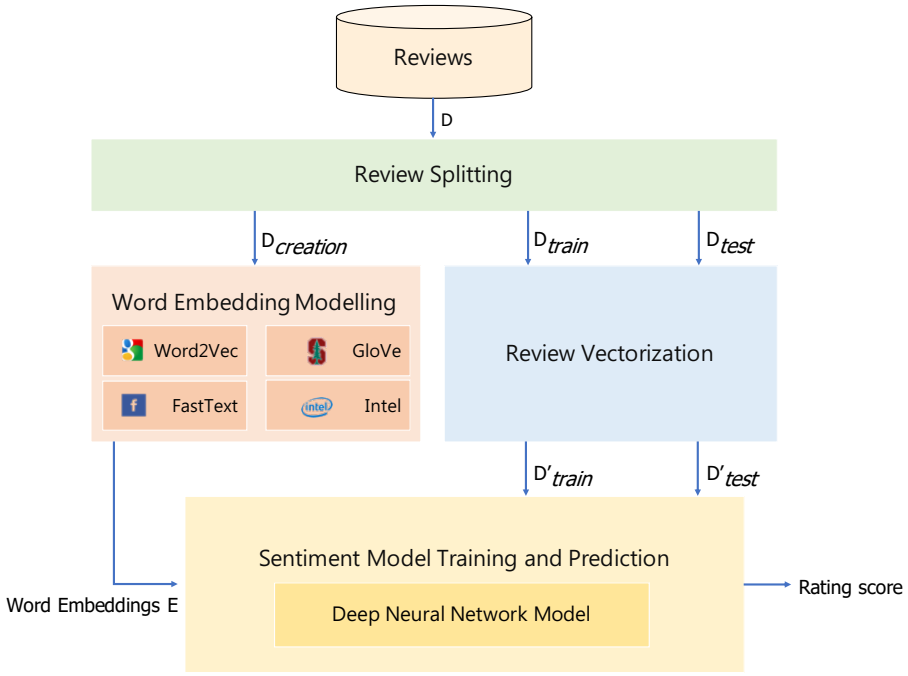

Fig. 2. The base components of our sentiment prediction model.

1. $D_{creation}$: the sample of data used to create word embeddings.
2. D_{train}: the sample of data used to fit the model (i.e., weights and biases).
3. D_{test}: the sample of data used as the gold standard to evaluate the model.

In order to do this, we set up two split ratios and we assign the text-score pairs in D to the different subsets $D_{creation}, D_{train}, D_{test}$ according to them:

1. $s_{creation} \in [0, 1]$: the percentage of reviews for each class $c \in C$ that are randomly chosen from the set D to create word embeddings, yielding $D_{creation}$.
2. $s_{training} \in [0, 1]$: the percentage of reviews for each class $c \in C$ that are randomly chosen from $D \setminus D_{creation}$ to train the model, yielding D_{train}.

The remaining reviews represent D_{test}. The overall procedure ensures that the subsets are disjoint and their union covers the entire dataset D.

5.2 Word Embedding Modelling

The state-of-the-art method to model a word with a vector is using word embeddings; it is common to see word embeddings that are 256-dimensional, 512-dimensional, or 1,024-dimensional when dealing with very large vocabularies. There are two ways to generate and leverage word embeddings:

1. Learn word embeddings jointly with the same context we are interested in by starting with random word vectors and, then, learning word vectors along the process, iteratively.
2. Load into the sentiment prediction model the word embeddings pre-computed using a different ML task than the one we are interested in. If the amount of training data in D_{train} is small, this is the common solution.

To the best of our knowledge, no word embedding database specifically targets the e-learning context. Therefore, this step goes through the first most general solution of learning word embeddings from scratch, while we also use word embedding pre-computed on other contexts for comparison along the chapter.

In order to generate word embeddings from scratch, the subset of pre-processed reviews $D_{creation}$ was employed. We concatenated them into a large corpus and this corpus was fed into a given word embedding generation algorithm selected among the following ones: *Word2Vec*, *GloVe*, *FastText*, or *Intel*. Each of them outputs a set of feature vectors E for words in that corpus. The feature values are non-negative real numbers. For each distinct word w in the vocabulary in $D_{creation}$, there exists a corresponding feature vector $e \in E$ which represents the word embedding for that word. All the feature vectors share the same size. The size of the resulting word embeddings and of the window where word embeddings generator algorithms look at contextual words can be arbitrarily selected.

5.3 Review Vectorization

The *review vectorization* is the process of transforming each review into a numeric sequence. This can be done in multiple ways (e.g., segment text into words and transform each word into a vector, segment text into characters and transform each character into a vector, extract n-grams of words or characters, and transform each n-gram into a vector). The different units into which the text is broken (words, characters, or n-grams) are called *tokens*, and breaking text into such tokens is called *tokenization*. The process consists of applying some tokenization schemes and then associating numeric vectors with the generated tokens. These vectors, packed into sequences, are needed for manipulating text during sentiment model training and inference.

In order to be treated by machines, we need to turn the datasets D_{train} and D_{test} into a set of integer-encoded pre-processed reviews defines as follows:

$$D'_{train} = \{(text'_1, score_1), ..., (text'_K, score_K)\} \forall (text_i, score_i) \in D_{train} \quad (2)$$

$$D'_{test} = \{(text'_1, score_1), ..., (text'_J, score_J)\} \forall (text_i, score_i) \in D_{test} \quad (3)$$

where each pair $(text'_i, score_i)$ includes an integer encoding of the text comment $text_i$ and the original rating $score_i$ from D_{train} and D_{test}, respectively.

The process for generating D'_{train} and D'_{test} works as follows. Each word has a unique associated integer value chosen from a range going from 0 to $|V| - 1$, where V is the vocabulary of words in D. For each input review $(text_i, score_i)$, we build an integer-encoded vector $text'_i$ from $text_i$, where an integer value at position j in $text'_i$ represents the mapped value for word w for that position in $text_i$. The sets D'_{train} and D'_{test} are thus vectorized.

5.4 Sentiment Model Definition

This step is necessary for defining the architecture of the deep neural network which takes pairs of integer-encoded texts and sentiment scores, maps such texts into word embeddings, and tries to predict the sentiment score from them.

The proposed architecture tailored for sentiment score prediction is shown in Fig. 3. Given that our training process requires running the network on a rather large corpus, our design choices are mainly driven by the computational efficiency of the network. Hence, differently from [2], which presents an architecture with two Bidirectional LSTM layers, we adopt a single Bidirectional LSTM layer architecture. Moreover, we configure the last layer to return a single continuous value, i.e., the predicted sentiment score. Therefore, our network is composed by an Embedding layer followed by a Bidirectional LSTM layer, a Neural Attention mechanism, and a Dense layer. Each layer works as follows:

1. **Embedding Layer** takes a two-dimensional tensor of shape (N, M) as input, where N represents the number of integer-encoded text comment samples, while M the maximum sequence length of such samples. Each entry is a sequence of integers passed by the Input Layer. The output of the Embedding layer is a two-dimensional vector with one embedding for each word w in the input sequence of words of each text comment t. Before receiving data, the Embedding Layer loads the pre-trained word embeddings computed during the previous step as weights. Such weights are frozen, so that the pre-trained parts are not updated during training and testing to avoid forgetting what they already know.
2. **Bidirectional LSTM Layer** is an extension of the traditional LSTM that generally improves model performance on sequence classification problems. It trains two LSTM instead of just one: the first is trained on the input sequence as it is and the second on a reversed copy of the input sequence. The forward and backward outputs are then concatenated before being passed on to the next layer, and this is the method often used in studies of bidirectional LSTM. Through this layer, the model is able to analyze a reviews as a whole, binding first and last words coming up with a more precise score. Moreover, exploiting the bidirectional version of a LSTM, the model is able to get patterns that depend on the learners' writing style.
3. **Attention Layer** enables the network referring back to the input sequence, instead of forcing it to encode all the information forward into one fixed-length vector. It takes n arguments $y_1, ..., y_n$ and a context c. It returns a

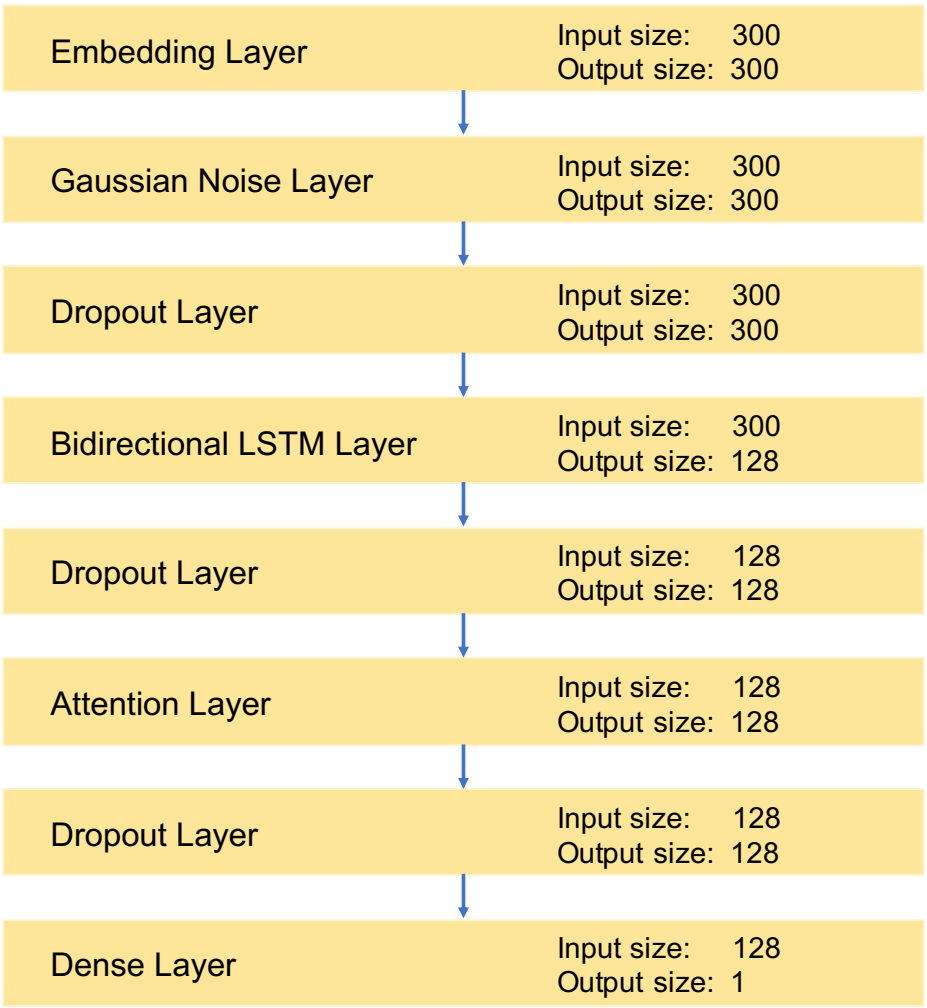

Fig. 3. The proposed deep learning model for sentiment score regression designed to leverage 300-dimensional input text sequences.

vector z which is supposed to be the summary of the y_i, focusing on information linked to the context c. More specifically, in our model it returns a weighted arithmetic mean of the y_i, and the weights are chosen according to the relevance of each y_i given the context c. This step can improve performance, detecting which words more influence the sentiment assignments.

4. **Dense Layer** is a regular densely-connected layer implementing a function $output = activation(dot(input, kernel) + bias)$ where $activation$ is the element-wise activation function, while $kernel$ and $bias$ are a weights matrix and a bias vector created by the layer, respectively. The layer uses a linear activation $a(x) = x$ and provides a single output unit representing the sentiment score.

To mitigate the overfitting, the network augments the cost function within layers with l_2-norm regularization terms for the parameters of the network. It also uses Gaussian Noise and Dropout layers to prevent feature co-adaptation.

5.5 Sentiment Model Training and prediction

The fresh instance of the sentiment model takes a set of neural Word Embeddings E together with a set of pre-processed reviews D'_{train}, as input. With these embeddings and reviews, the component trains the deep neural network. As an objective, the network measures the MSE (Mean Squared Error) of the predicted sentiment score against the gold standard value for each input sequence. Parameters are optimized using RMSProp (Root Mean Square Propagation) [38] with $learning\_rate = 0.001$. The network was configured for training on batches of size 128 along 20 epochs, shuffling batches between consecutive epochs. The trained deep neural network takes a set of unseen reviews D'_{test} and returns the sentiment score $score'$ predicted for that text comment $text'$, as output.

6 Experimental Evaluation

6.1 Dataset

The dataset used for our experiments is COCO [13], which includes data collected from one of the most popular online course platforms. It contains more than 43K courses distributed in 35 languages, involving over 16K instructors and 2,5M learners who provided 4.5M reviews about online courses.

In our experiments, we considered only reviews with non-empty English text comments. They are 1.396.312 in COCO. Each review includes a rating ranging from 0.5 to 5 with step of 0.5. Considering that our approach supports only integer ratings, we mapped COCO ratings on a scale from 0 to 9 with steps of 1. The dataset D included 1.396.312 reviews and the split ratios were $s_{creation} = s_{train} = 0.90$. Those values were selected since we wanted to keep both training and testing sets with balanced rating distributions. Moreover, we performed 10-fold stratified cross validation. Hence, 1.396.312 - 6.500 * 10 reviews were put in $D_{creation}$ to create embeddings, while 5.850 * 10 were put in D_{train} for training the model and 650 * 10 were put in D_{test} for testing it during each fold.

6.2 Baselines

We experimented and compared our deep learning approach with the following common ML algorithms:

- *Support Vector Machine.* Support Vector Machine (SVM) algorithm works by defining boundaries through hyperplanes in order to separate a class from the others. The aim of this algorithm is building hyperplanes among data samples in such a way that the separation between classes is as large as possible. The algorithm takes labeled pairs (x_i, y_i) where x_i is a vector representation of input data, and y_i is a numerical label. The algorithm then applies an optimization function in order to separate classes. When it is used for textual input, it is common to transform the text input into vectors of numbers representing features. To name an example, authors in [14] used vectors of numbers for assigning words with syntactical and semantic features to apply a SVM classifier. In the regression variant of SVM, generally named SVR (Support Vector Regressor), the algorithm try to find hyperplanes that can predict the distribution of information.
- *Random Forests.* Random Forests (RF) is based on an ensemble of decision trees, where each tree is independently trained and votes for a class for the data presented as an input [5]. We use a Random Forest with 10 trees with depth 20. Essentially, each decision tree splits data into smaller data groups based on the features of the data until there are small enough sets of data that only have data points with the same label. These splits are chosen according to a purity measure and, at each node, the algorithm tries to maximize the gain on it. For our regression problem, we consider Mean Squared Error (MSE).
- *Feed-forward Neural Network.* We used a common Feed-forward Neural Network (FF) with 10 hidden layers, as described in Section 4.1.

We exploited the regression algorithm implementations available within the scikit-learn library[10]. To feed data into these baseline models, we compute the average of word embeddings for each review. More specifically, given a review r with terms $\{t_0, ..., t_{n-1}\}$, we took the associated word embeddings $\{w_0, ..., w_{n-1}\}$ and computed their average w, which is used to represent the review text.

6.3 Metrics

In order to evaluate the performance of our model, we measured the MSE (Mean Squared Error) and the MAE (Mean Absolute Error) scores. More precisely, MAE and MSE are defined as follows:

$$MAE(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} |y_i - \hat{y}_i| \tag{4}$$

[10] https://scikit-learn.org/stable/index.html

$$MSE(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2 \tag{5}$$

where y_i is a true target value, $\hat{y}_i$ is a predicted target value, and n is the number of samples for both (4) and (5). It follows that given two tests t_1 and t_2, t_1 is better than t_2 if its related MSE and MAE are lower. During the experiments, we maintained the proportion of the reviews for each of the 10 classes of the original dataset for both training and test sets [42].

6.4 Deep Neural Network Model Regressor Performance

Figure 4 reports the MAE of regressors used in our experiments. First of all, our results confirm that Neural Networks, both using a single Feed-forward layer and using our model, perform better than common ML algorithms, showing a lower error. Comparing the Feed-forward baseline with our Deep Neural Network model, there is a little error difference. It is possible to note that the combination *FF + FastText* obtains similar performances of both *DNNR + GloVe* and *DNNR + FastText*. The best performance was obtained by *DNNR + Word2Vec*. Similar considerations also apply when analyzing the MSE. In fact the *DNNR* model gets best performance as well. In contrast with the *MAE*, no baseline obtains performances similar to our model.

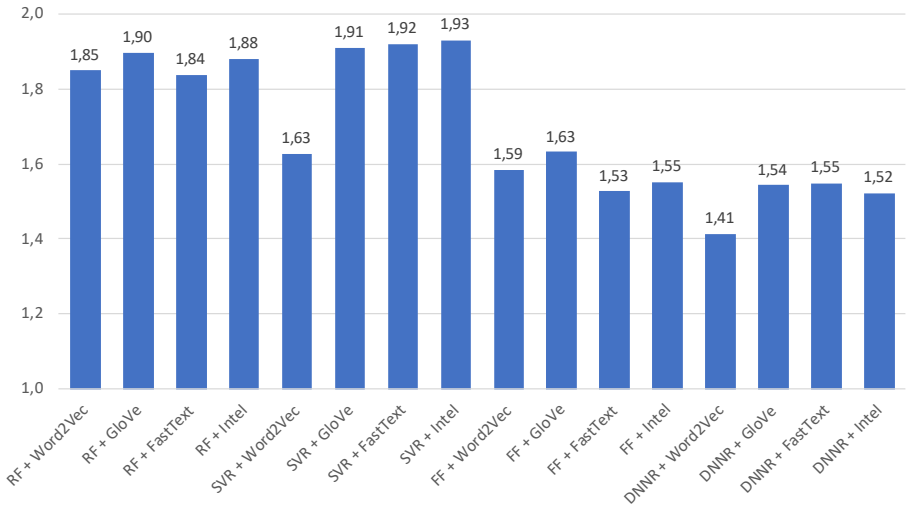

Fig. 4. Mean Absolute Error (MAE) of Experimented Regressors.

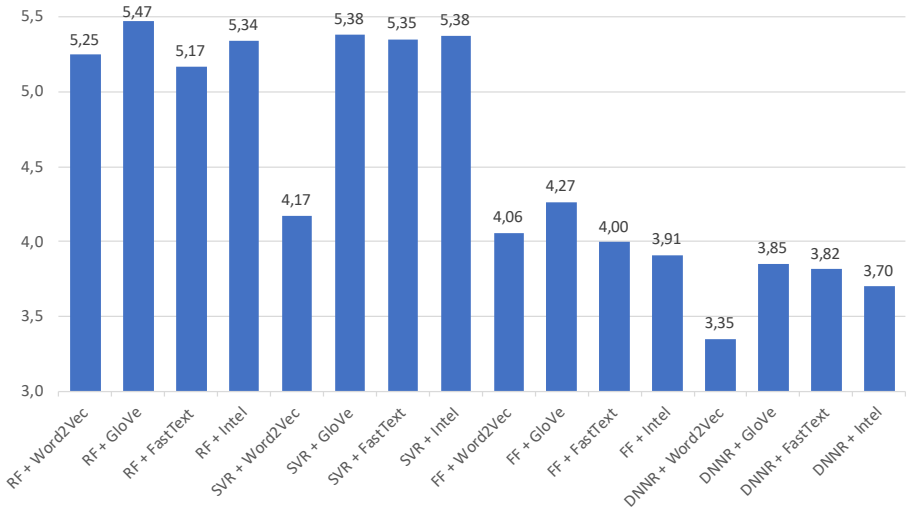

Fig. 5. Mean Square Error (MSE) of Experimented Regressors.

6.5 Contextual Word Embeddings Performance

This further experiment aims to show how the context-trained Word Embeddings we generated have advantage over reference generic-trained Word Embeddings, when they are fed into our deep neural network as frozen weights of the Embedding Layer. In order to evaluate the effectiveness of our approach, we performed experiments using embeddings of size 300 trained on COCO's online course reviews. We compared them against the following reference generic-trained Word Embeddings of size 300 commonly adopted in literature:

- The *Word2Vec*[11] Word Embeddings trained on a part of the Google News dataset including 100 billion words with a vocabulary of 3 million words.
- The *GloVe*[12] Word Embeddings trained on a Wikipedia dataset including one billion words with a vocabulary of 400 thousand words.
- The *FastText*[13] Word Embeddings trained on a Wikipedia dataset including four billion words with a vocabulary of 1 million thousand words.

Context-trained *Intel* word embeddings are compared with generic *Word2Vec* word embeddings because i) there are not public generic *Intel* word embeddings, and ii) the *Intel* algorithm is an evolution of *Word2Vec* algorithm.

Figure 6 shows that there is not a relevant difference between context-trained word and generic-trained embeddings when the MAE is used for the comparison.

[11] https://code.google.com/archive/p/word2vec/
[12] https://nlp.stanford.edu/projects/glove/
[13] https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.en.vec

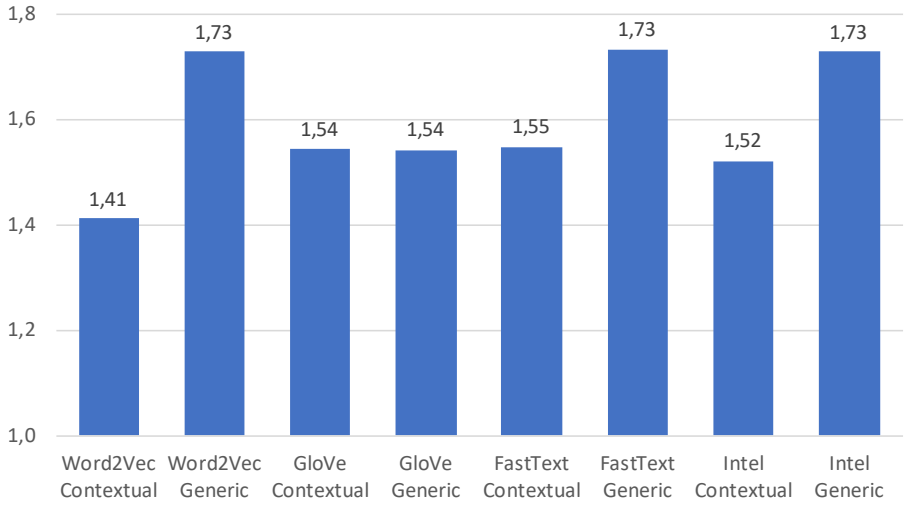

Fig. 6. Comparison between Contextual Word Embeddings and Generic Word Embeddings considering the Mean Absolute Error (MAE).

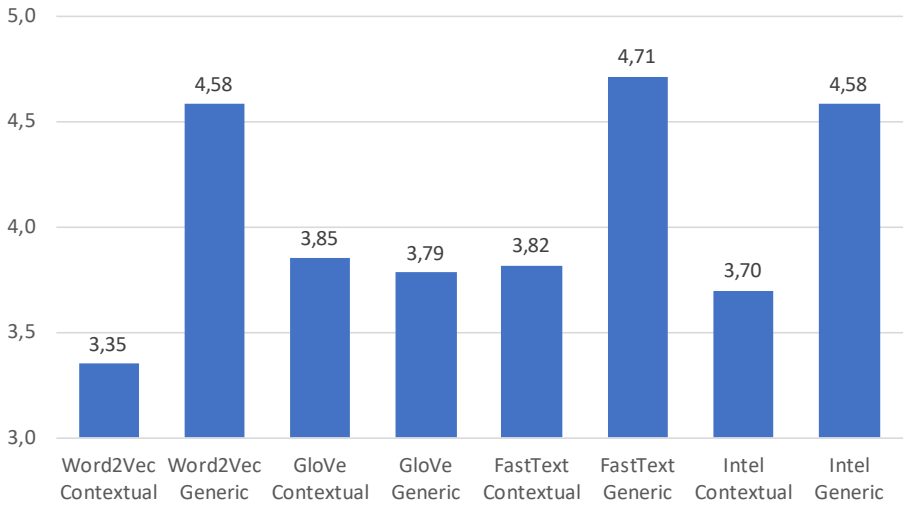

Fig. 7. Comparison between Contextual Word Embeddings and Generic Word Embeddings considering the Mean Square Error (MSE).

Nevertheless, it is worth underling how the type of embeddings enables to obtain better results in the E-learning domain. Context-trained *Word2Vec* embeddings show the lowest values of MAE compared to other embeddings types. In contrast, when the MSE is considered, context-trained embeddings perform better, as shown in Figure 7. In this case, context-trained embeddings have low values of MSE in almost all cases except for the *GloVe* Word Embeddings. The best performance was obtained by context-trained *Word2Vec* embeddings, proving that i) *Word2Vec* is the best algorithm to learn word representations from our dataset, and ii) context-trained Word Embeddings are able to capture specific patterns of the E-learning domain. This makes possible to adapt our DL model on the E-learning domain and improve the results in sentiment score prediction.

7 Conclusions, Open Challenges, and Future Directions

This chapter was structured around a case study on SA within the e-learning context. By combining state-of-the-art DL methodologies and word embedding text representations, we introduced and described a deep neural network aimed to predict a sentiment score for text reviews left by learners after attending online courses, as a targeted educational context.

The proposed chapter guides the readers on how to address this task by providing an overview of the common building blocks of a SA system, from input text representations to neural network components, followed by a recipe which combines them into a model able to outperform common ML baselines. As most of the current approaches for SA are built on top of different word embedding representations, we showed how some types of word embeddings can better represent the semantics behind the e-learning context and help the model to well predict the sentiment score. Furthermore, considering that word embeddings tend to be sensitive to the context where they are trained in and that the current publicly-available word embeddings were trained on general-purpose resources, we proved that the use of word embeddings generated from e-learning resources enables the model to capture more peculiarities from this target context.

Research on SA has produced a variety of solid methods, but still poses some interesting challenges that require further investigation:

1. **Public Datasets and Models**. Most SA studies in education have still used rather small datasets which were not made public available and make difficult to train a neural network. As education is a very heterogeneous research field, differentiated into informal, non-formal and formal learning, a larger collection with more diverse datasets is needed. Furthermore, sharing code and pre-trained models has not been a common practice. Only few authors made their code available, while, for others, people need to re-implement it from scratch. More datasets and models should be shared.
2. **Text Representation Modeling**. Current approaches, such as *Word2Vec*, exhibit limits which can help us understand future trends. For instance, there is only one word embedding per word, i.e. word embeddings can only represent one vector for each word. Therefore, the term *"learned"* only had one

meaning for *"I have learned that information last week"* and *"The instructor was a very learned individual"*. Moreover, word embeddings are difficult to train on large datasets, and, to tailor them to another context, they should be trained from scratch. This requires large datasets on the target context and high storage and computational resources. Finally, word embeddings have been generally trained on a neural network with only few hidden layers, and this has limited the semantic power of the corresponding representations.

3. **Sentiment Prediction Model Design**. SA systems have traditionally used an FNN as the underlying architecture. However, its densely-connected layers have access only to the current input and have no memory of any other input that was already processed. Recent research showed that RNNs can provide state-of-the-art embeddings that address most of the shortcomings of previous approaches. Emerging systems were trained on a multilayer RNN and learned word embeddings from context, enabling it to store more than one vector per word based on the context it was used in. More advanced architectures need further investigation.
4. **Transfer Learning across Contexts**. Existing models tend to be sensitive to the context targeted by the underlying training data. This has favored the creation of semantic models that, after being trained with data from a given context, do not generalize well in other contexts. With the new availability of public datasets and pre-trained models, it will become easier to plug them into a task different from the one they were originally thought. Previously, performing SA required to train a model or use an API to get the sentiment predictions. By sharing more pre-trained models, cooperation between researchers in SA for education and researchers from other application areas can be promoted. Hence, people could build a new service on top of pre-trained models and quickly train them with small amounts of context-specific data.
5. **Model Explainablity and Interpretability**. Most ML and DL algorithms built into automation and artificial intelligence systems lack transparency, and may contain an imprint of the unconscious biases of the data and algorithms underlying them. Hence, it becomes important to understand how we can predict what is going to be predicted, given a change in input or algorithmic parameters. Moreover, it requires attention how the internal mechanics of the ML or DL system can be explained in human terms. As a context like education looks to deploy artificial intelligence and DL systems, understanding how an algorithm is actually working can help to better align the activities of data scientists and analysts with the key questions and needs of the involved stakeholders.
6. **Data and Algorithmic Bias Impact**. With the advent of ML and DL, addressing bias within education analytics will be a core priority due to several reasons. For instance, some biases can be introduced through the use of training data which is not an accurate sample of the target population or is influenced by socio-cultural stereotypes. Moreover, the methods used to collect or measure data and the algorithms leveraged for predicting sen-

timents can propagate biases. Future research should control these biases in the developed models, promoting fair, transparent, and accountable systems.

7. **Multi-Aspect Sentiment Modeling**. SA in education has focused on determining either the overall polarity (i.e., positive or negative) or the sentiment rating (e.g., one-to-five stars) of a review. However, only considering overall ratings does not allow to represent the multiple potential aspects on which an educational element can be reviewed (e.g., the course content, the instructor, and the platform). To get insightful knowledge on how people perceive each of them, more research taking into account these various, potentially related aspects discussed within a single review is needed.

We expect that the case study on SA within the E-learning context covered in this chapter will help researchers, developers and other interested people to get a clear view and shape the future research on this field.

Acknowledgments

Danilo Dessì and Mirko Marras acknowledge Sardinia Regional Government for the financial support of their PhD scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2014-2020, Axis III "Education and Training", Specific Goal 10.5).

The research leading to these results has received funding from the EU's Marie Curie training network PhilHumans - Personal Health Interfaces Leveraging HUman-MAchine Natural interactionS under grant agreement 812882.

Furthermore, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

References

1. Araque, O., Corcuera-Platas, I., Sanchez-Rada, J.F., Iglesias, C.A.: Enhancing deep learning sentiment analysis with ensemble techniques in social applications. Expert Systems with Applications **77**, 236–246 (2017)
2. Atzeni, M., Reforgiato, D.: Deep learning and sentiment analysis for human-robot interaction. In: Europ. Semantic Web Conference. pp. 14–18. Springer (2018)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017), https://transacl.org/ojs/index.php/tacl/article/view/999
4. Boratto, L., Carta, S., Fenu, G., Saia, R.: Using neural word embeddings to model user behavior and detect user segments. Knowledge-Based Systems **108**, 5–14 (2016). https://doi.org/10.1016/j.knosys.2016.05.002, cited By 10
5. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (Oct 2001). https://doi.org/10.1023/A:1010933404324, https://doi.org/10.1023/A:1010933404324
6. Buscaldi, D., Gangemi, A., Reforgiato Recupero, D.: Semantic Web Challenges: Fifth SemWebEval Challenge at ESWC 2018, Heraklion, Crete, Greece, June 3 - June 7, 2018, Revised Selected Papers. Springer Publishing Company, Incorporated, 3st edn. (2018)

7. Cela, K.L., Sicilia, M.Á., Sánchez, S.: Social network analysis in e-learning environments. Educational Psychology Review **27**(1), 219–246 (2015)
8. Chauhan, G.S., Agrawal, P., Meena, Y.K.: Aspect-based sentiment analysis of students feedback to improve teaching–learning process. In: Information and Communication Technology for Intelligent Systems, pp. 259–266. Springer (2019)
9. Clarizia, F., Colace, F., De Santo, M., Lombardi, M., Pascale, F., Pietrosanto, A.: E-learning and sentiment analysis: a case study. In: Proceedings of the 6th International Conference on Information and Education Technology. pp. 111–118. ACM (2018)
10. Deng, L.: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing **3** (2014)
11. Dessi, D., Dragoni, M., Fenu, G., Marras, M., Reforgiato Recupero, D.: Evaluating neural word embeddings created from online course reviews for sentiment analysis. In: The 34th ACM/SIGAPP Symposium on Applied Computing. pp. 2124–2127. SAC (2019)
12. Dessì, D., Fenu, G., Marras, M., Reforgiato Recupero, D.: Bridging learning analytics and cognitive computing for big data classification in micro-learning video collections. Computers in Human Behavior (2018)
13. Dessì, D., Fenu, G., Marras, M., Reforgiato Recupero, D.: Coco: Semantic-enriched collection of online courses at scale with experimental use cases. In: Trends and Advances in Infor. Systems and Technologies. pp. 1386–1396. Springer (2018)
14. Dessì, D., Fenu, G., Marras, M., Recupero, D.R.: Leveraging cognitive computing for multi-class classification of e-learning videos. In: European Semantic Web Conference. pp. 21–25. Springer (2017)
15. Dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 69–78 (2014)
16. Dragoni, M., Reforgiato Recupero, D.: Challenge on fine-grained sentiment analysis within eswc2016. In: Sack, H., Dietze, S., Tordai, A., Lange, C. (eds.) Semantic Web Challenges. pp. 79–94. Springer International Publishing, Cham (2016)
17. Dragoni, G., P.M.: A neural word embeddings approach for multi-domain sentiment analysis. IEEE Trans. Affect. Comput. **8**(4), 457–470 (2017)
18. Dridi, A., Reforgiato, D.: Leveraging semantics for sentiment polarity detection in social media. Int. Jour. of Machine Learning and Cybernetics (2017)
19. Esparza, G., de Luna, A., Zezzatti, A.O., Hernandez, A., Ponce, J., Álvarez, M., Cossio, E., de Jesus Nava, J.: A sentiment analysis model to analyze students reviews of teacher performance using support vector machines. In: Int. Symp. on Distributed Computing and Artificial Intelligence. pp. 157–164. Springer (2017)
20. Giatsoglou, M., Vozalis, M.G., Diamantaras, K., Vakali, A., Sarigiannidis, G., Chatzisavvas, K.: Sentiment analysis leveraging emotions and word embeddings. Expert Systems with Applications **69**, 214–224 (2017)
21. Ji, S., Satish, N., Li, S., Dubey, P.: Parallelizing word2vec in multi-core and many-core architectures. arXiv preprint arXiv:1611.06172 (2016)
22. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: Fasttext. zip: Compressing text classification models. arXiv:1612.03651 (2016)
23. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning. pp. 1188–1196 (2014)
24. Li, Y., Pan, Q., Yang, T., Wang, S., Tang, J., Cambria, E.: Learning word representations for sentiment analysis. Cogn. Computation **9**(6), 843–851 (2017)
25. Li, Y., Pan, Q., Yang, T., Wang, S., Tang, J., Cambria, E.: Learning word representations for sentiment analysis. Cognitive Computation **9**(6), 843–851 (2017)

26. Lin, C.C., Ammar, W., Dyer, C., Levin, L.: Unsupervised pos induction with word embeddings. arXiv preprint arXiv:1503.06760 (2015)
27. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proc. of the Annual Meeting of the Assoc. for Computational Linguistics: Human Language Technologies - Vol. 1. pp. 142–150 (2011)
28. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
30. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. CoRR **abs/cs/0506075** (2005), http://arxiv.org/abs/cs/0506075
31. 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). pp. 1532–1543 (2014)
32. Poria, S., Cambria, E., Gelbukh, A.: Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In: Proceedings of the 2015 conference on empirical methods in natural language processing. pp. 2539–2544 (2015)
33. Reforgiato Recupero, D., Cambria, E.: Eswc'14 challenge on concept-level sentiment analysis. In: Presutti, V., Stankovic, M., Cambria, E., Cantador, I., Di Iorio, A., Di Noia, T., Lange, C., Reforgiato Recupero, D., Tordai, A. (eds.) Semantic Web Evaluation Challenge. pp. 3–20. Springer International Publishing, Cham (2014)
34. Reforgiato Recupero, D., Cambria, E., Di Rosa, E.: Semantic sentiment analysis challenge eswc2017. In: Semantic Web Challenges. pp. 109–123. Springer (2017)
35. Reforgiato Recupero, D., Dragoni, M., Presutti, V.: Eswc 15 challenge on concept-level sentiment analysis. In: Gandon, F., Cabrio, E., Stankovic, M., Zimmermann, A. (eds.) Semantic Web Evaluation Challenges. pp. 211–222. Springer International Publishing, Cham (2015)
36. Rodrigues, M.W., Zárate, L.E., Isotani, S.: Educational data mining: a review of evaluation process in the e-learning. Telematics and Informatics (2018)
37. Rodriguez, P., Ortigosa, A., Carro, R.M.: Extracting emotions from texts in e-learning environments. In: 2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems. pp. 887–892. IEEE (2012)
38. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)
39. Rudkowsky, E., Haselmayer, M., Wastian, M., Jenny, M., Emrich, S., Sedlmair, M.: More than bags of words: Sentiment analysis with word embeddings. Communication Methods and Measures **12**(2-3), 140–157 (2018)
40. Saif, H., He, Y., Fernandez, M., Alani, H.: Semantic patterns for sentiment analysis of twitter. In: International Semantic Web Conference. pp. 324–340. Springer (2014)
41. Socher, R., Pennington, J., Huang, E.H., Ng, A.Y., Manning, C.D.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: Proceedings of the conference on empirical methods in natural language processing. pp. 151–161. Association for Computational Linguistics (2011)
42. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing & Management **45**(4), 427–437 (2009)

43. Tang, D., Wei, F., Qin, B., Yang, N., Liu, T., Zhou, M.: Sentiment embeddings with applications to sentiment analysis. IEEE Transactions on Knowledge and Data Engineering **28**(2), 496–509 (2016)
44. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proc. of the Annual Meeting of the Association for Computational Linguistics. pp. 1555–1565 (2014)
45. Tang, D., Qin, B., Liu, T.: Document modeling with gated recurrent neural network for sentiment classification. In: Proceedings of the 2015 conference on empirical methods in natural language processing. pp. 1422–1432 (2015)
46. Tripathy, A., Agrawal, A., Rath, S.K.: Classification of sentiment reviews using n-gram machine learning approach. Expert Systems with Applications **57**, 117–126 (2016)
47. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research **37**, 141–188 (2010)
48. Vu, T., Nguyen, D.Q., Vu, X., Nguyen, D.Q., Catt, M., Trenell, M.: NIHRIO at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. CoRR **abs/1804.00520** (2018), http://arxiv.org/abs/1804.00520
49. Zhang, L., Wang, S., Liu, B.: Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **8**(4), e1253 (2018)
50. Zhang, Z., Lan, M.: Learning sentiment-inherent word embedding for word-level and sentence-level sentiment analysis. In: 2015 International Conference on Asian Language Processing (IALP). pp. 94–97 (Oct 2015)