Received: 11 February 2016 | Revised: 28 April 2016 | Accepted: 23 September 2016

DOI 10.1111/lnc3.12228

WILEY

ARTICLE

Deep learning for sentiment analysis

Lina Maria Rojas-Barahona

Department of Engineering, University of Cambridge, Cambridge, UK

Correspondence
Lina Maria Rojas-Barahona, University of Cambridge, Department of Engineering, Trumpinton Street, Cambridge CB2 1PZ, UK.
Email: lina.rojas@eng.cam.ac.uk

Funding Information
French Services, Industry, and Competitivity General Direction

Abstract
Research and industry are becoming more and more interested in finding automatically the *polarised opinion* of the general public regarding a specific subject. The advent of *social networks* has opened the possibility of having access to *massive blogs*, *recommendations*, and *reviews*. The challenge is to extract the *polarity* from these data, which is a task of *opinion mining* or *sentiment analysis*. The specific difficulties inherent in this task include issues related to *subjective* interpretation and *linguistic phenomena* that affect the polarity of words. Recently, *deep learning* has become a popular method of addressing this task. However, different approaches have been proposed in the literature. This article provides an overview of deep learning for sentiment analysis in order to place these approaches in context.

1 | INTRODUCTION

Society has long been interested in public opinion. Surveys are organized frequently in order to detect the acceptance or rejection of a situation, a measure, or a product. Democracy itself depends on the analysis of mass opinion. *Sentiment analysis* (SA), also called *opinion mining*, is a task of natural language processing (NLP) that finds *automatically* subjective information conveyed by a text such as opinions, sentiments, evaluations, appraisals, and attitudes, among others (Liu, 2012b). In this paper, we focus on sentiment polarity detection.

With the revolution of social media, opinions are constantly generated from all over the world every day and are easily accessible via the web. People follow opinions in social networks, blogs, and tweets, and more and more companies evaluate client satisfaction, preferences, and explore product reviews. As a consequence, there is a need to analyze these data in order to detect the polarized opinion with respect to *something* (e.g., a particular product, a movie, a public person, or a political situation).

Recently, deep learning (DL) has been proposed to solve different NLP tasks (Collobert et al., 2011). Its main advantage is that, contrary to supervised learning, it does not require manually tuned features based on expert knowledge and available linguistic resources. We aim to explain in this paper the basis for understanding DL architectures. The reader is referred to the references for details about more sophisticated architectures (also called variants throughout this paper). This review is targeted to students and researches in the broad area of linguistics, who want to understand deep neural networks and to devise novel trends of research in sentiment analysis.

This paper introduces SA and surveys the use of DL for opinion detection. We start by defining SA (Section 2); afterwards, we introduce the field of DL, the conventional architectures, and their initialization (Section 3). In Section 4, we discuss models for SA, where we identify three main architectures: recursive neural networks (RsNN), non-recursive neural networks, and the combination of both. We present the outcomes and compare the different approaches in Section 5. Finally, we provide conclusions and ideas for future research.

2 | SENTIMENT ANALYSIS

Sentiment analysis is the task of extracting automatically subjective information conveyed by a text. It covers extracting the *polarity (*e.g., *positive* or *negative)* (Turney 2002), the stance (e.g., pros and cons) (Somasundaran & Wiebe 2010), or identifying the target (e.g., a product and a measure) (Hu & Liu 2004) or the holder of an opinion (Kim & Hovy 2006). It also involves analyzing implicatures or effects (Deng & Wiebe 2014) and determining aspects of the target that people like or dislike (Jo & Oh 2011). This paper focuses mainly in polarity detection. Depending on the application, there can be a coarse grained (i.e., positive and negative) or a fine-grained set of polarity categories (i.e., very negative, negative, neutral, positive, and very positive), which reflects the intensity of the polarity.

Three levels of SA have been presented in Liu (2012): (i) document level, (ii) sentence level, and (iii) entity and aspect level. Document level studies the polarity of a document with respect to a single entity (e.g., a product). Sentence level studies the polarity of sentences, analyzing clauses and phrases. The third one, entity and aspect level, analyzes what people like or dislike. An entity-aspect might be a phrase or a single token and its polarity might be different from the overall polarity of the whole text. We cover in this paper the first and the second level of SA.

2.1 | Difficulties

One major difficulty is that the judgment of the polarity of an opinion can be subjective. Humans, for instance, usually disagree when judging the sentiment of a sentence, typically reaching only modest inter-annotator agreement (Bermingham & Smeaton, 2009).

Moreover, polarity has to be interpreted within a context (*contextual polarity*). Thus, the polarity of a word can be *modified* in a sentence due to linguistic phenomena such as *negation*, *modality*, *intensifiers*, *diminishers*, and *word sense* (Wilson et al., 2005).

The following are examples of three linguistic phenomena: negation, valence-shifting (Polanyi & Zaenen, 2006), and irrealis.

A. *no one thinks it is good*
B. *this is a missed opportunity*
C. *it would be good if...*

(A) *Negation*: the word “good” has a positive polarity; however, the sentence is clearly expressing negativity. (B) *Valence-shifting*: the shifter verb “missed” changes the polarity of opportunity, which is a positive word. For instance in the sentence: “The medicine *kills* cancer cells.”, while the phrase cancer cells has negative polarity, the word *kills* reverses the polarity, and the whole sentence has positive connotation (Nakagawa et al. 2010). (C) *Irrealis*: the sentence is expressing an unknown/unreal situation.

2.2 | Application domains

Sentiment analysis has been applied to a variety of domains, from commercial to political and social applications. Commercial applications involve mining reviews (Miller et al., 2011; Pang et al., 2002; Socher et al., 2011; Socher et al., 2013; Tai et al., 2015; Turney, 2002); predicting movie success (Mishne and Glance, 2006; Sadikov et al., 2009); analyzing social influence in online book reviews (Sakunkoo & Sakunkoo, 2009); extracting product opinions (Hu & Liu, 2004); and predicting sales (Liu et al., 2007). There are also political/social applications such as assessing citizens satisfaction of public policies (Kim & Hovy, 2006), predicting election results from twitter (Tumasjan et al., 2010), and mining opinions in social media (Martínez-Cámara et al., 2014; Nakov et al., 2013; Ortigosa et al., 2014; Rosenthal et al., 2014). The reader is referred to Liu (2012) for an extensive survey.

2.3 | Early methods

The first proposed algorithms for SA explored purely linguistic analysis and resources such as syntactic rules, sentiment lexica, and polarity-shifting rules (Polanyi & Zaenen, 2006). They carefully studied the complex linguistic phenomena involved in sentiment detection. A lexicon-based approach to SA, in which dictionaries containing words annotated with their polarities are utilized, is presented in Taboada et al. (2011).

With the advent of *machine learning*,[1] supervised techniques were widely adopted (Nakagawa et al., 2010; Pang et al., 2002, Liu, 2012, Rosenthal et al., 2014). As a consequence, features were carefully designed by experts, and opinion-rich resources (e.g., lexicons and annotated corpora) were made available to the research community.

3 | DEEP LEARNING

Deep learning is an emergent area of machine learning that offers methods for *learning feature representation* in a *supervised* or *unsupervised* fashion within a *hierarchy*. Therefore, *high layers* in the hierarchy will have a more *abstract representation* (i.e., distributed representation) than *lower layers*. High layers evolve during training to exploit complex compositional nonlinear functions of the lower layers (Deng & Yu, 2014).

Artificial neural networks (ANNs) are learning models inspired by biological neural networks that approximate functions that depend *on a large number of inputs* (features or data representation). *Deep neural networks* (DNNs) are ANNs with *multiple hidden layers* between the input and output layers. Thus, from a given input, they are able to learn features (hidden layers) and to give a classification result (output layer). They have been very successful in NLP for part-of-speech tagging, chunking, named entity recognition, and semantic role labeling (Collobert et al., 2011). DNNs are good examples of DL and are the focus of this paper. In the remainder of this section, we introduce the topology of conventional neural networks and their initialization.

3.1 | Feed-forward neural network

Artificial neural networks are composed of small processing units, namely, neurons, which are connected to each other by weighted connections. Thus, a neuron is activated when it receives a signal, then it spreads out the activation to all the neurons connected to it. The feed-forward NN shown in Figure 1 shows a simple type of network with one input layer $\boldsymbol{i}$, one hidden layer $\boldsymbol{h}$, and one output layer $\boldsymbol{o}$.

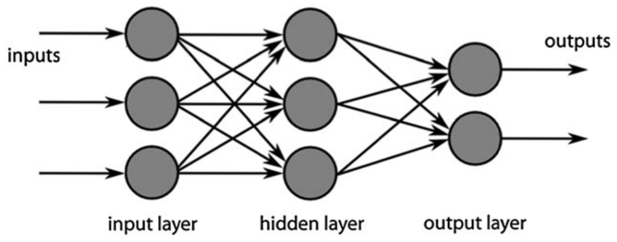

FIGURE 1 Feed-forward neural network. A simple architecture with one hidden layer. All the layers are vectors (i.e., *i*, *h*, and *o*) composed by neurons (units)

The hidden layer $\boldsymbol{h}$ is equal to

$$\boldsymbol{h} = f(\boldsymbol{W}_{ih}\boldsymbol{i}), \tag{1}$$

where $\boldsymbol{i}$ is the vector of the input units, $\boldsymbol{W}_{ih}$ is a weight matrix, and *f(.)* is the hidden layer activation function. Likewise, the output layer vector $\boldsymbol{o}$ is equals to

$$\boldsymbol{o} = g(\boldsymbol{W}_{ho}\boldsymbol{h}), \tag{2}$$

where $\boldsymbol{W}_{ho}$ is a weight matrix and *g(.)* is the output layer activation function. Typically, *f(.)* and *g(.)* are nonlinear compositional functions, such as the sigmoid function [Equation 3)] or the hyperbolic tangent function [Equation 4)].

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{4}$$

The type of output activation depends on the task. For classification tasks such as SA, the output activation function is the softmax function (Bishop 1995):

$$y_k = \frac{e^{h_k}}{\sum_{k'} e^{(h_{k'})}} \tag{5}$$

where k is the index of the output neuron representing one target label or category. The softmax function effectively transforms the network output into a probability distribution over the target categories. For sentiment polarity detection, categories might be the set of binary polarities T = {positive, negative} or a finer grained set of polarities, such as T = {positive, negative, neutral}. Therefore, the output of the neural network can return the probability for each input item to belong to a given polarity. The target with highest probability will then be the result of the prediction.

The process of computing the output layer given the input layer is known as the *forward propagation*. For training neural networks, the weights $\boldsymbol{W}_{ih}$ and $\boldsymbol{W}_{ho}$ need to be adjusted in order to minimize a loss function. The most used loss function for classification tasks is the cross-entropy error (Bishop 1995) between the true target[2] and the predicted target y_k [Equation 6)], where k is the index of a polarity category and $\lambda\|\theta\|^2$ is the L2 regularization term.

$$E(\theta) = \sum_k t_k \log y_k + \lambda \|\theta\|^2 \tag{6}$$

Neural networks are then optimized using gradient descend with error *backpropagation*. Thus, the derivative of the loss function with respect to each of the weights $\boldsymbol{W}$ is computed to adjust the weights in the direction of the negative slope. The reader is referred to Bishop (1995) for detailed information about backpropagation.

3.2 | Conventional neural networks

In this section, we present sequence NNs. These networks can be seen as single blocks that can be used to build deeper architectures (or variants) by further connecting the output layer of one network with the input layer of another network, namely, multi-layer NN. We present the basis for understanding conventional network topologies without detailing their more sophisticated variants.

3.3 | Recurrent neural network

An NN that contains direct cycles in their hidden connections is a recurrent neural network (RNN). The hidden layer of an RNN is equal to

$$\boldsymbol{h}_t = \sigma(\boldsymbol{W}_{ih}\boldsymbol{i}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1}) \tag{7}$$

where t denotes the time step and $\boldsymbol{W}_{hh}$ is the weight matrix representing the recurrent connection. Figure 2 illustrates the architecture of an Elman-type RNN[3] showing the connection between its hidden units from time t-1 to t.

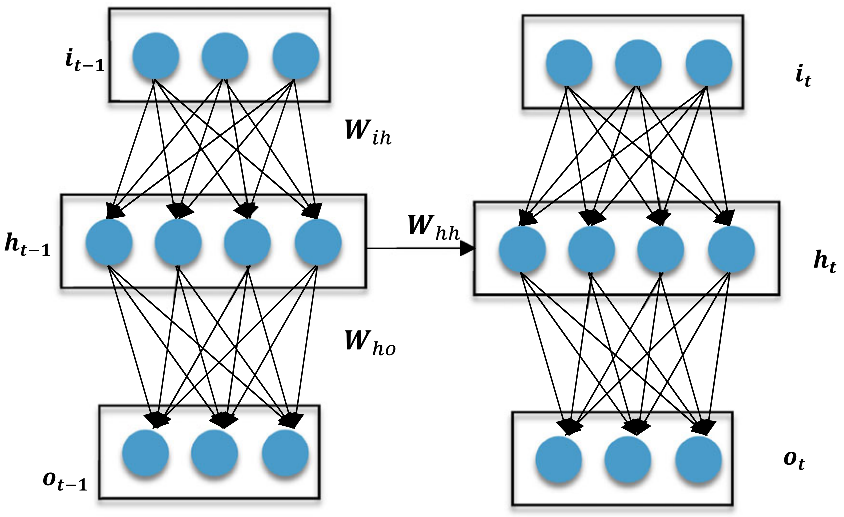

FIGURE 2 A simple recurrent neural network (RNN). Note that the recurrent connection connects two hidden layers. This is an Elman-type RNN

3.3.1 | Long short-term memory

The length of the sequences an RNN (Section 3.2) can process is limited, because of *the exploding* or *vanishing gradient problem* (Bengio et al., 1994).[4] Long short-term memory (LSTM) networks[5] introduce a memory cell that is able to preserve states over long periods of time, overcoming the long-distance dependencies problem of RNNs (Hochreiter & Schmidhuber, 1997). The core of the LSTM is a memory cell, c_t, which is recurrently connected to itself. It has three multiplication units: *an input gate* i_t, *a forget gate* f_t, and *an output gate* o_t (Figure 3). These gating vectors are in [0,1]. The cell makes selective decisions about what information is preserved, and when to allow access to units, via gates that open and close.

The LSTM transition equations are the following:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}\right),\\
\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}\right),\\
\boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}\right),\\
\boldsymbol{u}_t &= \tanh\left(\boldsymbol{W}^{(u)}\boldsymbol{x}_t + \boldsymbol{U}^{(u)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(u)}\right),\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \boldsymbol{u}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1},\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{8}$$

where $\boldsymbol{h}_t$ is the hidden unit at time step t, $\boldsymbol{x}_t$ is the input at the current time step, $\boldsymbol{b}$ is a bias, σ is the logistic sigmoid function, and $\odot$ denotes elementwise multiplication. LSTM can be seen as an NN that contains smart memory units that can remember for an arbitrary length of time. They contain gates that determine when the input is relevant to remember, whether or not it should continue to remember it, and whether or not it should output the value. The value of the gating variables varies for each hidden vector $\boldsymbol{h}_t$; thus the model can learn to represent information over multiple time scales t.

3.3.2 | Convolutional neural network

Initially proposed in Computer Vision, convolutional neural networks (CNNs) are variants of feed-forward NNs with the following properties: (i) convolutional layers: A CNN usually has one or more convolutional layers[6] that produce spatially contiguous features (hidden units); (ii) *sparse connectivity*: Instead of having fully connected neurons, inputs of hidden units in the layer l are from a subset of units in layer l–1 that have spatially contiguous features; (iii) *shared weights* (shown in Figure 4): Units belonging to the same local features share the same weights (weight vector and bias); and (iv)

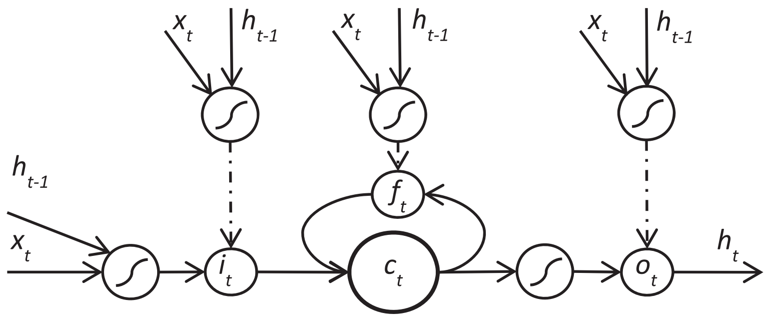

FIGURE 3 A long short-term memory cell c_t with the sigmoid gates: input i_t, forget f_t, and output o_t

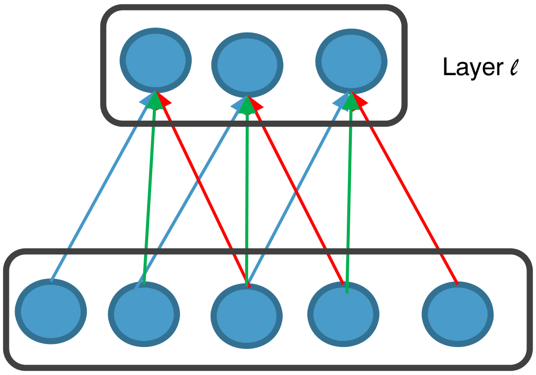

FIGURE 4 A convolutional neural network with three hidden units on Layer l that belong to the same feature map. Arrows with the same color represent shared weights in the activation

pooling: Instead of using all the local features at the next level, they have a pooling layer that computes either the average or the minimum or the maximum of these features. For NLP tasks, convolutional layers extract local features around a window of a given sequence of words. They are often stacked to extract higher level features (Collobert et al., 2011).

3.4 | Initialization of input vectors (word embeddings)

The hidden layers in a DNN encode a *distributed representation* of the input layers. Distributed representations of words are often called *word embeddings* and are *d*-dimensional space representation of words. They are usually the top hidden units of a DL architecture *trained on a very large amount of data*. These *pre-trained* embeddings can be used as input to another task-specific deep neural architecture.

A single word is then represented by a vector that encodes its *distributed properties*. Different word embeddings have been made public. They usually encode syntactic/semantic similarities[7] (Collobert & Weston, 2008 or Turian et al., 2010); contextual information, *Word2Vec* (Mikolov et al., 2013); or correlation between words, *Glove* (Pennington et al., 2014). Meaningful relations between word vectors were introduced by Mikolov et al., 2013); for example, the analogy "King is to Queen as man is to woman" can be encoded in the vector space as **king–queen ≈ man–woman**

Embeddings representing words that co-occur in similar contexts have been proven useful for Named Entity Recognition (Turian et al., 2010). Nevertheless, they might not be as informative for SA, because words with opposite polarities often appear in the same contexts (e.g., it was still a *remarkable performance* vs it was still a *mediocre performance*). Indeed, learning polarity rich word embeddings is an interesting area of research (see Sections 4.3 and 5.3).

4 | DEEP LEARNING MODELS FOR SENTIMENT ANALYSIS

We distinguish in this section three main architectures for SA: *non-recursive neural networks*, *RsNN*, and the *combination of both* as depicted in Table 1. Non-recursive neural networks, which correspond to the conventional networks introduced in the previous section, process words sequentially (e.g., RNN, LSTM, and CNNs). The structure of an RsNN on the contrary follows a *tree structure*. Thus, words are combined based on a *syntactic structure*, starting from the leaves up to the root in a

TABLE 1 Summary of the Deep Learning Models for Sentiment Analysis discussed in this paper

Architectures	Conventional (Sequence or Non-Recursive)	Recursive	Combination of Conventional & Recursive
Basic Types	RNN, LSTM and CNN	RAE, RsNN	Tree-LSTM:
Variants	Bi-RNN, Bi-LSTM, CNN-multichannel, CNN-non-static, DCNN	MV-RsNN, RsNTN	DTree-LSTM CTree-LSTM Deep RsNN

bottom-up fashion. The main motivation for recursive methods is to exploit syntax in order to learn long-distance dependencies between words in a sentence.

4.1 | Non-recursive neural networks

In sequence neural networks, a word will affect the hidden representation of all the subsequent words. Consider the sentence displayed in Figure 5: "I love that movie". In an RNN, words are processed sequentially as follows: first, at ($t = t_0$), it processes the first word "I" and produces a hidden unit h_1. At ($t = t_1$), it processes the second word "love," and it feeds the hidden layer h_1 into h_2, and so on until the end of the sentence is reached. At each step t, the hidden layer h_{t-1} encodes the *full history* of the word sequence from time t_0 to time t–1.

Because sequence models can encode long-range dependencies in the word sequence, any time a negation word or a shifted verb appears in the sentence, the hidden unit of this word will affect the polarity of the subsequent words, and as consequence, the polarity of the sentence. For this reason, sequence models and their variants have been used for SA (Li et al., 2015, Tai et al., 2015). CNNs, which are able to learn invariant features or recurrent patterns, have also been used for SA in Kim (2014) and Kalchbrenner et al. (2014). As we see in Section 5, these learned features seem to work well for SA.

4.2 | Recursive neural network

Recursive deep models for semantic compositionality are presented in this section. These approaches are linguistically motivated in that they explore tree structures (e.g., syntactic structures) and aim to learn elegantly compositional semantics (Table 1).

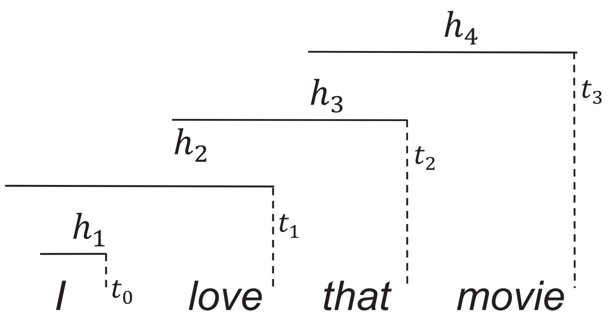

FIGURE 5 A sentence is modeled as tokens processed sequentially by combining the current token with the previous hidden unit at each time step

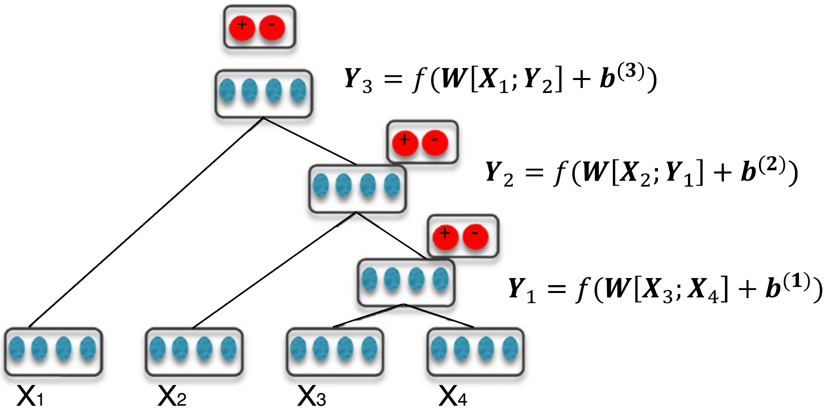
+ -
$Y_3 = f(W[X_1; Y_2] + b^{(3)})$
$Y_2 = f(W[X_2; Y_1] + b^{(2)})$
$Y_1 = f(W[X_3; X_4] + b^{(1)})$
X1
X2
X3
X4

FIGURE 6 Instance of a recursive autoencoder applied to a binary tree. Taken from Socher et al. (2011). Each node of the tree is a vector of units. On top of each parent node, there is a softmax layer that predicts its polarity. The polarity of the sentence is the polarity of the root node

4.2.1 | Recursive autoencoders

The seminal work of Socher et al. (2011) introduced semi-supervised recursive autoencoders (RAE). They learn a tree structure in an unsupervised fashion. The RAE builds a recursive data structure by minimizing the reconstruction error of all vector pairs of words in a sentence. It first composes pairs of word vectors into a parent node, then from the parent reconstructs the children. Thus, the reconstruction error is the Euclidean distance between the original children and the reconstructed ones.

Let us suppose that in Figure 6, the vectors **X**1, **X**2, **X**3, and **X**4 are the (*d*-dimensional) embeddings of the words {“I”, “love”, “that”, “movie”}, respectively. The RAE combines input vectors. It searches all input pairings and finds that by combining vectors **X**3 and **X**4, the parent **Y**1 has the minimal reconstruction error, so it chose **Y**1. Likewise, it computes recursively the parent **Y**2 by combining **Y**1 and **X**2 until it computes the root node **Y**3. Note that the learned binary tree is not necessarily a syntactic tree.

The RAE proposes a parent node *p* as described in Equation 9.[8]

$$p = f\left(\boldsymbol{W}_{(1)}[c_1; c_2] + b_{(1)}\right), \tag{9}$$

where $\boldsymbol{W}_{(1)}$ are the weights and $b_{(1)}$ is the bias applied from the children to the parent and c_1 and c_2 are adjacent child nodes. From this parent, it tries to reconstruct its children.

$$c_1'; c_2' = f\left(\boldsymbol{W}_{(2)} p + b_{(2)}\right), \tag{10}$$

where $\boldsymbol{W}_{(2)}$ and $b_{(2)}$ are the weights and bias. Thus, the reconstruction error for a single triplet of the tree (p, c_1, c_2) is defined as

$$E_{rec}([c_1; c_2]) = \frac{1}{2}[c_1; c_2] - [c'_1; c'_2]^2. \tag{11}$$

The goal of RAE is then to search over all possible binary trees and minimize the reconstruction error of the parent nodes:

$$RAE_{\theta}(x) = argmin_{y \in A(x)} \sum_{s \in T(y)} E_{rec}\left([c_1; c_2]_s\right) \tag{12}$$

where $A(x)$ is the set of all possible trees and $T(y)$ returns all triplets of a given tree (p, c_1, c_2) indexed by s.

Once the optimal binary structure has been learned, a softmax layer is used to predict polarity distributions at every node of the tree [see Equation 5)]. The RAE solves the joint optimization [Equation 13] of the reconstruction [Equation 12] and prediction error [Equation 6].

$$\alpha E_{rec}\left([c_1; c_2]_s; \theta\right) = (1-\alpha)E_{cE}(p_s, t; \theta) \tag{13}$$

Although autoencoders do not follow a syntactic structure, in practice, they produce a competitive baseline when predicting the polarity of a sentence (as shown in Section 5). It learns in an unsupervised fashion (i.e., without manual annotations) phrase representation and phrase structure while using supervision to learn the sentiment distribution of the sentence.

4.2.2 | Constituency tree recursive neural networks

Recursive neural networks in which the recursive structure is not learned but is given by a syntactic parser were introduced in Socher et al. (2013). Constituency trees were used to define the recursive structure, and every phrase of the tree (every parent node) was manually annotated with a polarity and released in the Sentiment Treebank.[9]

Once a constituency tree with polarity annotations is given, a softmax [see Equations 5 and 6] layer can be used to predict polarity distributions at every node. Therefore, this model is able to capture polarity shifting, scope of negation, and contrastive conjunctions such as "but."

Consider in Figure 7 the excerpt of a sentence "… not very good … ." The corresponding embeddings for these words are **a**, **b,** and **c,** respectively. Hence, the parent node resulting of the composition of **b** and **c** is given by

$$\boldsymbol{p}_1 = \boldsymbol{f}\left(\boldsymbol{W}\,[\boldsymbol{b}; \boldsymbol{c}]\right), \tag{14}$$

where $\in \mathbb{R}^{\boldsymbol{d} \times 2\boldsymbol{d}}$.[10]

It is worth noting that in RAE, the polarity of the whole sentence has been replicated at every node of the recursive structure during the learning process, because of the absence of

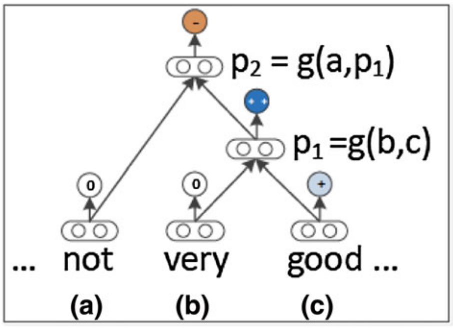

FIGURE 7 Recursive neural networks for sentiment analysis. Taken from Socher et al. (2013).

polarity annotations in the learned tree. Instead, in RsNN, human annotations are provided at each node of the tree.

Other sophisticated RsNN variants were also proposed by these authors, such as matrix-vector RsNN (MV-RsNN) and recursive neural tensor network (RsNTN). For more details, the reader is referred to Socher et al. (2013).

4.3 | Combination of recursive and non-recursive methods

Lately, some authors have proposed the combination of tree-structured neural networks and conventional network architectures, such as LSTM or stacked neural networks. In this section, we present tree-LSTM and deep recursive neural networks (DRsNN).

4.3.1 | Tree-long short-term memory

The Tree-LSTM (Figure 8) aims to exploit syntax to combine words as in natural language while keeping the benefits of LSTM models. In a Tree-LSTM, gating vectors and memory cell updates are dependent on the states of possibly many child units (Tai et al., 2015). Moreover, it contains one forget gate f_k for each child k, selecting in this way which information from each child is worth to be incorporated. Therefore, it can learn to preserve the representation of sentiment-rich children. Two Tree-LSTM are presented in Tai et al. (2015): the child-sum Tree-LSTM and the N-ary Tree-LSTM. The child-sum Tree LSTMs are suitable for dependency tree structures in which a governor (or head) might have many children, namely, dependency Tree-LSTM. Binary Tree-LSTMs are suitable to use constituency trees, namely, constituency-Tree-LSTM.

In a constituency tree, the left child will have different parameters than the right child. Thus, the left hidden state might have either an excitatory or inhibitory effect on the forget gate of the right child.

For SA, at each node j, a softmax layer is used to predict the category $\widehat{y}_j$ given the inputs $\{x\}_j$ observed at nodes in the sub-tree rooted at j. The cost function is the negative log-likehood as defined in Equation 15.

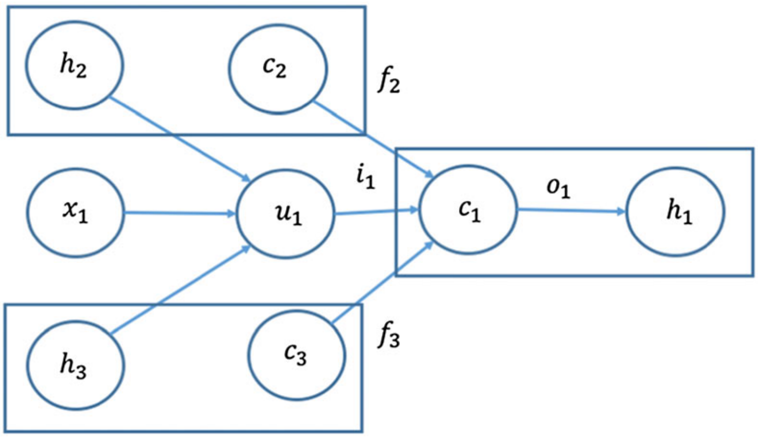

FIGURE 8 Taken from Tai et al. (2015). The memory cell c_1 and hidden layer h_1 in a Tree-LSTM incorporate information from two children: c_2,h_2 and c_3,h_3. Labeled edges represent the gates: input i_1, output o_1, and forget f_2, f_3

$$J(\boldsymbol{\theta}) = -\frac{1}{\boldsymbol{m}}\sum_{k=1}^{m} \log \hat{\boldsymbol{p}}_{\boldsymbol{\theta}}\left(\boldsymbol{y}^{(k)} | \{\boldsymbol{x}\}^{(k)}\right) + \frac{\boldsymbol{\lambda}}{2}\boldsymbol{\theta}_2^2 \tag{15}$$

4.3.2 | Deep recursive neural networks

Deep recursive neural networks are presented in Irsoy and Cardie (2014) for SA and is constructed by stacking multiple layers of recursive neural networks:

$$\boldsymbol{h}_{\eta^{(i)}} = \boldsymbol{f}\left(\boldsymbol{W}^{(i)}[\boldsymbol{b};\boldsymbol{c}]^{(i)} + \boldsymbol{V}^{(i)}\boldsymbol{h}_{\eta^{(i-1)}}\right) \tag{16}$$

where i indexes the multiple stacked layers, the first term has been defined in (14) within each layer, and $V^{(i)}$ is the weight matrix that connects the $(i-1)$th hidden layer to the ith hidden layer. An important benefit is the hierarchy among hidden representations.

4.4 | Initialization, word embeddings

As mentioned in Section 3.3 (see also Figure 9), one can use word embeddings to initialize the input layer. However, considering semantic co-occurrence is not enough to deal with the linguistic phenomena presented in Section 2.2. There are words that *affect* the polarity of other words. Negation can be seen as an operator that rotates the polarity vector of a word, inverting positive polarity into negative and vice versa. One of the first works that deal with this problem is presented in Maas et al. (2011). They learned vectors that capture syntactic and semantic similarities by training jointly probabilistic topic models such as latent dirichlet allocation (LDA) as well as a supervised model that constrains words expressing similar sentiment to have similar representations.

More recently, Tang et al. (2014) presented three distinct DL models to learn sentiment-specific word embeddings (SSWE). SSWEs were trained on tweets that were annotated automatically with a polarity based on the presence or absence of a set of emoticons (e.g., :(, :), etc). Their work is an extension of Collobert & Weston (2008; C&W), which learns syntactic contexts of words by having an objective function that gives a higher score to correct ngrams and lower score to corrupted ngrams.

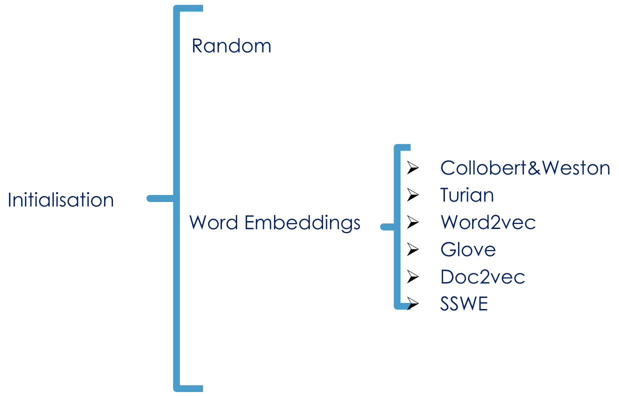

FIGURE 9 Summary of the embeddings discussed in this paper for initialising DL models

The first model $SSWE_h$ adds a softmax layer to C&W model to predict the polarity (i.e., positive and negative). The second model $SSWE_r$ uses a ranking objective function in order to relax the constraint to be positive [1,0] or negative [0,1], allowing distributions such as [0.7,0.3]. The last one $SSWE_u$ is a unified model that combines C&W ngram model and a polarity-sensitive model (i.e., $SSWE_h$ and $SSWE_r$) to predict a two-dimensional vector with language model score and sentiment score of the input ngram.

5 | COMPARISON OF DEEP LEARNING MODELS

We present in this section a collation of the evaluation results of the foregoing DL models on the movie review, the Sentiment Treebank, and the twitter sentiment datasets.

5.1 | Movie reviews dataset

The movies reviews corpus (Pang & Lee, 2005) contains 10,662 reviews, annotated with the categories *positive* and *negative*, fairly distributed (50% of the reviews are negative and 50% are positive).

Table 2 shows the performance of the various models on this corpus in terms of accuracy.[11] Interestingly, RNN and RsNN have similar performance; likewise, their variants Bi-LSTM (i.e., a bidirectional LSTM[12]) and MV-RsNN (Socher et al. 2013). LSTM models learn long-distance dependencies in long reviews (Li et al., 2015) and outperforms RNN and RAE showing that gated sequential models are able to capture relevant contextual polarity.

Recursive models are able to learn the polarity of every tree node in a sentence. However, these results suggest that on this dataset, having syntactic trees is not as informative as processing words sequentially. This can be explained by the fact that this corpus does not provide tree-based annotations. Therefore, the polarity of the sentence is replicated at each node of the tree, and the property of learning the polarity at every tree node remains untapped.

The best models on this dataset are CNNs. The best one is the dynamic convolutional neural networks (DCNN), which combine time delay NN, dynamic k-max pooling, and wide convolutional layers (Kalchbrenner et al., 2014). Other CNNs are ranked second and third. CNN-non-static is a model that tunes pre-trained embeddings during training. CNN-multichannel is a model with two sets of embeddings. It changes the embeddings of one channel while keeping static the embeddings of the other channel (Kim, 2014). In general, CNNs are able to learn contextual semantic features that impact the prediction of the polarity.

5.2 | Sentiment Treebank

A *Sentiment Treebank* was released by Socher et al. (2013) with annotations of five categories: *negative*, *somewhat negative*, *neutral*, *positive*, *somewhat positive*, and *positive*. The treebank contains constituency trees of single sentences extracted from the movies reviews dataset (Section 5.1), with annotations of the polarity at each parent node (i.e., at the phrase level). In total, there are 11,855 sentences and 215.154 phrases. The corpus was split into train, dev, and test datasets with 8544, 1101, and 2210 reviews per dataset, respectively.

Table 3 shows the performance of the models on the semantic treebank. We expect recursive methods to perform better on this corpus, because they were explicitly designed for tree structures. Accordingly, RsNNs outperform RNNs; however, for fine-grained classification, their performance is equivalent to Bi-RNN (i.e., bidirectional-RNN[12]). The accuracy improves when using RsNN variants: MV-RsNN and RsNTN (Socher et al., 2013). However, LSTM models

TABLE 2 Comparison of the accuracy of the models on the movie reviews dataset for the binary classification (i.e., positive/negative)

Model	Accuracy
RNN(Li et al., 2015)	74.0
RsNN(Li et al., 2015)	74.2
RsNTN(Socher et al., 2013)(*)	75.0
Bi-RNN(Li et al., 2015) (+)	76.2
RAE(Socher et al., 2011)	77.7
LSTM(Li et al., 2015)	78.6
Bi-LSTM(Li et al., 2015)(+)	79.0
MV-RsNN (Socher et al., 2013)(*)	79.0
CNN-multichannel (Kim, 2014)	81.1
CNN-non-static (Kim, 2014)	81.5
DCNN (Kalchbrenner et al., 2014)	86.8

Note. Asterisks (*) and plus (+) denote recursive (Socher et al., 2013) and sequence (Li et al., 2015) variants, respectively.

Bi-RNN, bidirectional-recurrent neural network; CNN, convolutional neural network; DCNN, dynamic convolutional neural networks; LSTM, long short-term memory; MV-RsNN, matrix-vector recursive neural networks; RNN, recurrent neural network; RsNN, recursive neural network; RsNTN, recursive neural tensor network.

are competitive enough to overcome RsNN and its variants, proving that LSTM can indeed learn to memorize selectively polarity contributions of words in a sentence. CNN models perform better than LSTM, showing that convolutional layers extract relevant features (invariant patterns in subsequent words).

TABLE 3 Results on the Sentiment Treebank for binary and fine-grained classification

	Accuracy	
Model	Fine-grained	Binary
RNN (Li et al., 2015)	42.0	80.7
RsNN (Socher et al., 2013)	43.2	82.4
Bi-RNN (Li et al., 2015) (+)	43.5	81.6
MV-RsNN (Socher et al., 2013)(*)	44.4	82.9
RsNTN(Socher et al., 2013)(*)	45.7	85.4
LSTM (Tai et al., 2015)	46.4	84.9
CNN-multichannel (Kim, 2014)	47.4	88.1
CNN-non-static (Kim, 2014)	48	87.2
DTree-LSTM (Tai et al., 2015)	48.4	85.7
DCNN(Kalchbrenner et al., 2014)	48.5	86.8
2-layer Bi-LSTM (Tai et al., 2015) (+)	48.5	87.5
Paragraph-Vec (Le & Mikolov, 2014)	48.7	87.8
DRsNN (Irsoy & Cardie, 2014)	49.8	86.6
CTree-LSTM(Tai et al., 2015)	**51**.0	88.0

Note. Asterisks (*) and plus (+) denote recursive (Socher et al., 2013) and sequence (Tai et al., 2015) variants respectively.

Bi-RNN, bidirectional-recurrent neural network; CNN, convolutional neural network; CTree-LSTM, constituency Tree-long short-term memory; DCNN, dynamic convolutional neural networks; DRsNN; deep recursive neural networks; DTree-LSTM, dependency Tree-long short-term memory; LSTM, long short-term memory; MV-RsNN, matrix-vector recursive neural networks; RNN, recurrent neural network; RsNN, recursive neural network; RsNTN, recursive neural tensor network.

Interesting, Paragraph-Vec or Doc2Vec (Le and Mikolov, 2014) achieves a competitive performance. This model is an extension of Word2vec, in which embeddings for entire documents are involved in the prediction of contexts of words.

Accuracy increases even more when using models that combine recursive and conventional networks. The constituency Tree-LSTM (CTree-LSTM) outperforms all the other models. This suggests that the combination of syntactic trees with gated neural networks allows a selective composition of hidden units at the parent nodes. This selective process can learn, for instance, that most of the words remain neutral, and only a few words in a sentence affect the polarity of the others. Interestingly, dependency Tree-LSTM (DTree-LSTM) does not reach the same results because they were trained with less data (150 k labeled nodes vs 319 k for constituency trees), as the authors explained in Tai et al. (2015). Similarly, DRsNN, which has stacked recursive neural networks, performs well for fine-grained classification.

Unsurprisingly, when predicting fine-grained categories, the accuracy of the models is lower than for binary classification. In fact, the achieved accuracy for fine-grained categories is still below 60%, suggesting that new strategies are still needed to improve DL models on this task.

5.3 | Sentiment analysis in twitter

Tweets annotated with the polarities *positive*, *negative*, and *neutral* were made available for the evaluation: *SemEval-2013* (Nakov et al., 2013), *2014* (Rosenthal et al., 2014), and *2015* (Rosenthal et al., 2015). The dataset contains reference to tweet ids that need to be downloaded from twitter. Table 4 shows the SemEval-2013 corpus as downloaded by Tang et al. (2014).

Although the corpus contains three polarity categories, the authors in Tang et al. (2014) reported results on binary classification. They evaluated the impact of using sentiment word embeddings (SSWE) in a CNN by concatenating vectors derived from min, max, and average convolutional layers ($SSWE_u$). Moreover, they compared against the top performing system classifier in SemEval- 2013 (namely, NRC, a supervised model that incorporates polarity lexicons and manually tuned features such as emoticons, hashtags, part of speech, punctuation, etc.).

The results shown in Table 5 suggest that sentiment word embeddings can learn discriminative features for SA *from a large amount of tweets*, comparable with manually designed features. The

TABLE 4 SemEval-2013 sentiment analysis in twitter as download by Tang et al. (2014)

Dataset	Positive	Negative	Neutral	Total
Train	2642	994	3436	7072
Dev	408	219	493	1120
Test	1570	601	1639	3810

TABLE 5 Macro-F1 on binary classification of tweets (as reported in Tang et al., 2014)

Model	Macro-F1
RAE	75.12
NRC	84.73
SSWE	84.98
SSWE + NRC	86.58
SSWE + NRC-ngram[13]	86.48

Note: RAE, recursive autoencoders; SSWE, sentiment-specific word embeddings.

performance is further improved after concatenating SSWE embeddings and the feature vector used in NRC. Moreover, they showed that SSWEs outperform other semantic embeddings (SSWE 83.37 vs C&W 75.89, Word2vec 76.32). Therefore, sentiment word vectors capture better the polarity relation between words. RAE performance is lower. However, it was trained strictly on the SemEval-2013 dataset. The performance would have been much better if it had been trained on a very large twitter dataset as was the case for SSWE.

Finally, in SemEval-2015, the top system for task A, phrase-level polarity, is a deep convolutional network (Rosenthal et al., 2015), showing again that CNNs are able to learn sentiment-relevant features. The interested reader is referred to (Giachanou & Crestani 2016) for a recent survey on sentiment analysis in twitter.

5.4 | Experimental settings

In most of these experiments, the hyper-parameters (learning rate, batch size, regularization term) were learned by using the development set when given in the corpus or cross-validation otherwise and by using AdaGrad for parameter updates (Li et al., 2015, Socher et al., 2013, Kim, 2014, Irsoy & Cardie, 2014 , Tang et al., 2014). The dimension of the word embeddings varies from 25 to 30 in (Socher et al., 2013), 50 in (Tang et al., 2014), 100 in (Socher et al., 2011), 50 and 250 in (Li et al., 2015), 300 in (Kim, 2014), and 400 in (Le and Mikolov, 2014), and these embeddings were pre-trained on a large amount of data.

6 | CONCLUSION

We have reviewed recent work on DL for SA. We presented the distinct proposed architectures and the results achieved so far. We grouped the models into three groups: (i) recursive, (ii) not recursive, and (iii) the combination of both. First, we compared the models for document-level and sentence-level SA, on two datasets, the movies reviews and the Sentiment Treebank, respectively. The best models on the first dataset were CNNs, showing that relevant features can be learned after applying convolutional filters. In addition, variants of LSTM are statistically indistinguishable from variants of recursive models. Models in the third group perform significantly better on the Sentiment Treebank. Unfortunately, these models were not evaluated on the first dataset. Moreover, we showed the importance of initializing the input layers with word embeddings trained on large amount of data. Indeed, CNNs were improved after initializing them with sentiment word embeddings on the SemEval-2013 dataset. Deep learning models have been shown to perform well for polarity detection, being comparable to supervised models with features carefully designed by experts. It is clear that these are partial results of recent ongoing research, and these models will certainly be improved even more in the near future.

7 | FUTURE LINES OF RESEARCH

Most of the work in RsNN use constituency trees. It would be interesting then to explore dependency trees in RsNN, in which case a dependency Sentiment Treebank would be required.

One major drawback of RsNNs is that they depend on a given parse tree. It would be worth jointly optimizing parsing and sentiment classification to form a sentiment parser. In this case, the tree structure would not be simply given but learned during training, following the idea of RAE but with supervision.

Sentiment word embeddings can be used as input to the best performing models (e.g., LSTM-tree and CNNs). Indeed, LSTM-Tree or CNNs can be used to train sentiment word embeddings.

The accuracy was the metric chosen by authors to compare their models; other metrics such as precision, recall, and f-measure will provide better insights about the quality of the prediction, particularly when evaluating fine-grained polarity detection.

A strong trend of research in sentiment analysis regards domain adaptation in deep learning models. Thus, models trained on reviews belonging to one domain can be tested on reviews of another domain and vice versa. Last but not least, richer linguistic-subjective phenomena still remain unexplored such as irony and sarcasm.

ACKNOWLEDGMENTS

The author would like to thank the Synalp team at LORIA, Nancy France, and the dialogue system group at the University of Cambridge. This work is the result of the research conducted in the context of the ITEA2 1105 Empathic Products project, supported by funding from the French Services, Industry, and Competitivity General Direction.

ENDNOTES

[1] Machine learning is an area of artificial intelligence that explores the study and construction of algorithms that can learn from and make predictions on data (Kohavi & Provost, 1998).

[2] A classification task is a supervised learning task; therefore, the data used for training need to be annotated with the true target label.

[3] Elman-type neural network has recurrent connections between hidden layers as in Figure 2. Jordan-type neural network feeds the output of the previous time step into the hidden layer of the current time step.

[4] Basically, when training, an RNN components of the gradient vector can grow or decay exponentially over long sequences, and the network trains very slowly.

[5] The reader can find an implementation of LSTM for sentiment analysis in http://deeplearning.net/tutorial/lstm.html

[6] Convolution is a mathematical operation on two functions. One-dimensional convolution is an operation, such as dot product, between a weight vector and a vector of inputs.

[7] Syntax is encoded in the objective function, which penalizes corrupted ngrams (Collobert & Weston, 2008). Collobert & Weston embeddings differ from those in Turian as reported in: http://ronan.collobert.com/senna/

[8] In Figure 6, to compute the node **Y**1, (c1,c2) = (**X**3,**X**4).

[9] The Stanford Sentiment Treebank is available in http://nlp.stanford.edu/sentiment/

[10] Note that Equation (14) is equivalent to Equation (9) with bias $b = 0$.

[11] The proportion of correct answers (true positive and true negative) over the total of answers considered in the evaluation.

[12] Two sequential networks running in parallel, one on the input sentence and the other on the reversed input, are called bidirectional.

[13] NRC-ngram refers to the feature set of NRC leaving out ngram features.

REFERENCES

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, *5*, 157–166.

Bermingham, A., & Smeaton, A. F. (2009). *A study of inter-annotator agreement for opinion retrieval*. Paper presented at the SIGIR.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford university press: New York, NY, USA.

Collobert, R., & Weston, J. (2008). *A unified architecture for natural language processing: Deep neural networks with multitask learning*. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, *12*, 2493–2537.

Deng, L., & Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, *7*, 197–387.

Deng, L., & Wiebe, J. (2014). Sentiment propagation via implicature constraints. *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 377–385). EACL.

Giachanou, A., & Crestani, F. (2016). Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys*, *49*(2), 1–41.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*, 1735–1780.

Hu, M., & Liu, B. (2004). *Mining and summarizing customer reviews*. Paper presented at the Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.

Irsoy, O., & Cardie, C. (2014). Opinion mining with deep recurrent neural networks: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

Jo, Y., & Oh, A. H. (2011). Aspect and sentiment unification model for online review analysis. *Proceedings of the fourth ACM international conference on Web search and data mining*, (pp. 815–824). ACM.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). *A convolutional neural network for modelling sentences*. Paper presented to the Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014.

Kim, Y. (2014). *Convolutional neural networks for sentence classification*. Paper presented at the Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar.

Kim, S.-M., & Hovy, E. (2006). *Extracting opinions, opinion holders, and topics expressed in online news media text*. Paper presented at the Proceedings of the Workshop on Sentiment and Subjectivity in Text.

Kohavi, R., & Provost, F. (1998). Glossary of terms. *Machine Learning*, *30*, 271–274.

Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053.

Li, J., Jurafsky, D., & Hovy, E. (2015). *When are tree structures necessary for deep learning of representations?* Paper presented at the roceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, *5*, 1–167.

Liu, J., Cao, Y., Lin, C.-Y., Huang, Y., & Zhou, M. (2007). *Low-quality product review detection in opinion summarization*. Paper presented at the EMNLP-CoNLL.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning word vectors for sentiment analysis*. Association for Computational Linguistics Stroudsburg, PA, USA.

Martínez-Cámara, E., Teresa Martín-Valdivia, M., Alfonso Urena-López, L., & Rturo Montejo-Ráez, A. (2014). Sentiment analysis in twitter. *Natural Language Engineering*, *20*, 1–28.

Mikolov, T., Chen, K., Corrado, G. & Dean J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Miller, M., Sathi, C., Wiesenthal, D., Leskovec, J., & Potts, C. (2011). *Sentiment flow through hyperlink networks*. Paper presented at the ICWSM.

Mishne, G., & Glance N. (2006). *Predicting movie sales from blogger sentiment*. Paper presented at the Proceedings of the Spring Symposia on Computational Approaches to Analyzing Weblogs, Stanford, US.

Nakagawa, T., Inui, K., & Kurohashi, S. (2010). Dependency tree-based sentiment classification using CRFs with hidden variables. Paper presented at the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Stroudsburg, PA, USA.

Nakov, P., Rosenthal, S., Kozareva, Z., Stoyanov, V., Ritter, A., & Wilson, T. (2013). *SemEval-2013 task 2: Sentiment analysis in twitter*. Paper presented at the Proceedings of the 7th International Workshop on Semantic Evaluation.

Ortigosa, A., Martín, J. M., & Carro, R. M. (2014). Sentiment analysis in Facebook and its application to e-learning. *Computers in Human Behavior*, *31*, 527–541.

Pang, Bo., & Lee, L. (2005). *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. Paper presented at the Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Stroudsburg, PA, USA.

Pang, Bo, Lee, L., & Vaithyanathan, S. (2002). *Thumbs up?: Sentiment classification using machine learning techniques*. Paper presented at the Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, Stroudsburg, PA, USA.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12.1532-43.

Polanyi, L., & Zaenen, A. (2006). Contextual Valence Shifters. In J. Shanahan, Y. Qu & J. Wiebe (Eds.), *Computing attitude and affect in text: Theory and applications*, (pp.1–10). Springer: Netherlands.

Rosenthal, S., Nakov, P., Ritter, A., & Stoyanov, V. (2014). *SemEval-2014 task 9: Sentiment analysis in twitter*. Paper presented at the Proceedings of the 8th International Workshop on Semantic Evaluation.

Rosenthal, S., Nakov, P., Kiritchenko, S., Mohammad, S. M., Ritter, A., & Stoyanov, V. (2015). Semeval-2015 task 10: Sentiment analysis in twitter.

Sadikov, E., Parameswaran, A., & Venetis, P. (2009). *Blogs as predictors of movie success*. Paper presented at the 3rd International AAAI Conference on Weblogs and Social Media, San Jose, California.

Sakunkoo, P. & Sakunkoo, N. (2009). *Analysis of social influence in online book reviews*. Paper presented at the 3rd International AAAI Conference on Weblogs and Social Media, San Jose, California.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). *Semi-supervised recursive autoencoders for predicting sentiment Distributions*. Paper presented at the Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank*. Paper presented at the Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Stroudsburg, PA.

Somasundaran, S., & Wiebe, J. (2010). Recognizing stances in ideological on-line debates. *Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, *37*, 267–307.

Tai, K. S., Socher, R., & Manning, C. D. (2015). *Improved semantic representations from tree-structured long short-term memory networks*. Paper presented at the Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, Beijing, China.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). *Learning sentiment-specific word embedding for twitter sentiment Classification*. Paper presented at the Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland.

Tumasjan, A., Sprenger, T. O., Sandner, P. G., & Welpe, I. M. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, *10*, 178–185.

Turian, J., Ratinov, L., & Bengio, Y. (2010). *Word representations: A simple and general method for semi-supervised learning*. Paper presented at the Proceedings of the 48th annual meeting of the association for computational linguistics.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. Paper presented at the Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). *Recognizing contextual polarity in phrase-level sentiment analysis*. Paper presented at the Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, CA.

How to cite this article: Rojas-Barahona LM. Deep learning for sentiment analysis, *Lang Linguist Compass* 2016;*10*:701–719. doi: 10.1111/lnc3.12228