A Neural Word Embeddings Approach for Multi-Domain Sentiment Analysis

Mauro Dragoni and Giulio Petrucci

Abstract—Multi-domain sentiment analysis consists in estimating the polarity of a given text by exploiting domain-specific information. One of the main issues common to the approaches discussed in the literature is their poor capabilities of being applied on domains which are different from those used for building the opinion model. In this paper, we will present an approach exploiting the linguistic overlap between domains to build sentiment models supporting polarity inference for documents belonging to every domain. Word embeddings together with a deep learning architecture have been implemented into the NeuroSent tool for enabling the building of multi-domain sentiment model. The proposed technique is validated by following the Dranziera protocol in order to ease the repeatability of the experiments and the comparison of the results. The outcomes demonstrate the effectiveness of the proposed approach and also set a plausible starting point for future work.

Index Terms—Sentiment analysis, natural language processing, neural networks, multi-domain sentiment analysis, deep learning

1 Introduction

Sentiment Analysis is a natural language processing (NLP) task [1] which aims at classifying documents according to the opinion expressed about a given subject [2]. Generally speaking, sentiment analysis aims at determining the attitude of a speaker or a writer with respect to a topic or the overall tonality of a document. In the recent years, the exponential increase in the use of the Web for exchanging public opinions about events, facts, products, etc. led to an extensive usage of sentiment analysis approaches, especially for marketing purposes.

The paper [3] formalizes the sentiment analysis problem by representing an "opinion" as a quintuple

$$\langle o_j, f_{jk}, so_{ijkl}, h_i, t_l \rangle, \tag{1}$$

where o_j is a target object, f_{jk} is a feature of the object o_j, so_{ijkl} is the opinion polarity value given by the opinion holder h_i about the feature f_{jk}, and t_l is the timestamp when the opinion is expressed. The value of so_{ijkl} can be classified as positive, negative, or neutral. In general, different and more fine-grained rating schemes can be used as well, depending on the level of satisfaction the opinion holder has with respect to the specific object feature.

Many works available in the literature address the sentiment analysis problem without distinguishing domain specific information of documents when sentiment models are built. The necessity of investigating this problem from a multi-domain perspective is led by the different influence that a term might have in different contexts. Let us consider the following examples. In the first one, we have an *emotion-based* context where the adjective *"cold"* is used differently based on the feeling, or mood, of the opinion holder:

1) Our new colleague behaves in a very *cold* way with us.
2) A *cold* drink is the best thing we can drink when the temperature is very hot.

In the second example instead we have a *subjective-based* context where the adjective *"small"* is used differently based on the product category reviewed by a user:

1) The sideboard is *small* and it is not able to contain a lot of stuff.
2) The *small* dimensions of this decoder allow to move it easily.

In the first context, we considered two different *emotional* domains: in the first one, a person is commenting about the behavior of his colleague by using the adjective *"cold"* with a *negative* polarity. Instead, in the second one, a person is referring to the adjective *cold* in a *positive* way as a good solution for the described situation.

Instead, in the second context, we considered the interpretation of texts referring to two different domains: "Furnishings" and "Electronics". In the first one, the polarity of the adjective *"small"* is *negative* because it highlights an issue of the described item. On the other hand, in the second domain, the polarity of same adjective may be considered *positive*.

The multiple facets with which textual information can be analyzed requires the design of approaches able to address different domains. The idea of adapting terms polarity to different domains emerged only in the last decade [4]. Multi-domain sentiment analysis approaches discussed in the literature (surveyed in Section 2) focus on building models for transferring information between pairs of domains [5]. While on the one hand such approaches

• *The authors are with Fondazione Bruno Kessler, University of Trento, Trento, TN 38123, Italy. E-mail: {dragoni, petrucci}@fbk.eu.*

Manuscript received 21 Jan. 2017; revised 2 June 2017; accepted 12 June 2017. Date of publication 20 June 2017; date of current version 5 Dec. 2017. (Corrresponding author: Mauro Dragoni.)
Recommended for acceptance by E. Cambria, A. Hussain, and A. Vinciarelli.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TAFFC.2017.2717879

allow to propagate specific domain information to others, their drawback is the necessity of building new transfer models every time a new domain has to be analyzed. Thus, such approaches do not have a great generalization capability of analyzing texts, because transfer models are limited to the N domains used for building the models.

The contribution presented in this paper aims at addressing the challenge of working in a multi-domain environment. The described approach, implemented into the NeuroSent tool, is based on the following pillars:

- the use of word embeddings for representing each word contained in raw sentences;
- the word embeddings are generated from an opinion-based corpus instead of a general purpose one (like news or Wikipedia);
- the design of a deep learning technique exploiting the generated word embeddings for training the sentiment model;
- the use of multiple output layers for combining domain overlap scores with domain-specific polarity predictions.

The last point enables the exploitation of linguistic overlaps between domains, which can be considered one of the pivotal assets of our approach. This way, the overall polarity of a document is computed by aggregating, for each domain, the domain-specific polarity value multiplied by a belonging degree representing the overlap between the embedded representation of the whole document and the domain itself. The use of this strategy ease the validation of our approach from two perspectives: (i) to measure the effectiveness of our model on the domains used for creating the model itself, and (ii) to observe how our model behaves in classifying document coming from domains different from the ones adopted for building the model (i.e., generalization of our approach). This point represents an innovative aspect with respect to the state of the art of multi-domain sentiment analysis.

The paper is structured as follows. Section 2 presents a survey on works about sentiment analysis in both the single and the multi-domain environment and provides a brief review of some advances in the field of NLP that have successfully exploited deep learning approaches. In Section 3, we included information about the material used for developing our approach. Then, Section 4 provides the elements for understanding how our deep network works from a mathematical perspective. In Section 5, we present the overall architecture of NeuroSent that is evaluated in Section 6. Finally, Section 7 concludes the article.

2 Related Work

In this Section, we briefly review the main contributions in the field of sentiment analysis and opinion mining, first from a general standpoint and then with a particular attention to the multi-domain scenario. A brief overview of significant recent contributions in the NLP field that based on deep learning approaches follows, pointing out some application to the sentiment analysis task.

2.1 Sentiment Analysis and Opinion Mining

The topic of sentiment analysis has extensively been studied in the literature [3], [6], where several techniques have been proposed and validated.

Machine learning techniques are the most common approaches used for addressing the sentiment analysis problem. For instance, in [2] and [7] the authors compared the performance of Naive-Bayes, Maximum Entropy, and Support Vector Machines classifiers in sentiment analysis, using different features like considering only unigrams, bigrams, combination of both, incorporating parts of speech and position information, or considering only adjectives.

The recent massive growth of online product reviews paved the way for using sentiment analysis techniques in marketing activities. The issue of detecting the different opinions concerning the same product expressed in the same review emerged as a challenging problem. This task has been carried out by identifying the aspect of the product that a sentence in the opinion may refer to. In the literature, many approaches have been proposed: conditional random fields (CRF) [8], hidden Markov models (HMM) [9], sequential rule mining [10], dependency tree kernels [11], and clustering [12].

Recently, the application of sentiment analysis approaches attracted a lot of interest also in the social networks research field [13]. The use of social networks for expressing opinions and comments about products, political or social events, significantly increased in the last years. However, the analysis of the social network environment brought to light new challenges mainly related to (i) the different ways people express their opinions (i.e., *multi-modality*) and to (ii) the management of noisy data contained in social network texts [14].

The social dimension of the Web fostered the development of multi-disciplinary approaches combining computer and social sciences to improve the interpretation, recognition, and processing of opinions and sentiments expressed in social networks. The synergy between these approaches has been called sentic computing [15]. Sentic computing has been employed for addressing several cognitive-inspired problems like the classification of natural language text [16] and the extraction of emotions from images [17].

Real-world solutions have been also developed. For example the authors of Sentilo [18], [19] presented an approach where a neo-Davinsonian theory of meaning is the backbone of a strategy used to process the text. This approach is then hybridized with Semantic Web technologies.

2.2 Multi-Domain Sentiment Analysis

According to the nomenclature widely used in the literature (see [4]), we call *domain* a set of documents about similar topics, e.g., a set of reviews about similar products like mobile phones, books, movies, etc.. The massive availability of multi domain corpora in which similar opinions are expressed about different domains opened the scenario for new challenges. Researchers tried to train models capable to acquire knowledge from a specific domain and then to exploit this knowledge for working on documents belonging to different ones. This strategy was called domain adaptation. The use of domain adaptation techniques demonstrated that opinion classification is highly sensitive to the domain from which the training data is extracted. The reason is that when using the same words, and even the same language constructs, we may obtain different opinions, depending on the domain. The classic scenario occurs when the same word has positive connotations in one domain and negative connotations in another one, as we showed within the examples presented in Section 1.

Several approaches related to multi-domain sentiment analysis have been proposed. Roughly speaking, all of these approaches rely on one of the following ideas: (i) the transfer of learned classifiers across different domains [4], [5], [20], [21], and (ii) the use of propagation of labels through graph structures [22], [23], [24], [25].

While on the one hand such approaches demonstrated their effectiveness in working in a multi-domain environment, on the other hand they suffered by the limitation in adapting to domain that are different from those used for building the model. With respect to this issue, our approach, by starting from a limited number of domain-based labeled data, is able to build models supporting the polarity inference of texts belonging to unseen domains. The model is then capable to deal with documents that are radically different from the ones used in the training process.

2.3 Deep Learning in NLP and Sentiment Analysis

Neural networks are a well known family of statistical learning models which have been largely exploited for problems of classification, regression and clustering. Along the years, they have been employed in different areas: image processing, time series analysis, digital signal processing, and NLP. A well known problem in training multi-layered networks in gradient-based methods and backpropagation is the *vanishing gradient* problem, first demonstrated in [26]: a smaller and smaller fraction of the error is backpropagated back through these layers, significantly slowing down the training process. Recently, the widespread of the Graphical Processing Unit (GPU) computing architectures provided researchers with a significantly increased amount of computational power, allowing to overcome such problem, even if only in a pragmatical way, as remarked in [27]. In this scenario, a new interest flourished in *deep* network architectures, which were already theoretically established and proved to be advantageous (see [28]). As a consequence, different complex network architectures exploiting different neural cell models, have been used for several tasks achieving remarkable results. A sort of new discipline arisen, commonly referred to as *deep learning*, aiming to exploit such systems to reduce the gap between Machine Learning and classical Artificial Intelligence.

The main feature of these systems is that they can easily deal with distributed representations of text, which are extremely flexible and robust. Moreover, the dimensions of distributed vector space models are typically smaller of vocabulary dimensions, helping to avoiding the *curse of dimensionality*. Deep networks have been used to build a distributed representations of single words and to train a language model in the seminal work [29]. In [30] a single network architecture is trained to perform several typical NLP tasks: part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. The main advantage of a unified learning process is that it is less dependent on intermediate representations or task-specific features: this ends up in a significant reduction of feature engineering costs. In such approach, the distributed representation of words are learned from large amount of unlabeled training data.

In Recursive Neural Networks (RecNNs, see [31]) two input vectors from a sequence are combined into a single vector of the same dimension, which is itself recursively combined with another vector from the input space: as long as we go through the recursion, the resulting vector will hold the distributed representation of larger chunks of text. Such architectures have been successfully used to parse Natural Language sentences, as in [32].

Recurrent Neural Networks (RNNs) are fed sequentially: at each step, their state depends on the current input value and on their state at the previous step. This aspect makes them particularly suitable to model natural language sentences as a sequence of words. A pivotal aspect of such architecture is that, as long as we keep feeding it, we obtain a distributed representation that goes beyond the single word. The usage of RNN architectures in the slot filling task has been explored in [33], with a minimum memory effect through windowing the input sequence. Long Short-Term Memory (LSTM) units (see [34]) in order to achieve more powerful memory capabilities, are used in [35] where a RNN is trained to execute short fragment of python code in a sequence-to-sequence translation fashion. In [36], Gated Recursive Units (GRUs) are used for a machine translation task. Such units are conceptually simpler and computationally less expensive than LSTM ones, providing though a shorter term memory. In this approach, two RNNs are combined. The first one encodes the input sequence into a single vector. Such vector is then fed into the second RNN which acts as a decoder, generating the output sequence. Recently, the approach presented in [37] aimed to train a RNN based system to translate natural language definitions into logic formulae in an end to end fashion, instead of the conventional approach based on the usage of statistical NLP toolkits and a catalog of handcrafted rules. Such rules can be learned by the system with a reduced number of annotated examples, relieving the knowledge engineer from the burden of manually encode and maintain such rules set.

In [38], the network memory is built on the neural implementation of traditional data structures as Stacks, Queues and Double-Ended Queues. Such network is trained for a generic language transduction problem.

Convolutional Neural Networks (CNNs) are feed-forward architectures that exploit local connectivity patterns: incoming connections for a hidden unit comes from other units of the previous layer which are *similar* w.r.t. the features they describe. Such networks are trained in [39] to learn distributed representations of sentences in different language without word alignment: the more two sentences from different languages have a similar distributed representation, the more are likely to have the same meaning. Such distributed representation can be seen as a sort of representation in a common, subsymbolic language. A similar approach specifically developed for the sentiment analysis task is presented in [40].

Several deep learning based approaches have been evaluated in Sentiment Analysis tasks. In [41], Recursive Neural Networks are used to handle the syntactic tree structure of a sentence: following the generated parse tree, the different distributed representations of sentence parts are recursively built. The model is trained on the Stanford Sentiment Treebank, which has annotation on the whole parse tree. Authors of [42] learn a distributed representation of reviews through Convolutional Neural Networks which are subsequently feed into Recurrent Neural Networks to learn distributed representation of the viewed products and of the opinion holders. In [43] a 7-layer deep Convolutional Neural Network has been trained to identify the target of an opinion within a text fragment, in conjunction with some linguistic patterns. An Extreme Learning Machine approach implemented over the data analytics framework Apache Spark[1] has been proposed in [44]. The approach deals with large amount of natural language text coming from the Social Web.

3 Material

Here, we briefly describe the preparatory material we used for developing the NeuroSent architecture. In particular, we focus on the dataset used for evaluating our system from the multi-domain perspectives and the basic tools used for building our model.

3.1 The Dranziera Dataset

The evaluation of our approach have been performed by following the Dranziera protocol [45] in order to ease comparisons with future systems. The protocol is composed by a multi-domain dataset of one million reviews crawled from product pages on the Amazon web site and by a precise methodology (explained in Section 6).

Reviews belong to twenty different domains, called in-model domains (IMD): “Amazon Instant Video”, “Automotive”, “Baby”, “Beauty”, “Books”, “Clothing Accessories”, “Electronics”, “Health”, “Home Kitchen”, “Movies TV”, “Music”, “Office Products”, “Patio”, “Pet Supplies”, “Shoes”, “Software”, “Sports Outdoors”, “Tools Home Improvement”, “Toys Games”, and “Video Games”.

For each domain, we extracted twenty-five thousands positive and twenty-five thousands negative reviews that have been split in five folds containing five thousands positive and five thousands negative reviews each. This way the dataset is balanced with respect to both the polarities of the reviews and to the domain which they belong to. The choice between positive and negative documents has been inspired by the strategy used in [4], where reviews with 4 or 5 stars have been marked as positive, while the ones having 1 or 2 stars have been marked as negative. Furthermore, each domain dataset is balanced with respect to positive and negative polarities as well. The choice of having a balanced dataset is based on our idea not to provide a dataset which reflects the same proportion of the reality. Indeed, the proportion measured in the training set reflects only a part of the reality; thus, an unbalanced dataset would be misleading during systems training. Instead, by providing a balanced dataset, systems are able to analyze the same number of positive and negative contexts for each domain. This way, built models are supposed to be fairly used in any kind of opinion inference testing environment.

The split of each domain in five folds allows to easily have a clear distinction between the samples used for training the system and the ones used for testing it. Most of the work in the literature cites the dataset used, but without specifying which part has been used for the training phase and which for the testing phase.

Besides the domains available in the Dranziera dataset, we used other 7 test sets. These test sets were necessary for measuring the general effectiveness of the proposed approach in inferring polarities of documents belonging to domains different from the ones included in the models. These domains have been called out-model domains (OMD): “Cell Phones Accessories”, “Gourmet Foods”, “Industrial Scientific”, “Jewelry”, “Kindle Store”, “Musical Instruments”, and “Watches”.

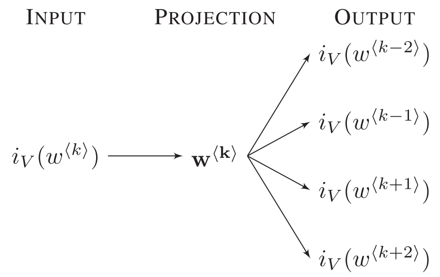

Input
Projection
Output
$i_V(w^{\langle k \rangle})$
$\mathbf{w}^{\langle \mathbf{k} \rangle}$
$i_V(w^{\langle k-2 \rangle})$
$i_V(w^{\langle k-1 \rangle})$
$i_V(w^{\langle k+1 \rangle})$
$i_V(w^{\langle k+2 \rangle})$

Fig. 1. The Skip-gram model.

3.2 Neural Word Embeddings

The training of word embeddings have been performed on a different dataset in order to verify the generalization of our embeddings. We used the Blitzer dataset [4] to pre-train the word embeddings that will be used to represent words feeding our neural network model. We extracted the plain text from the Blitzer dataset and tokenized it *as is* using the Stanford CoreNLP Toolkit [46]. N been performed against the text. We used the skip-gram model to learn our word embeddings. The main intuition behind this model is that, given a word $w^{\langle k \rangle}$ at the kth position within a sentence, it tries to predict the most probable surrounding context, as depicted in Fig. 1.

The word is represented as its index i within the vocabulary V and fed into a projection layer that turns this index into a continuous vector given by the corresponding ith row in the layer weight matrix. Such vector is subsequently used to predict the surrounding words. At the end of the training process, the weights of the layer will be the embedding matrix of our vocabulary. The Skip-gram Model has been presented in detail in [47].

3.3 Deep Learning Library

Deeplearning4j[2] (DL4J) is an open-source, distributed deep-learning library written for Java and Scala. The library integrates Hadoop and Spark technologies and it is designed to be used in business environments on distributed GPUs and CPUs. DL4J aims to be a cutting-edge framework providing fast-prototyping capabilities also to non researchers. It is largely customizable and can import neural net models from most major frameworks via Keras,[3] including TensorFlow, [4] Caffe,[5] Torch,[6] and Theano,[7] bridging the gap between the Python and the Java Virtual Machine ecosystems, with a cross-team toolkit for data scientists, and data engineers.

The DL4J has been used in our work for developing the neural network model described in Section 4. From the

1. https://spark.apache.org/

2. https://deeplearning4j.org
3. https://keras.io/
4. https://www.tensorflow.org/
5. http://caffe.berkeleyvision.org/
6. http://torch.ch/
7. https://github.com/Theano/Theano

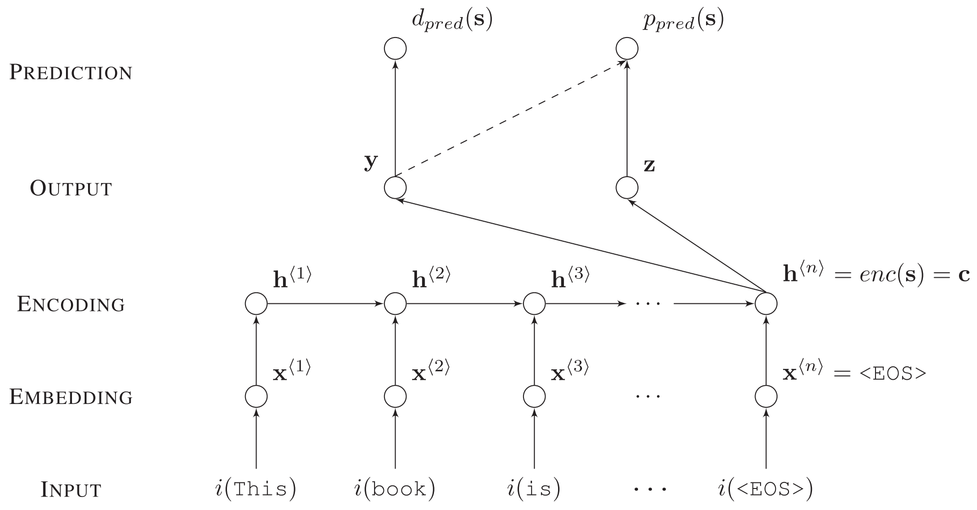

$d_{pred}(\mathbf{s})$
$p_{pred}(\mathbf{s})$
PREDICTION
OUTPUT
$\mathbf{y}$
$\mathbf{z}$
ENCODING
$\mathbf{h}^{\langle 1\rangle}$
$\mathbf{h}^{\langle 2\rangle}$
$\mathbf{h}^{\langle 3\rangle}$
$\mathbf{h}^{\langle n\rangle} = enc(\mathbf{s}) = \mathbf{c}$
EMBEDDING
$\mathbf{x}^{\langle 1\rangle}$
$\mathbf{x}^{\langle 2\rangle}$
$\mathbf{x}^{\langle 3\rangle}$
$\mathbf{x}^{\langle n\rangle}$ = <EOS>
INPUT
i(This)
i(book)
i(is)
i(<EOS>)

Fig. 2. The generic network architecture.

implementation perspective, the DL4J library has been used as back-end layer instantiating the structure of our neural model. In particular, our network has been build by starting from the *ComputationGraphConfiguration* class.[8] This class is a configuration object for neural networks with arbitrary connection structure. It is analogous to the *MultiLayerConfiguration*, but it allows considerably greater flexibility for building network architectures. Specifically, the network architecture is a directed cyclic graph, where each vertex in the graph may for example be a layer or a vertex/object that defines arbitrary forward and backward pass functionality. A *ComputationGraphConfiguration* object may have an arbitrary number of inputs (multiple independent inputs, possibly of different types), and an arbitrary number of output layers. The latter is the reason for which we adopted this class for developing our network. Indeed, our model foresees two separate output layers: a first layer for computing the belonging degree of a document with respect to each domain, and a second layer supporting the computation of document polarity with respect to each domain. Section 5 discusses the actual integration of the network and its usage.

4 Network Description

The main idea underlying our approach is the following: we summarize the whole review into a single distributed vector (the so called *encoding* phase) and use it to predict the binary polarity–positive or negative–of the review. We also exploit domain identification as a parallel task: given the same embedding vector, we want to predict the belonging of the review to one of the 20 domains in our dataset. The network is trained jointly on both task. In this section, after a brief overview about the notation and conventions we used, the different components of the network models we tested will be described in detail, while and overall representation is provided in Fig. 2.

4.1 Notation and Conventions

We used bold uppercase letters for matrices, say $\mathbf{W}$, and bold lowercase for vectors, say $\mathbf{b}$. When representing a vector explicitly in its components, we use square brackets and subscripts to indicate the position of each component within the vector, like in $\mathbf{x} = [x_1, \dots, x_n]$. To represent a time series, we use the superscript between angle brackets for the components. So, if a $l \times h$ matrix is a time series of l vectors of size h, we write $\mathbf{H} = [\mathbf{h}^{\langle 1\rangle}, \dots, \mathbf{h}^{\langle l\rangle}]$. Similarly, if a vector represents a time series of scalar, we will write $\mathbf{x} = [x^{\langle 1\rangle}, \dots, x^{\langle n\rangle}]$. Finally, we will indicate the set of parameters of our model with the capital Greek letter θ.

8. The official DL4J example can be found at https://goo.gl/vYtex3

4.2 Input and Embedding Layer

We represent a review as a sequence of words, followed by a conventional symbol <EOS> that marks the end of the sequence. Each word is represented with its position, or *index*, within the vocabulary V of all the known words in our model. If the word book is the 23rd word in our vocabulary, we will have that $i_V(\texttt{book}) = 23$.

Given a sentence as a temporal sequence of symbols, each of which representing a word, say $\mathbf{s} = [s^{\langle 1\rangle}, \dots, s^{\langle n\rangle} = <\texttt{EOS}>]$, we represent it as the the sequence of the indexes in the vocabulary V of each word, $\mathbf{i}_s = [i^{\langle 1\rangle}, \dots, i^{\langle n\rangle}] = [i_V(s^{\langle 1\rangle}), \dots, i_V(s^{\langle n\rangle}) = i_V(<\texttt{EOS}>)]$. The kth item of such sequence will be the index in the vocabulary V of the kth word in the sentence. From now on, we will omit both the vocabulary subscript and the word argument and will write just $i^{\langle k\rangle}$ to indicate the kth element of the indexes sequence.

Each index in the vocabulary corresponds to a row in the embedding matrix $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, which is the distributed word vector for the given word—where $|V|$ is the dimension of the vocabulary and d is the dimension of the word vector, an hyper-parameter of the model.

Said $\mathbf{x}^{\langle k\rangle}$ the word embedding for the kth word in the sentence and $i^{\langle k\rangle}$ the index of the word, we have that $\mathbf{x}^{\langle k\rangle} = \mathbf{E}[i^{\langle k\rangle}]$. Our sentence can be represented as a sequence of word embedding vectors, $\mathbf{X} = [\mathbf{x}^{\langle 1\rangle}, \dots, \mathbf{x}^{\langle n\rangle}]$. Such embedding matrix is previously learned via the skip-gram model (presented in 3.2) in a separate phase.

4.3 The Encoding Layer

The meaning of the sentence must be captured into a single distributed vector. This operation is called *encoding*. Given a sentence $\mathbf{s}$ of n words, we indicate the result of the encoding phase as $enc(\mathbf{s}) = \mathbf{c}$, where $\mathbf{c} \in \mathbb{R}^h$. The dimension h is an hyper-parameter. We used a recurrent layer to encode our input sentence. In such layers, the activation of the hidden layer at the kth time step, depends on the current input *and* on the activation at the previous step. Generically, we have that

$$\mathbf{h}^{\langle k \rangle} = g(\mathbf{x}^{\langle k \rangle}, \mathbf{h}^{\langle k-1 \rangle}; \theta_{enc}); \tag{2}$$

where g is the so called *cell function* that depends on a set of parameters θ_{enc} that are learned during the training phase. The final vector is the activation of the last time step, namely

$$enc(\mathbf{s}) = \mathbf{c} = \mathbf{h}^{\langle n \rangle} = g(\mathbf{x}^{\langle n \rangle}, \mathbf{h}^{\langle n-1 \rangle}; \theta_{enc}). \tag{3}$$

We used the well known LSTM cell function (see [34]) to implement our encoder. LSTM cells have the capability to maintain a memory of the input they have seen across a large number of time steps, partially overcoming the problem of gradient vanishing. The cell is internally structured with four *gates*, namely the forget gate $\mathbf{f}$, the input gate $\mathbf{i}$, the cell update gate $\mathbf{g}$, and the output gate $\mathbf{o}$. At each time step, we have that

$$\begin{aligned}
\mathbf{f}^{\langle k \rangle} &= \sigma(\mathbf{W}_f \mathbf{x}^{\langle k \rangle} + \mathbf{U}_f \mathbf{h}^{\langle k-1 \rangle} + \mathbf{b}_f);\\
\mathbf{i}^{\langle k \rangle} &= \sigma(\mathbf{W}_i \mathbf{x}^{\langle k \rangle} + \mathbf{U}_i \mathbf{h}^{\langle k-1 \rangle} + \mathbf{b}_i);\\
\mathbf{g}^{\langle k \rangle} &= tanh(\mathbf{W}_g \mathbf{x}^{\langle k \rangle} + \mathbf{U}_g \mathbf{h}^{\langle k-1 \rangle} + \mathbf{b}_g);\\
\mathbf{o}^{\langle k \rangle} &= \sigma(\mathbf{W}_o \mathbf{x}^{\langle k \rangle} + \mathbf{U}_o \mathbf{h}^{\langle k-1 \rangle} + \mathbf{b}_o);
\end{aligned} \tag{4}$$

where $\theta_{enc} = [\mathbf{W}_f, \mathbf{U}_f, \mathbf{b}_f, \mathbf{W}_i, \mathbf{U}_i, \mathbf{b}_i, \mathbf{W}_g, \mathbf{U}_g, \mathbf{b}_g, \mathbf{W}_o, \mathbf{U}_o, \mathbf{b}_o]$ is the set of parameters to be learned. The cell state $\mathbf{q}^{\langle k \rangle}$ is updated according to the following

$$\mathbf{q}^{\langle k \rangle} = \mathbf{f}^{\langle k \rangle} \odot \mathbf{q}^{\langle k-1 \rangle} + \mathbf{i}^{\langle k \rangle} \odot \mathbf{g}^{\langle k \rangle}; \tag{5}$$

where the $\odot$ symbol represents the element-wise product between two vectors, $\sigma(\cdot)$ the sigmoid function and the $\tanh(\cdot)$ function can be replaced by any other nonlinear function. Intuitively we can see how the forget gate $\mathbf{f}^{\langle k \rangle}$ controls how much of the previous cell state gets into the current state, while the input gate controls how much of the cell update signal gets into it. Finally, the cell activation is given by the:

$$\mathbf{h}^{\langle k \rangle} = \mathbf{o}^{\langle k \rangle} \odot tanh(\mathbf{q}^{\langle k \rangle}); \tag{6}$$

where the output gate controls the output signal coming from the cell state.

4.4 Output Layers and Loss Functions

Since we are training the network jointly for the task of domain and polarity identification, we have two output layers, one for each task. Detailed descriptions follow.

4.4.1 Domain Identification

The output layer for the domain identification models the probability of the review–the input sentence–to belong to one of the domains in our dataset. We feed the encoder output, the vector $\mathbf{c}$, into a projection layer ending up into

$$\mathbf{y} = softmax(\mathbf{W}_y \mathbf{c} + \mathbf{b}_y); \tag{7}$$

where $\mathbf{W}_y \in \mathbb{R}^{h \times |D|}$; $\mathbf{b}_y \in \mathbb{R}^{|D|}$, said D the set of all the domains in the dataset. The vector $\mathbf{y}$ is a vector of dimension $|D|$, with one component per domain. The softmax operator ensures all the components of the vector to fall within the interval $[0; 1]$ and that their value sums up to 1: in this way, the value of the jth component of the vector will represent the probability that the input review belongs to the jth domain

$$y_j = p(domain(\mathbf{s}) = j|\theta). \tag{8}$$

The predicted domain is the one whose probability is the maximum, namely

$$d_{pred} = \arg\max_j \, p(domain(\mathbf{s}) = j|\theta). \tag{9}$$

Said $\hat{\mathbf{y}}$ as the gold truth value for a given sentence, we compute the loss function for the given example as the categorical cross entropy between the gold truth and the predicted output vector

$$\mathcal{L}_{\mathbf{s}} = -\sum_{j=1}^{|D|} \hat{y}_j log(y_j). \tag{10}$$

Please note that $\hat{\mathbf{y}}$ is a one-hot vector, with a component of value 1 for the correct domain and components of value 0 for all the other. The parameters of this layer that must be learned during the training phase are $\theta_y = [\mathbf{W}_y, \mathbf{b}_y]$.

4.4.2 Polarity Identification

The output layer for the polarity identification is slightly different from the one used for the domain. The output is still a vector of dimension $|D|$, one for each known domain, filled with values in the interval $[-1; 1]$, obtained via the following

$$\mathbf{z} = tanh(\mathbf{W}_z \mathbf{c} + \mathbf{b}_z); \tag{11}$$

where $\mathbf{W}_z \in \mathbb{R}^{h \times |D|}$; $\mathbf{b}_z \in \mathbb{R}^{|D|}$. Each component of the vector represents a sort of polarity score for that domain, namely

$$z_j = p(polarity(\mathbf{s}) = j|\theta). \tag{12}$$

We compute the loss function for this task from the $\mathbf{z}$ vector. The gold truth value for the polarity is represented with another vector $\hat{\mathbf{z}}$ of $|D|$ dimensions, filled with 0 except for the position corresponding to the proper domain, where the value of the negative or positive label, namely $\{0; 1\}$ is set. We use the categorical cross entropy as the loss function. The contribution to the loss function value from each example will be

$$\mathcal{L}_{\mathbf{s}} = -\sum_{j=1}^{|D|} \hat{p}_j \, log(p_j). \tag{13}$$

The final polarity prediction is performed after a weighted sum with the different domain probabilities. The process is described in detail in Section 5.3. The parameters of this layer that must be learned during the training phase are $\theta_z = [\mathbf{W}_z, \mathbf{b}_z]$.

5 Platform Architecture

Fig. 3 shows the overall architecture of the proposed approach. This architecture has been entirely developed in

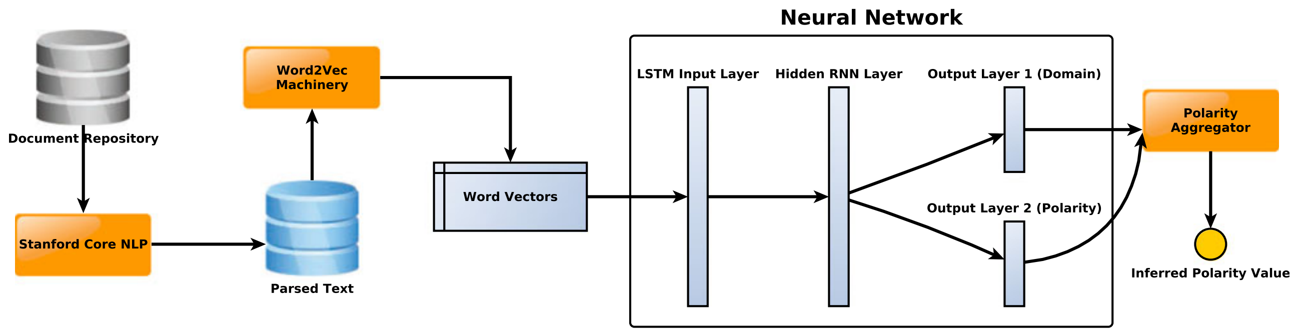


Fig. 3. Architecture of the proposed approach.

Java with the support of the DL4J library (see Section 3.3 for details) and it is composed by three main phases:

- Generation of Word vectors (Section 5.1): raw text, appropriately tokenized using the Stanford CoreNLP Toolkit, is provided as input to a 2-layers neural network implementing the skip-gram approach described in Section 3.2 with the aim of generating word vectors.
- Learning of Sentiment Model (Section 5.2): word vectors are used for training a recurrent neural network implementing two output layers supporting the two classification tasks mentioned in the previous section: "polarity classification", where the word vector sequence is classified as positive or negative, and "domain classification" in which it is provided an overlap degree between the word vector sequence and each domain used for training the model.
- Computation of Document Polarity (Section 5.3): numeric data provided by the output layers are aggregated for inferring the overall polarity value of a text.

In the following subsections, we describe in more detail each phase by providing also the settings used for managing our data.

5.1 Generation of Word Vectors

The generation of the word vectors has been performed by applying the skip-gram algorithm mentioned in Section 3.2 on the raw natural language text extracted from the Blitzer dataset. The rationale behind the choice of this dataset focuses on three reasons:

- the dataset contains only opinion-based documents. This way, we are able to build word embeddings describing only opinion-based contexts.
- the dataset is multi-domain. Information contained into the generated word embeddings comes from specific domains, thus it is possible to evaluate how the proposed approach is general by testing the performance of the created model on test sets containing documents coming from the domains used for building the model or from other domains.
- the dataset is smaller with respect to other corpora used in the literature for building other word embeddings that are currently freely available, like the Google News ones.[9] Indeed, as introduced in Section 1, one of our goal is to demonstrate how we can leverage the use of dedicated resources for generating word embeddings, instead of corpora's size, for improving the effectiveness of classification systems.

These three points represent the main original contributions of this work, in particular the aspect of considering only opinion-based information for generating word embeddings. While embeddings currently available are created from big corpora of general purpose texts (like news archives or Wikipedia pages), ours are generated by using a smaller corpus containing documents strongly related to the problem that the model will be thought for. On the one hand, this aspect may be considered a limitation of the proposed solution due to the requirement of training a new model in case of problem change. However, on the other hand, the usage of dedicated resources would lead to the construction of more effective models.

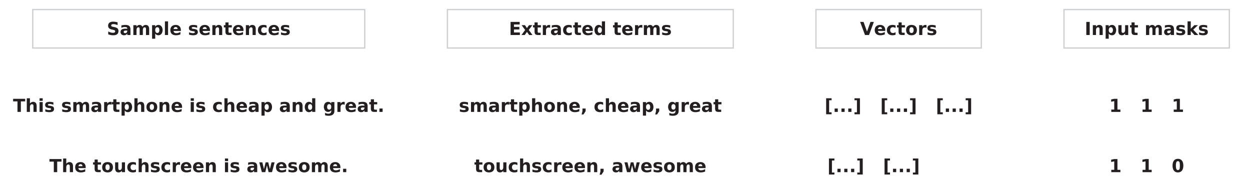
Word embeddings have been generated by the Word2-Vec implementation integrated into the Deeplearning4j library briefly presented in Section 3.3. The algorithm has been set up with the following parameters: the size of the vector to 64, the size of the window used as input of the skip-gram algorithm to 5, and the minimum word frequency was set to 1. The reason for which we kept the minimum word frequency set to 1 is to avoid the loss of rare but important words that can occur in domain specific documents like the ones contained in the Blitzer dataset.

5.2 Learning of the Sentiment Model

The sentiment model is built by starting from the word embeddings generated during the previous phase. In Section 4.4, we already introduced both the tasks performed by our model and we generally refer to Section 4 for the mathematical details of our approach. Here, we summarize the main steps performed for building our model, from the conversion of the input sentences to the strategy used for inferring the overall polarity of a document.

The first step consists in converting each textual sentence contained within the dataset into the corresponding numerical matrix $\mathbf{S}$ where we have in each row the word vector representing a single word of the sentence, and in each column an embedding feature. Given a sentence s, we extract all tokens t_i, with $i \in [0, n]$, and we replace each t_i with the corresponding embedding $\mathbf{w}$. During the conversion of each word in its corresponding embedding, if such embedding is not found, the word is discarded. At the end of this step, each sentence contained in the training set is converted in a matrix $\mathbf{S} = [\mathbf{w}^{\langle 1 \rangle}, \ldots, \mathbf{w}^{\langle n \rangle}]$.

9. https://github.com/mmihaltz/word2vec-GoogleNews-vectors



Sample sentences	Extracted terms	Vectors	Input masks
This smartphone is cheap and great.	smartphone, cheap, great	[...] [...] [...]	1 1 1
The touchscreen is awesome.	touchscreen, awesome	[...] [...]	1 1 0

Fig. 4. Example of sentence transformation and association with the input mask.

Before giving all matrices as input to the neural network, we need to include both padding and masking vectors in order to train our model correctly. Padding and masking allows us to support different training situations depending on the number of the input vectors and on the number of predictions that the network has to provide at each time step. In our scenario, we work in a many-to-one situation where our neural network has to provide one prediction (sentence polarity and domain overlap) as result of the analysis of many input vectors (word embeddings).

Padding vectors are required because we have to deal with the different length of sentences. Indeed, the neural network needs to know the number of time steps that the input layer has to import. This problem is solved by including, if necessary, into each matrix $\mathbf{S}_k$, with $k \in [0, z]$ and z the number of sentences contained in the training set, null word vectors that are used for filling empty word's slots. These null vectors are accompanied by a further vector telling to the neural network if data contained in a specific positions has to be considered as an informative embedding or not.

Fig. 4 shows an example where we have two sentences of different lengths in the training set and how they are represented when they are sent to the input layer of the neural network. From the left, in the first block we have the original sentence; in the second block we extracted the relevant terms that are three for the first sentence, and two for the second one. Then, for each word, we get the corresponding word vector. Finally, we create input masks vectors through which we are able to tell to the neural network that the third element of the second sentence does not exist.

Beside padding, masking is needed for telling to the neural network that the prediction does not have to be provided for each step, but only at the end of each vector sequence, as mentioned above. Fig. 5 shows how we addressed this problem. By supposing to have four word vectors in our sentence, at each step, the network reads one word vector and, after consumed the last one, the prediction is computed and the error is back propagated. As it is possible to observe in Fig. 5: (i) we do not use the mask for the input layer because all vectors have to be considered (exception is done for the vectors that are used as padding); and, (ii) we use a mask associated with each output layer in order to have the prediction only after the elaboration of the last time step.

A final note concerns the back propagation of the error. Training recurrent neural networks can be quite computationally demanding in cases when each training instance is composed by many time steps. A possible optimization is the use of truncated back propagation through time (BPTT) that was developed for reducing the computational complexity of each parameter update in a recurrent neural network. On the one hand, this strategy allows to reduce the time needed for training our model. However, on the other hand, there is the risk of not flowing backward the gradients for the full unrolled network. This prevents the full update of all network parameters. For this reason, even if we work with recurrent neural networks, we decided to do not implement a BPTT approach but to use the default backpropagation implemented into the DL4J library.

Concerning information about network structure, the input layer was composed by 64 neurons (i.e., embedding vector size), the hidden RNN layer was composed by 128 nodes, and the output layers contained 20 nodes each. The network has been trained by using the Stochastic Gradient Descent with 1000 epochs and a learning rate of 0.002.

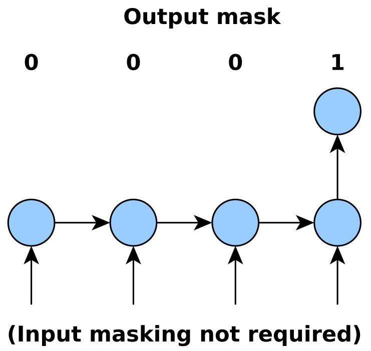


Fig. 5. How input and output mask are set in our scenario.

5.3 Computation of Document Polarity

The operation of computing the entire polarity of a document is performed by combining the numeric data provided by the two output layers of the network. As explained earlier, one of the assumptions of the proposed approach is to exploit possible linguistic overlaps between different domains for compensating missing knowledge from the training set. From here, the intuition of using two output layers: the first one for computing the belonging degree between a sentence and each domain, and the second one for computing the polarity of the document with respect to each domain. Given $\mathbf{y}$ the vector of dimension $|D|$ containing the belonging degree of a sentence to each domain, and $\mathbf{z}$, the vector of dimension $|D|$ containing the polarity score for each domain, the overall predicted polarity of a document T is computed by scaling the dot product between the two vectors by their dimension, as follow

$$p_T = \frac{\mathbf{y} \cdot \mathbf{z}}{|D|}. \tag{14}$$

For completeness, please consider the following numerical example. Let us assume to have three domains A, B, and C and a document j for which hold the following values

$$y_1 = d(A) = 0.81,$$
$$y_2 = d(B) = 0.12,$$
$$y_3 = d(C) = 0.07,$$
$$z_1 = p(A) = 0.72,$$
$$z_2 = p(B) = 0.32,$$
$$z_3 = p(C) = -0.45,$$

where $d(\cdot)$ and $p(\cdot)$ represents the domain belonging degree and the polarity score for a given domain, respectively. The polarity score of the document j will be

$$p = \frac{(0.81 \cdot 0.72) + (0.12 \cdot 0.32) + (0.07 \cdot -0.45)}{3} = 0.1967, \tag{15}$$

where a value greater or equal than zero indicates a positive polarity, while the polarity is considered negative if the score is less than zero.

6 Evaluation

The NeuroSent approach discussed in Section 4 and the platform presented in Section 5 have been evaluated by adopting the Dranziera protocol[10] [45]. The dataset adopted in our evaluation campaign has been introduced in Section 3. Here, we describe the implemented validation procedure and we discuss the obtained results.

6.1 Evaluation Procedure

The validation procedure leverage on a five-fold cross evaluation setting in order to validate the robustness of the proposed solution. The approach has been compared with seven baselines:

- Support Vector Machine (SVM): classification was run with a linear kernel type by using the Libsvm [48]. Libsvm uses a sparse format so that zero values do not need to be captured for training files. This can cause training time to be longer, but keeps Libsvm flexible for sparse cases.
- Naive Bayes (NB) and Maximum Entropy (ME): the MALLET: MAchine Learning for LanguagE Toolkit [49] was used for classification by using both Naive Bayes and Maximum Entropy algorithms. For the experiments conducted in our evaluation, the Maximum Entropy classification has been performed by using a Gaussian prior variance of 1.0.
- Domain Belonging Polarity (DBP): we computed the text polarity by using only information of the domain that a document belongs to. This means that the linguistic overlap between domains has not been considered.
- Domain Detection Polarity (DDP): we computed the text polarity by using only information of the domain guessed as the most appropriate for the document that has to be evaluated. This means that the similarity between text content and domain is preferred with respect to the domain used for tagging the text.
- IRMUDOSA System [50]: the text polarity is computed by aggregating fuzzy polarities associated with each opinion concept. This approach implements a multi-domain strategy where polarities computed over different domains are aggregated by using the Equation (14). In our evaluation, we applied the approach described in [50] to the Dranziera dataset.
- Convolutional Neural Network [40] (CNN): we compared our architecture with a classic CNN. Models have been trained with the embeddings created from the Blitzer dataset.
- Google Word Embeddings (GWE): we trained our models by using the pre-trained Google-News word vectors[11] instead of the ones created from the Blitzer dataset. The goal of this comparison is to show how the embeddings created by using a smaller, but more opinion-oriented, dataset may lead to better results with respect to the use of bigger, but general, embeddings.

For each baseline, we measured the overall accuracy, the precision and recall averaged over the two classes (positive and negative) and the F1-score. In the end, we reported their averages together with the standard deviation measured over the five folds.

The same baselines have been used to evaluate the model against the OMD test sets. In this case, the DBP baseline has not been applied due to the mismatch between the domains used for building the model and the ones contained in the test sets. Each OMD test set has been applied to all five models built, and the scores averaged.

6.2 Results

Here, we show the results of the evaluation campaign conducted to validate the presented approach. Tables 1 and 2 present a summary of the performance obtained by NeuroSent and by the seven baselines on IMD and OMD, respectively. The first column contains the name of the approach, the second, third, and fourth contain the average precision, recall and F1 score computed over all domains, while the fifth contains the average standard deviation computed on the F1 score during the cross-fold validation. Finally, the sixth and seventh columns contain the minimum and the maximum F1 score measured during the evaluation.

Table 1 shows the results obtained on the domains contained in the Dranziera dataset; while in Table 2, we presented the results obtained by testing our approach on documents coming from domains that are not contained in the Dranziera dataset.

For completeness, we reported the comparison between the accuracies obtained by the evaluated system for each domain. Table 3 and 4 presents the results on the domains contained or not within the Dranziera dataset, respectively.

By considering the overall results obtained on the IMD and reported in Table 1, we may observe how NeuroSent outperforms the considered baselines. The measured F1 scores (average, minimum, and maximum) are higher of

10. All the material used for the evaluation and the built models are available at http://www.maurodragoni.com/research/opinionmining/dranziera/protocol.php

11. https://github.com/mmihaltz/word2vec-GoogleNews-vectors

TABLE 1
Comparison Between the Results Obtained by the Baselines and the Ones Obtained by NeuroSent on the Dranziera Dataset

Approach	Avg. Precision	Avg. Recall	Avg. F1	Avg. Deviation	Avg. Min. F1	Avg. Max. F1
Support Vector Machine	0.6890	0.7097	0.6987	0.0119	0.6636	0.7395
Naive-Bayes	0.6956	0.6915	0.6929	0.0062	0.6544	0.7205
Maximum Entropy	0.7073	0.7085	0.7074	0.0098	0.6543	0.7357
Domain Belonging Polarity	0.7108	0.7331	0.7218	0.0228	0.7153	0.7543
Domain Detection Polarity	0.6731	0.7546	0.7115	0.0384	0.7121	0.7456
IRMUDOSA System	0.7410	0.7984	0.7686	0.0154	0.7469	0.7981
CNN Architecture	0.8037	0.7727	0.7879	0.0516	0.7026	0.8647
Google Word Embeddings	0.8008	0.7921	0.7964	**0.0042**	0.7537	0.8413
NeuroSent	**0.8515**	**0.8407**	**0.8460**	0.0102	**0.7966**	**0.8730**

TABLE 2
Comparison Between the Results Obtained by the Baselines and the Ones Obtained by NeuroSent on the Domains Not Contained in the Dranziera Dataset

Approach	Avg. Precision	Avg. Recall	Avg. F1	Avg. Deviation	Avg. Min. F1	Avg. Max. F1
Support Vector Machine	0.6507	0.6437	0.6571	0.0090	0.6004	0.6950
Naive-Bayes	0.6435	0.6459	0.6260	**0.0023**	0.6103	0.6683
Maximum Entropy	0.6524	0.6475	0.6501	0.0038	0.6302	0.6905
Domain Belonging Polarity	-	-	-	-	-	-
Domain Detection Polarity	0.6609	0.6931	0.6766	0.0096	0.6706	0.7198
IRMUDOSA System	0.8115	0.6985	0.7508	0.0160	0.6889	0.7755
CNN Architecture	0.7848	0.8120	0.7982	0.0457	0.6543	0.8675
Google Word Embeddings	0.7930	0.7864	0.7896	0.0078	0.7597	0.8328
NeuroSent	**0.8442**	**0.8343**	**0.8392**	0.0125	**0.8054**	**0.8712**

TABLE 3
Detailed Results Obtained on Domains Contained within the Dranziera Dataset by the Baselines and by NeuroSent

Domain	Tested System								
	SVM	NB	ME	DBP	DDP	IRMUDOSA	CNN	GWE	NeuroSent
Amazon Instant Video	0.7017	0.6544	0.7026	0.7230	0.7147	0.7751	0.8002	**0.8047**	0.8017
Automotive	0.7166	0.7172	0.7172	0.7202	0.6943	0.7412	0.7335	0.7829	**0.8537**
Baby	0.6885	0.6929	0.7155	0.7088	0.6938	0.7652	0.8405	0.7964	**0.8518**
Beauty	0.6982	0.7023	0.7230	0.7481	0.7341	0.7797	0.8279	0.8099	**0.8550**
Books	0.6923	0.6873	0.6887	0.6957	0.6926	0.7315	0.7848	0.7537	**0.7966**
Clothing Accessories	0.6988	0.6904	0.7224	0.8038	0.7856	0.8462	0.7778	0.8115	**0.8696**
Electronics	0.6851	0.6880	0.6988	0.7309	0.7035	0.7492	0.7945	0.7711	**0.8641**
Health	0.6717	0.7205	0.6629	0.6887	0.6867	0.7527	0.7794	0.7865	**0.8611**
Home Kitchen	0.7217	0.7178	0.6900	0.7137	0.6929	0.7683	0.7474	0.8189	**0.8686**
Movies TV	0.7354	0.6915	0.7160	0.7030	0.7122	0.7743	0.7453	0.7913	**0.8090**
Music	0.6936	0.6701	0.6542	0.7171	0.7216	0.7834	0.7635	0.7713	**0.8083**
Office Products	0.7321	0.6910	0.7314	0.7298	0.7017	0.7523	0.7990	0.8028	**0.8730**
Patio	0.6875	0.6923	0.7142	0.7024	0.6926	0.7459	**0.8566**	0.8026	0.8564
Pet Supplies	0.6817	0.7078	0.7302	0.6680	0.6626	0.7195	0.8164	0.7908	**0.8361**
Shoes	0.6705	0.7164	0.7276	0.8324	0.8115	0.8434	0.8143	0.8413	**0.8655**
Software	0.7395	0.6762	0.6872	0.7196	0.7151	0.7462	0.7913	0.7645	**0.8479**
Sports Outdoors	0.6685	0.7050	0.7314	0.7084	0.7129	0.7927	0.7581	0.8151	**0.8669**
Tools Home Improvement	0.7325	0.6896	0.7356	0.6842	0.6887	0.7438	0.7920	0.7905	**0.8518**
Toys Games	0.6636	0.6664	0.6948	0.7383	0.7108	0.8018	0.7673	0.8365	**0.8624**
Video Games	0.6954	0.6808	0.7038	0.6999	0.7012	0.7590	0.7286	0.7853	**0.8206**
Average	0.6987	0.6929	0.7074	0.7218	0.7115	0.7686	0.7878	0.7964	**0.8460**

around 6 percent with respect to the GWE and CNN baselines and of more than 8–10 percent with respect to the others. This first consideration demonstrated the superiority of the use of word embeddings with respect to strategies that do not implement this technique. The poor performance of the classic machine learning approaches: SVM, NB, and ME, are surprising. A more in depth analysis of the results obtained by these three baselines shown how these approaches failed in classifying long reviews containing many opinion-based sentences.

A similar rank can be observed by analyzing the precision and recall values, where the differences between NeuroSent and the baselines remains more or less the same. Concerning the stability of the algorithm, we can notice that the exploitation of the domain information leads to a higher standard deviation. In both IMD and OMD test sets, the

TABLE 4
Detailed Results Obtained on Domains That Are Not Contained within the Dranziera Dataset by the Baselines and by NeuroSent

Domain	Tested System								
	SVM	NB	ME	DBP	DDP	IRMUDOSA	CNN	GWE	NeuroSent
Cell Phones Accessories	0.6671	0.6209	0.6904	-	0.6675	0.7032	0.7385	0.7597	**0.8431**
Gourmet Foods	0.6376	0.6257	0.6384	-	0.6738	0.7638	0.8194	0.7898	**0.8227**
Industrial Scientific	0.6175	0.6234	0.6301	-	0.6392	0.6821	0.7457	0.7717	**0.8155**
Jewelry	0.6003	0.6191	0.6423	-	0.6628	0.7826	0.7578	0.8328	**0.8712**
Kindle Store	0.6877	0.6102	0.6337	-	0.7105	0.7560	0.7774	0.7635	**0.8054**
Musical Instruments	0.6949	0.6140	0.6827	-	0.6938	0.7811	0.8226	0.8018	**0.8597**
Watches	0.6944	0.6682	0.6330	-	0.6889	0.7867	**0.8825**	0.8080	0.8567
Average	0.6571	0.6260	0.6501	-	0.6766	0.7508	0.7982	0.7896	**0.8392**

domain-based approaches registered a higher standard deviation with respect to the SVM, NB, and ME baselines.

A last consideration can be done concerning the detailed results reported in Table 3: in 18 out of 20 domains, the proposed approach outperforms the other baselines. The only exceptions are the "Amazon Instant Video" and "Patio" domains, where, respectively, the GWE and CNN baselines registered a small improvement with respect to F1 score obtained by NeuroSent.

The second overall evaluation concerns the analysis of the results obtained on the set of OMD. Results are shown in Table 2. The first thing that we may observe is how the effectiveness obtained by the proposed system is very close to the one obtained in the IMD evaluation. Indeed, the difference between the two F1 averages is less than 1 percent. This aspect remarked the capability of the proposed approach of working in a cross-domain environment and of exploiting the domain linguistic overlaps for estimating the overall document polarity. Concerning detailed results, here the improvement of NeuroSent is clearer. The only exception is the "Watches" domain, where the CNN baseline obtained an improvement of around 3 percent with

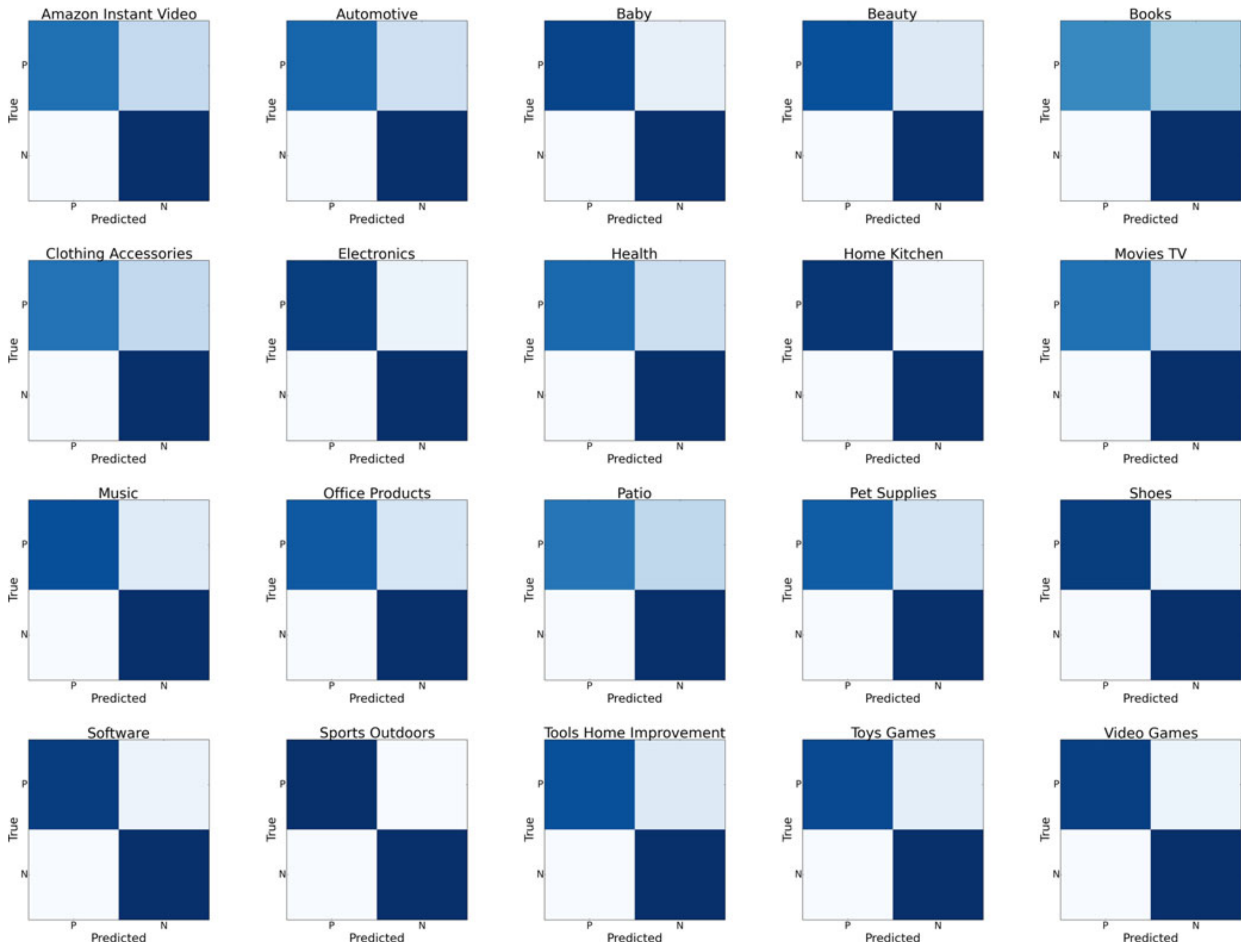


Fig. 6. Confusion matrices for the predictions on the In-Model Domains.

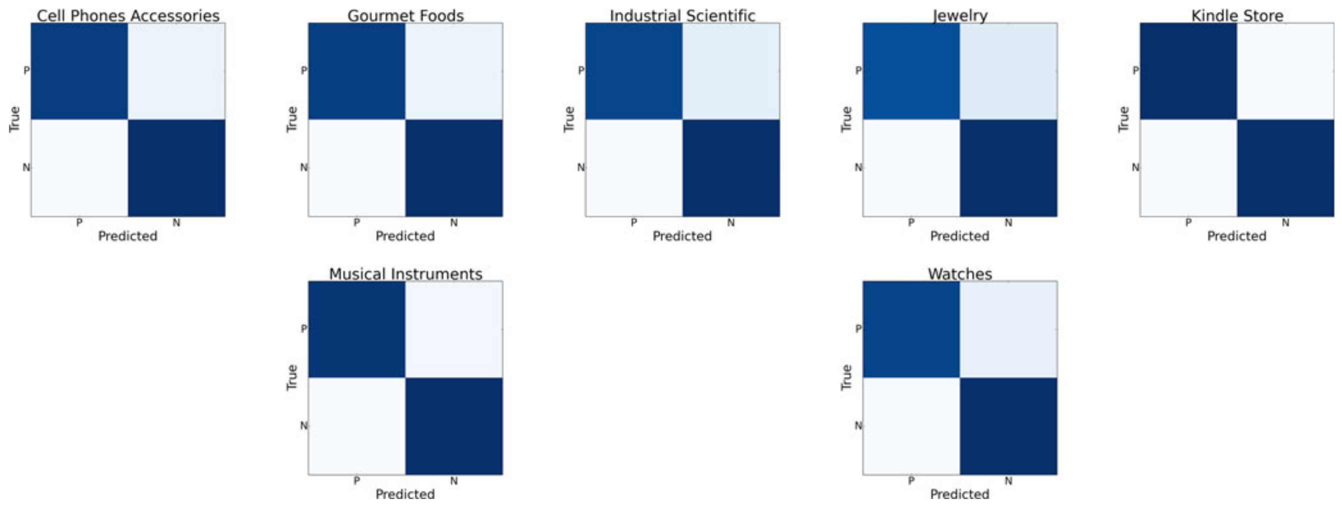


Fig. 7. Confusion matrices for the predictions on the Out-Model Domains.

respect to NeuroSent. On the contrary, for all the other domains, our approach outperforms all the baselines of more than 5 percent.

In this second evaluation, it is also possible to notice how the IRMUDOSA baseline obtained a higher precision with respect to the GWE one. By considering that also the IRMUDOSA system implements a polarity computation algorithm based on the aggregation of domain polarities, this outcome confirms that a solution based on exploiting overlapping information coming from different domains is an effective approach that has to be investigate in depth in the future. Thus, we may state that the exploitation of domain linguistic overlaps is a suitable solution for compensating the possible lack of knowledge when limited training sets are used for building opinion models.

Finally, we want to report some considerations about the efficiency of the platform. The time required for building the entire sentiment model was around 8.5 hours (single core equivalent) on a server equipped with a double Xeon X5650 and 32 Gb of RAM. While, during the testing session, on the same machine, the computation of a single document polarity required an average of 658 ms. This result supports the possible implementation of the polarity computation component as a real-time service due to the low time required for computing document polarity.

6.3 Error Analysis

We performed a detailed error analysis concerning the performance of the proposed strategy. Figs. 6 and 7 report the set of confusion matrices for the IMD and the OMD test sets, respectively. In general, we can observe how our strategy tends to provide false negative predictions. An in depth analysis of some incorrect predictions highlighted that the embedded representations of some positive opinion words are very close to the space region of negative opinion words. Even if we may state that the confidence about positive predictions is very high, this scenario leads to have a predominant negative classification for borderline instances.

On the one hand, a possible action for improving the effectiveness our strategy is to increase the granularity of the embeddings (i.e., augmenting the size of the embedding vectors) in order to increase the distance between the positive and negative polarities space regions. On the other hand, by increasing the size of embedding vectors, the computational time for building, or updating, the model and for evaluating a single instance increases as well. Part of the future work, will be the analysis of more efficient neural network architectures able to manage augmented embedding vectors without negatively affecting the efficiency of the platform.

7 Conclusion and Future Work

In this paper, we presented NeuroSent: a tool for multi-domain sentiment analysis exploiting linguistic overlaps between domains for inferring document polarity. The tool implements a deep learning architecture using distributed vectors to represent words.

Models are built by using a recurrent neural network trained with information extracted from the Blitzer dataset, a small but opinion-oriented collection of user-generated reviews. The inference of the overall polarity of a document is performed by combining belonging degrees of a document to each domain and domain-specific polarities scores computed by the model.

The performance of the system has been evaluated by applying the Dranziera protocol. Results shown the effectiveness of the proposed approach with respect to the baselines demonstrating its viability. Moreover, the protocol used for the evaluation enables an easy reproducibility of the experiments and the comparison with other systems.

Future work will focus on two main directions: (i) the integration of knowledge embeddings and (ii) the injection of fuzzy logic for representing uncertainty associated with word embeddings. Concerning the integration of knowledge embeddings, we want to build embeddings describing complex sentiment patterns instead of single terms (or concepts). This way, the system will be able to learn more complex linguistic constructs with the aim of improving the overall classification effectiveness. While, concerning the injection of fuzzy logic, our intent is to represent each embedding's feature by using fuzzy sets. On the one hand, this solution allows to manage uncertainty associated with each feature; however, on the other hand, the complexity of the overall architecture will increase. The challenge will focus on finding an efficient way to manage all such information. Some effort will be also focused on the creation of domain-dependent sentiment lexicons and on validating their effectiveness with respect to general purpose ones.

Finally, we foresee the integration of a concept extraction approach in order to provide the system with further semantic capabilities, for example the extraction of finer-grained information, that can be used during the model construction.

References

[1] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research [review article]," *IEEE Comp. Int. Mag.*, vol. 9, no. 2, pp. 48–57, May 2014. [Online]. Available: http://dx.doi.org/10.1109/MCI.2014.2307227

[2] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proc. ACL-02 Conf. Empirical Methods Natural Language Process.*, Jul. 2002, pp. 79–86.

[3] B. Liu and L. Zhang, "A survey of opinion mining and sentiment analysis," in *Mining Text Data*, C. C. Aggarwal and C. X. Zhai, Eds. Berlin, Germany: Springer, pp. 415–463, 2012.

[4] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*, 2007. [Online]. Available: http://aclweb.org/anthology-new/P/P07/P07-1056.pdf

[5] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 751–760. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772767

[6] E. Cambria, "Affective computing and sentiment analysis," *IEEE Intell. Syst.*, vol. 31, no. 2, pp. 102–107, Mar/Apr. 2016. [Online]. Available: http://dx.doi.org/10.1109/MIS.2016.31

[7] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc.42nd Annu. Meeting Assoc. Comput. Linguistics*, 2004, pp. 271–278. [Online]. Available: http://aclweb.org/anthology-new/P/P04/

[8] N. Jakob and I. Gurevych, "Extracting opinion targets in a single and cross-domain setting with conditional random fields," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2010, pp. 1035–1045. [Online]. Available: http://www.aclweb.org/anthology/D10-1101

[9] W. Jin, H. H. Ho, and R. K. Srihari, "OpinionMiner: A novel machine learning system for web opinion mining and extraction," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 1195–1204. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557148

[10] B. Liu, M. Hu, and J. Cheng, "Opinion observer: Analyzing and comparing opinions on the web," in *Proc. 14th Int. Conf. World Wide Web*, 2005, pp. 342–351. [Online]. Available: http://doi.acm.org/10.1145/1060745.1060797

[11] Y. Wu, Q. Zhang, X. Huang, and L. Wu, "Phrase dependency parsing for opinion mining," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2009, pp. 1533–1541. [Online]. Available: http://www.aclweb.org/anthology/D09-1159

[12] Q. Su, X. Xu, H. Guo, Z. Guo, X. Wu, X. Zhang, B. Swen, and Z. Su, "Hidden sentiment association in Chinese web opinion mining," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 959–968. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367627

[13] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," , Standford University, Stanford, CA 94305, Rep. no. CS224N, 2009.

[14] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proc. COLING 23rd Int. Conf. Comput. Linguistics*, Aug. 2010, pp. 36–44. [Online]. Available: http://aclweb.org/anthology-new/C/C10/C10-2005.pdf

[15] E. Cambria and A. Hussain, *Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis*. Berlin, Germany: Springer, 2015.

[16] Q. F. Wang, E. Cambria, C. L. Liu, and A. Hussain, "Common sense knowledge for handwritten Chinese recognition," *Cognitive Comput.*, vol. 5, no. 2, pp. 234–242, Jun. 2013.

[17] E. Cambria and A. Hussain, "Sentic album: Content-, concept-, and context-based online personal photo management system," *Cognitive Comput.*, vol. 4, no. 4, pp. 477–496, Dec. 2012.

[18] A. Gangemi, V. Presutti, and D. R. Recupero, "Frame-based detection of opinion holders and topics: A model and a tool," *IEEE Comp. Int. Mag.*, vol. 9, no. 1, pp. Feb. 20–30, 2014. [Online]. Available: https://doi.org/10.1109/MCI.2013.2291688

[19] D. R. Recupero, V. Presutti, S. Consoli, A. Gangemi, and A. G. Nuzzolese, "Sentilo: Frame-based sentiment analysis," *Cognitive Comput.*, vol. 7, no. 2, pp. 211–225, 2015. [Online]. Available: https://doi.org/10.1007/s12559-014-9302-z

[20] D. Bollegala, D. J. Weir, and J. A. Carroll, "Cross-domain sentiment classification using a sentiment sensitive thesaurus," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1719–1731, Aug. 2013.

[21] R. Xia, C. Zong, X. Hu, and E. Cambria, "Feature ensemble plus sample selection: Domain adaptation for sentiment classification," *IEEE Int. Systems*, vol. 28, no. 3, pp. 10–18, May/Jun. 2013.

[22] N. Ponomareva and M. Thelwall, "Semi-supervised versus cross-domain graphs for sentiment analysis," in *Recent Advances Natural Language Process.*, 571–578, 2013. [Online]. Available: http://aclweb.org/anthology/R/R13/R13-1075.pdf

[23] A. C.-R. Tsai, C.-E. Wu, R. T.-H. Tsai, and J. Y. jen Hsu, "Building a concept-level sentiment dictionary based on commonsense knowledge," *IEEE Int. Systems*, vol. 28, no. 2, pp. 22–30, Mar. 2013.

[24] M. Dragoni, A. G. B. Tettamanzi, and C. da Costa Pereira, "Propagating and aggregating fuzzy polarities for concept-level sentiment analysis," *Cognitive Comput.*, vol. 7, no. 2, pp. 186–197, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1007/s12559-014-9308-6

[25] M. Dragoni, "Shellfbk: An information retrieval-based system for multi-domain sentiment analysis," in *Proc. 9th Int. Workshop Semantic Evaluation*, Jun. 2015, pp. 502–509.

[26] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, Institut Für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München," 1991.

[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *CoRR*, vol. abs/1404.7828, 2014. [Online]. Available: http://arxiv.org/abs/1404.7828

[28] Y. Bengio, "Learning deep architectures for ai," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1561/2200000006

[29] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Resources*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[30] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Resources*, vol. 12, pp. 2493–2537, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078186

[31] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *Proc. IEEE Int. Conf. Neural Netw.*, 1996, vol. 1, pp. 347–352.

[32] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 129–136.

[33] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Proc. 14th Annu. Conf. Int. Speech Commun. Assoc.*, 2013, pp. 3771–3775. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2013/i13_3771.html

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] W. Zaremba and I. Sutskever, "Learning to execute," *CoRR*, vol. abs/1410.4615, 2014. [Online]. Available: http://arxiv.org/abs/1410.4615

[36] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[37] G. Petrucci, C. Ghidini, and M. Rospocher, "Ontology learning in the deep," in *Proc. Knowl. Eng. Knowl. Manag.: 20th Int. Conf.*, 2016, pp. 480–495.

[38] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom, "Learning to transduce with unbounded memory," *CoRR*, vol. abs/1506.02516, 2015. [Online]. Available: http://arxiv.org/abs/1506.02516

[39] K. M. Hermann and P. Blunsom, "A simple model for learning multilingual compositional semantics," *CoRR*, vol. abs/1312.6173, 2013, http://arxiv.org/abs/1312.6173

[40] I. Chaturvedi, E. Cambria, and D. Vilares, "Lyapunov filtering of objectivity for spanish sentiment model," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 4474–4481. [Online]. Available: https://doi.org/10.1109/IJCNN.2016.7727785

[41] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Natural Language Process.*, Oct. 2013, pp. 1631–1642.

[42] T. Chen, R. Xu, Y. He, Y. Xia, and X. Wang, "Learning user and product distributed representations using a sequence model for sentiment analysis," *IEEE Comput. Int. Mag.*, vol. 11, no. 3, pp. 34–44, Aug. 2016. [Online]. Available: https://doi.org/10.1109/MCI.2016.2572539

[43] S. Poria, E. Cambria, and A. F. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," *Knowl.-Based Syst.*, vol. 108, no. C, pp. 42–49, 2016. [Online]. Available: https://doi.org/10.1016/j.knosys.2016.06.009

[44] L. Oneto, F. Bisio, E. Cambria, and D. Anguita, "Statistical learning theory and ELM for big social data analysis," *IEEE Comput. Int. Mag.*, vol. 11, no. 3, pp. 45–55, Aug. 2016. [Online]. Available: https://doi.org/10.1109/MCI.2016.2572540

[45] M. Dragoni, A. G. B. Tettamanzi, and C. da Costa Pereira, "DRANZIERA: An evaluation protocol for multi-domain opinion mining," in *Proc. 10th Int. Conf. Language Resources Evaluation*, May 2016, pp. 267–272.

[46] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford corenlp natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, 2014, pp. 55–60. [Online]. Available: http://aclweb.org/anthology/P/P14/P14-5010.pdf

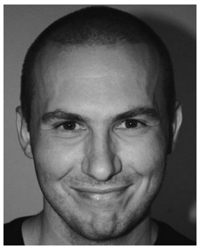
[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[48] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[49] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002. [Online]. Available. http://mallet.cs.umass.edu

[50] G. Petrucci and M. Dragoni, "The IRMUDOSA system at ESWC-2016 challenge on semantic sentiment analysis," in *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*, ser. Communications in Computer and Information Science, H. Sack, S. Dietze, A. Tordai, and C. Lange, Eds., Berling, Germany: Springer, vol. 641, 2016, pp. 126–140. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46565-4_10

Mauro Dragoni received the PhD degree from the University of Milan, in 2010. He is a researcher at the Process and Data Intelligence Unit, Fondazione Bruno Kessler, Trento, Italy. His main research topics concerns knowledge management, cognitive computing, information retrieval, and machine learning by focusing on the development of real-world prototypes as outcome of his research activities.



Giulio Petrucci is working toward the PhD degree at the University of Trento, Trento, Italy. He is carrying his research in the Data and Knowledge Management and the Process and Data Intelligence Units in Fondazione Bruno Kessler, Trento, Italy. His main research focus is on deep learning techniques for knowledge extraction from natural language text.