International Multidisciplinary Research Journal 2020, 10: 18-24 Special Issue on “Ricerca è Sviluppo 2020” Doi: 10.25081/imrj.2020.v10.6476

Investigating the role of Word Embeddings in sentiment analysis

Samuele Martinelli

smartinelli@uninsubria.it
Department of Theoretical and Applied Science, Insubria University, Varese, Italy

Gloria Gonella

ggonella@uninsubria.it
Department of Theoretical and Applied Science, Insubria University, Varese, Italy

Dario Bertolino

dbertolino@studenti.uninsubria.it
Department of Theoretical and Applied Science, Insubria University, Varese, Italy

Abstract

During decades, Natural language processing (NLP) expanded its range of tasks, from document classification to automatic text summarization, sentiment analysis, text mining, machine translation, automatic question answering and others. In 2018, T. Young described NLP as a theory-motivated range of computational techniques for the automatic analysis and representation of human language. Outside and before AI, human language has been studied by specialists from various disciplines: linguistics, philosophy, logic, psychology. The aim of this work is to build a neural network to perform a sentiment analysis on Italian reviews from the chatbot customer service. Sentiment analysis is a data mining process which identifies and extracts subjective information from text. It could help to understand the social sentiment of clients, respect a business product or service. It could be a simple classification task that analyses a sentence and tells whether the underlying sentiment is positive or negative. The potentiality of deep learning techniques made this simple classification task evolve, creating new, more complex sentiment analysis, e.g. Intent Analysis and Contextual Semantic Search [1].

1 Introduction

Over the years, the study of sentiment in human language has involved many areas including linguistics, philosophy, logic and psychology. This complex task, previously performed only by a human brain, can now be performed by an artificial intelligence algorithm. These algorithms are used in different sectors, such as finance, politics and marketing. In our work, we will show an example of application in the field of customer experience. Document classification is one of the first tasks that Natural Language Processing [2] has tried to solve; new solutions were based on a Knowledge Engineering (KE) approach. By building a classifier in such a way, all effort was dedicated to the construction of single classifier for a problem. In 2001, the research community dominant approach to text classification was based on machine learning (ML) techniques: a general inductive process automatically builds a classifier by learning, from a set of pre-classified documents, the characteristics of the categories [3]. The advantages of the ML approach over the

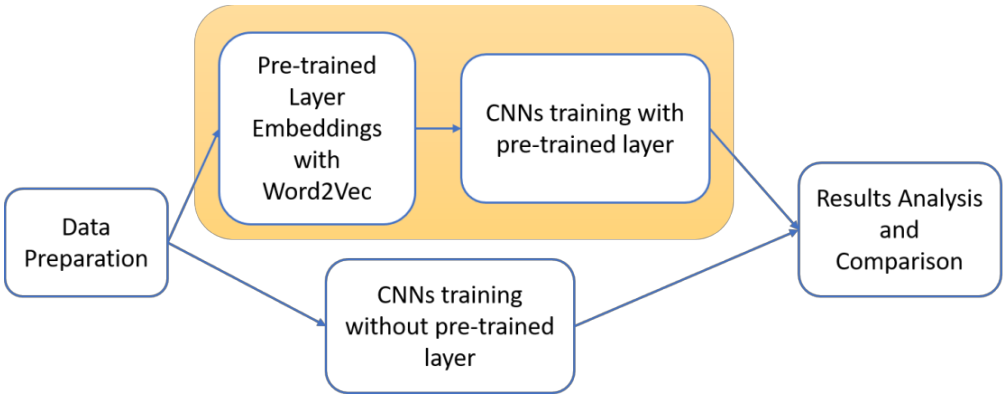

Figure 1: Visual overview of the experiment Workflow

KE are evident because the engineering effort goes toward the construction not of a classifier, but of an automatic builder of classifiers, the learner. This means that all that is needed are classified documents to make the automatic builder learn from them.

2 Sentiment analysis: experiment

The focus of the experiment will be on the presence of a pre-trained word embeddings layer in the network, as shown in Figure 1. The idea is to understand if such a layer, pre-trained on a considerable amount of text, can boost performances of a classifier, which cannot be training on many data, so the amount of reviews is right for this purpose. It was decided to maintain the classification problem as a binary one, positive or negative label.

2.1 Data preparation

Reviews from the chatbot service were unlabeled, so three different users labelled the comments, as shown in Figure 2. An Extract, Transform, and Load process (ETL) generate a dataset that can be used to train a neural network. However, the generated sets were not equal; for this reason, the ETL objectives are to select all labelled comments in the sets and then merge them into one unique set with average values as final labels. The entire process was implemented with a lightweight ETL framework for Python 3.5+, called Bonobo [4]. The custom data flow defined composed by three sub-jobs (one for each set of reviews), each of which selects all reviews with a label, clean the text deleting punctuation, backslashes, numbers, and others useless characters.

After the execution of this part of the ETL process, a dataset containing 1015 cleaned and labelled reviews is ready to be used. The next task consists of prepare the dataset to perform k-fold cross-validation, with k equals to 10. The performance of the model is then an approximation of the k models performances (P_i) trained and tested on different combinations of the dataset:

$$P_{tot} = \frac{1}{10} \sum_{i=1}^{10} P_i$$

The last step is to properly code the reviews so that they can be placed on a network. All sentences must have the same length, calculating the length of the sessions, an imbalance in the distribution of the lengths has been noticed, as described in Figure 3. For this reason, a standard length of 30 was chosen, reviews with more than 30 words were truncated while reviews with less than 30 words were filled with an internal spacing value. All the words have also been coded, mapping every single word to a numeric value. The coding operations were implemented using a Machine Learning framework called Tensorflow, which is an end-to-end open source platform [5]. The map containing the coded words is implemented with an object called Tokenizer.

SET ONE:	SET TWO:	SET THREE:
review one → 0	review one → 0	review one → 0
review two → 0	review two → 0	review two → n/a
review three → 0	review three → 0	review three → 0
review four→ 1	review four→ 1	review four→ 1
review five→ n/a	review five→ 0	review five→ 1
review six → 1	review six → 1	review six → 1
review seven→ 0	review seven→ 0	review seven→ 0
. . .	. . .	. . .

Figure 2: Three different users labelled the Dataset containing the reviews. The label "0" identifies a negative review, "1" identifies a positive review, "n/a" identifies not available because it has not been voted by the user. Each user voted a review according to his feeling with respect to the polarity (positive or negative) of the sentence.

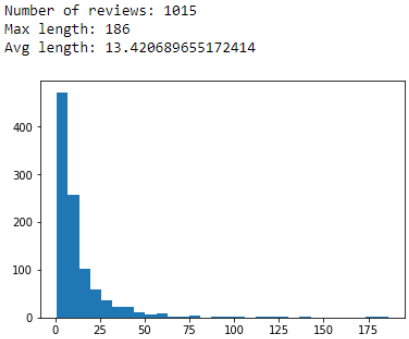

Figure 3: Histogram about chatbot reviews length

2.2 Pre-trained Layer Embeddings with Word2Vec

For the creation of embeddings, we relied on the algorithm word2vect. Word2Vec is a famous collection of two-layer neural networks invented in 2013 by T. Mikolov et al. [6]. These models process text and create word vector representations. The purpose and usefulness of Word2vec are to group the vectors of similar words together in a vector space, detecting similarities mathematically, as shown in Figure 4. It creates vectors that are distributed numerical representations of word features. Considering the context of a word as the other words that surround it, two models differ from the target to be predicted. The former called Continuous Bag of Words (CBOW), it uses the context to predict a target word. The latter is the reverse of CBOW, called SKIP-GRAM, it uses a word to predict the target context.

The structure of a skip-gram neural network to represent a vocabulary of 10’000 of words in a vector space of 300 dimensions is described in Figure 5. The input layer is composed of so many neurons as is the vocabulary size because every word in the input is a one-hot encoded vector.

The weights of the connections from the input layer to the hidden one represent the coordinates of every single word in the final vector space, as shown in Figure 6. Train the network means to update these weights, a process which results in moving the words within the vector space. Once the model is trained, the matrix of weights from the input layer to the hidden one contains all word vectors. This is the part of the network that can be extracted, saved and reused in other networks, even deep ones. The structure of the skip-gram Figure 5 model ends with a SoftMax layer as output, which contains so many neurons as is the vocabulary size, so each neuron corresponds again to a unique word.

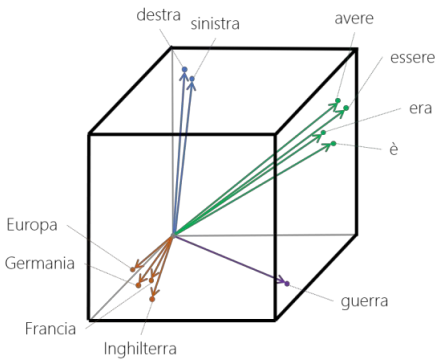

Figure 4: Graphic representation of some Italian words in the three-dimensional vector space where the colour of the vector identifies the same context.

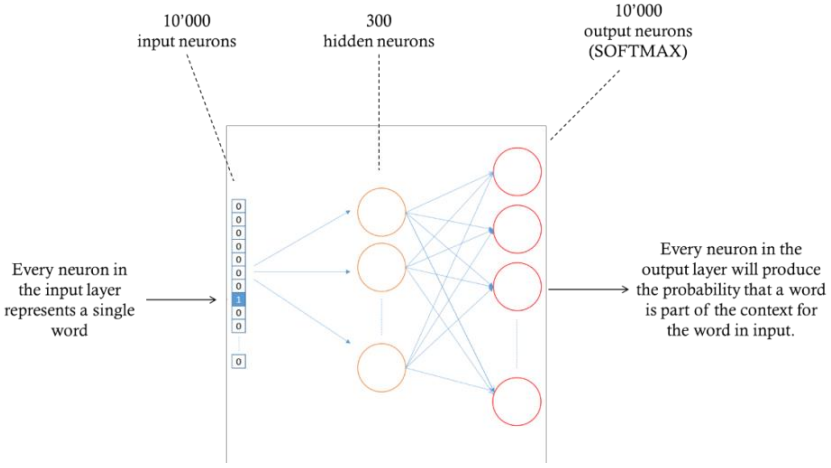

Figure 5: Structure of the Word2Vec skip-gram model for a vector space of 300 dimensions and a vocabulary of 10’000 words

We then carried out three different experiments, taking chatbot reviews, Twitter corpus and Wikipedia to build the embedding Matrix (code in section 2.4).

2.3 Pre-trained Embeddings in CNNs

The purpose of this experimentation is to find out whether a convolutional neural network can improve its performance by applying a layer of pre-trained word embeddings. The network has been implemented using the Keras library:

```
PAD_LEN = 30
EMBEDDING_DIM = 128

# define neural network
model = Sequential()
model.add(Embedding(vocabSize, EMBEDDING_DIM, weights=[embeddingMatrix],input_length=PAD_LEN))
model.add(Conv1D(128, 2, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

The architecture, shown in Figure 7, is taken from Y. Kim works on sentence classifications [7]. The input layer is a sentence comprised of word embeddings. A convolutional layer

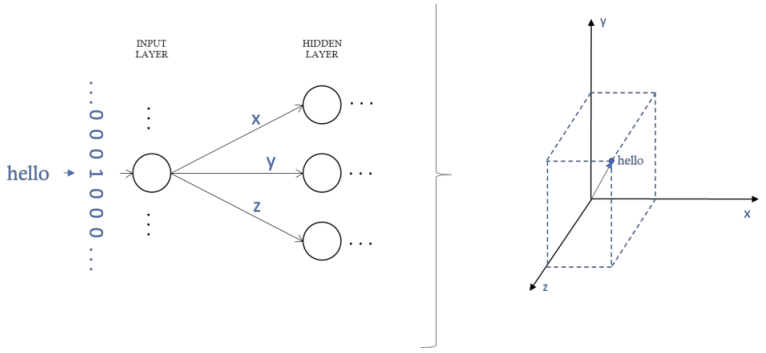

Figure 6: Graphical representation of a three dimensions vector space created by the weights from input to hidden layer in a Word2vec skip-gram model

with multiple filters follows that then a max-pooling layer, and finally the classifier neuron. The convolution operation, which is applied to a window of 2 words to produce a new feature, is applied to each possible window of words in the sentence to produce a feature map. Then the max-overtime pooling operation over the feature map extracts the maximum value as the feature for each filter.

The network captures the most important feature for each feature map. Metrics, optimizer and loss function remains the same as the previous network. Again, each model is tested applying the k cross-validation technique, training, and testing each model ten times on different combinations of the original dataset. All metrics showed in the graphs are the average value of the ten iterations. The three types of pre-trained embeddings (chatbot reviews, Twitter corpus, Wikipedia corpus) are again compared with randomly initialized layers of the same size (64, 128, 300). As shown in Figure 8, vectors created with SKIP-GRAM algorithm and chatbot reviews text, do not improve the CNN performance. While the recall of class 0 and precision of class 1 has been increased, the precision of class 0 and recall of 1 have been decreased. The network has not improved its ability to avoid misclassifications, and the overall accuracy almost remains at the same value (0,857 vs 0,856).

CNN performance has not been improved; the incorporation of Twitter and Wikipedia pre-trained layers led to an improvement in the CNN performance. As shown in Figure 9 and Figure 10, in both cases, the values of all the metrics have been improved, so the CNN enhanced its ability to avoid misclassifications for each class and the overall accuracy. The best results until now have been achieved with the Wikipedia pre-trained layer, although the Twitter dataset is composed of short sentences like the reviews. Moreover, the Twitter vectors set is missing only 379 words of reviews vocabulary (4617 words), against the 780 missing words for the Wikipedia vectors set. The last difference between Twitter vectors and Wikipedia ones is the model used to generate them, respectively, SKIP-GRAM and CBOW. So, this result suggests that the SKIP-

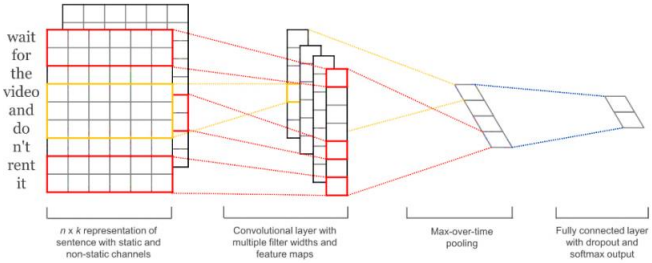

Figure 7: CNN architecture from Y. Kim work on sentence classification [7]

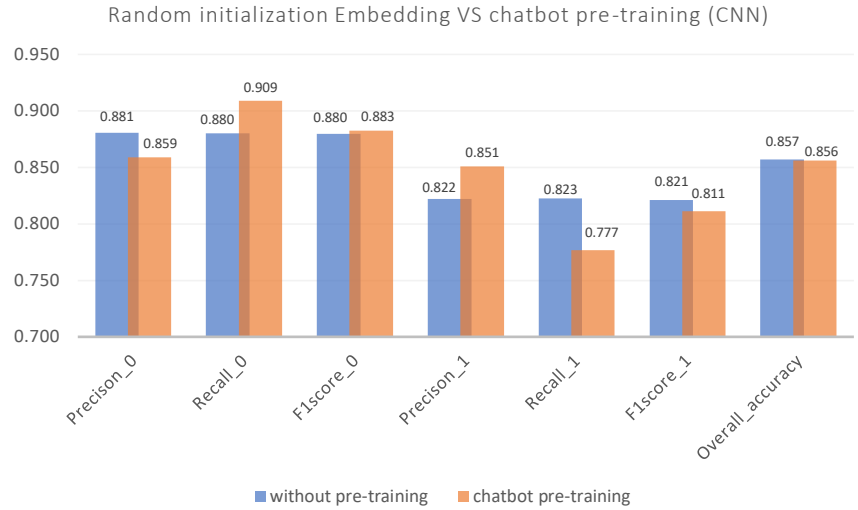

Figure 8: CNN performances comparison between a random initialized embedding layer and pre-trained one on 2789 chatbot reviews

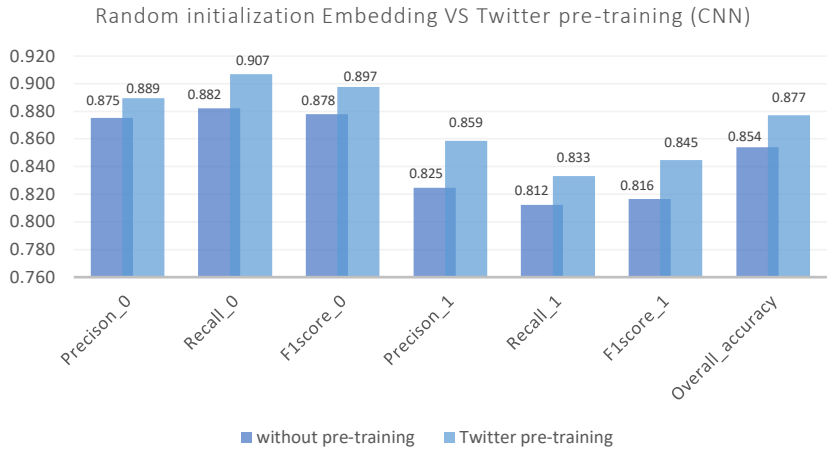

Figure 9: CNN performances comparison between a random initialized embedding layer and pre-trained one on big Twitter corpus

GRAM algorithm could be a better choice to create word features to be used in sentence classification tasks.

2.4 Results discussion

The incorporations of pre-trained word vectors have led to performance improvements in CNNs. The convolutions applied to 2 words at a time seems to take advantage of the pre-training of word embeddings coming from the previous layer. Word2Vec models need to be trained on a considerable amount of text in order to produce excellent features. Vectors trained on 2789 reviews with a skip-gram algorithm did not produce improvements in any case, and the distances between word vectors were not optimized at all. The sentiment classification task that has been performed is probably too simple to observe significant differences between randomly initialized layers and pre-trained ones. Metrics have always varied no more than 5-10% and all models, with and without pre-trained layers, always achieved an overall accuracy between 80% and 90%.

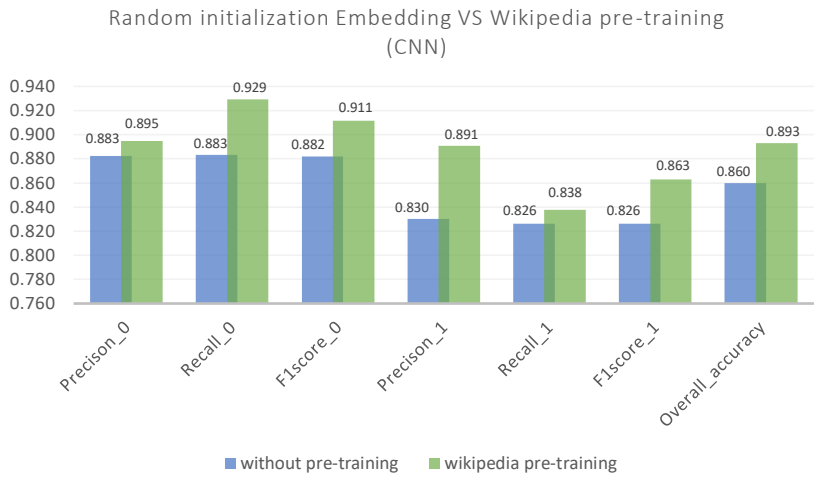

Figure 10: CNN performances comparison between a random initialized embedding layer and pre-trained one on a big Wikipedia corpus

3 Conclusions

The sentiment analysis test was performed on a binary classification using a tiny dataset. One future objective certainly is to find out if pre-trained word vectors can improve CNNs performance even if the complexity of the task is increased, adding a neutral class or more sentiments to recognize besides positive and negative.

References

[1] S. Gupta, “Sentiment Analysis: Concept, Analysis and Applications,” 2018. [Online]. Available: https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17.

[2] T. Young, D. Hazarika, S. Poria and E. Cambria, “Recent Trends in Deep Learning Based Natural Language Processing,” arXiv.org, 2018.

[3] F. Sebastiani, “Machine Learning in Automated Text Categorization,” Consiglio Nazionale delle Ricerche, Italy, 2002.

[4] R. Dorgueil, “Bonobo project,” [Online]. Available: https://www.bonobo-project.org.

[5] Open-source, “An end-to-end open source machine learning platform,” [Online]. Available: https://www.tensorflow.org.

[6] T. Mikolov, “Efficient Estimation of Word Representations in Vector Space,” 2013.

[7] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” arXiv, New York University, 2014.

[8] M. Koppel and J. Schler, “The Importance of Neutral Examples for Learning Sentiment,” Computational Intelligence, 2006.