Journal of Intelligent & Fuzzy Systems 42 (2022) 4599–4610
DOI:10.3233/JIFS-219248
IOS Press

A content spectral-based text representation

Melesio Crespo-Sanchez[a], Ivan Lopez-Arevalo[a,*], Edwin Aldana-Bobadilla[b] and Alejandro Molina-Villegas[c]
[a]*Centro de Investigación y de Estudios Avanzados del I.P.N. Unidad Tamaulipas, Victoria, Mexico*
[b]*Conacyt - Centro de Investigación y de Estudios Avanzados del I.P.N. Unidad Tamaulipas, Victoria, Mexico*
[c]*Conacyt - Centro de Investigación en Ciencias de Información Geoespacial, Merida, Mexico*

Abstract. In the last few years, text analysis has grown as a keystone in several domains for solving many real-world problems, such as machine translation, spam detection, and question answering, to mention a few. Many of these tasks can be approached by means of machine learning algorithms. Most of these algorithms take as input a transformation of the text in the form of feature vectors containing an abstraction of the content. Most of recent vector representations focus on the semantic component of text, however, we consider that also taking into account the lexical and syntactic components the abstraction of content could be beneficial for learning tasks. In this work, we propose a content spectral-based text representation applicable to machine learning algorithms for text analysis. This representation integrates the spectra from the lexical, syntactic, and semantic components of text producing an abstract image, which can also be treated by both, text and image learning algorithms. These components came from feature vectors of text. For demonstrating the goodness of our proposal, this was tested on text classification and complexity reading score prediction tasks obtaining promising results.

Keywords: Text representation, text analysis, content spectre

1. Introduction

Nowadays many text analysis tasks have been used in a recurring manner to address real-world issues such as classification, sentiment analysis, text summarizing, text generation, automatic translation, hate detection, among others [1]. Many of these tasks resort to the use of machine learning algorithms to find patterns on texts [2]. Usually, these algorithms require numeric representations of text. Recently the most used representations are in the form of feature vectors of words, paragraphs, or entire documents, which can be extracted by several means according to the needs of the problem. These types of text representations emphasize on abstracting the semantics of documents, which is one of the most important features for handling text [3–5]. However, sometimes these representations put aside other types of content information such as lexical and syntactic by considering them in a partial way. Such information could help to improve the results of the learning models in which they are used.

Several works have exploited the semantics on text for clustering [6], topic modeling [7], or classification [6] by using word-embeddings [8]. While the usefulness of the semantic component has been demonstrated with this approach, the integration of lexical and syntactic components in a same representation could be an advantage to enhance the performance of this representation. We also can see that given the nature in which these embeddings are obtained, typically by means of deep neural network techniques, these modern approaches are black boxes, in which the user does not have control or visual interpretation on how these models build these embeddings at their outputs.

*Corresponding author. Ivan Lopez-Arevalo, Centro de Investigación y de Estudios Avanzados del I.P.N. Unidad Tamaulipas, 87130 Victoria, Mexico. E-mail: ilopez@cinvestav.mx.

ISSN 1064-1246/$35.00

In this work we propose a novel text representation, which includes the three components of text together: 1) the lexical component, associated with the vocabulary of texts, from which the possible topic of text can be derived, 2) the syntactic component, associated with how do words structure a text, denoting the writing style, and 3) the semantic component, associated with the meaning of text, relating words based on the contexts where these ones occur [9]. Our proposal is capable to take into account these components in a fully, partial, or marginal manner. By taking into account these three components of text together into a single representation, we found that better results can be obtained in machine learning algorithms for text analysis. As part of this proposal, we made a comparison of combinations of components of text to know the gain of use them. So, briefly, the contributions of this work are:

- A novel text representation integrating the spectra of lexical, syntactic, and semantic components of text. Such spectra can be used in a joint, partial, or marginal manner.
- A comparison study testing all combinations of the components of text to asses the performance of two machine learning algorithms.

The rest of this paper is structured as follows. In Section 2 some text representations are described, as well as the strategy they follow to extract features from text. In Section 3 the proposed methodology to obtain our text representation is described. In Section 4 the experimentation is reported by means of two case studies, in these the text representation was tested and statistical significance tests were obtained to determine the goodness of the proposal. Finally, in Section 5 the conclusions of the proposed representation and some remarks of the experimentation are given.

2. Related work

The most important aspect of using machine learning algorithms in text analysis tasks is the representation of text; most of times, it is required a numerical representation. These numerical representation abstract some components of text (lexical, syntactic, and semantic). This section describes some of these representations of text, as well as the main characteristics of each one.

Early text representations considered as *traditional* take into account that documents from a corpus can be modeled in a vector space of the form $\vec{x} = (x_1, x_2, \ldots, x_{d-1}, x_d)$, where d corresponds to the number of words in the corpus' vocabulary. In this sense, a matrix of size $n \times d$ is build where n corresponds to the number of documents in the corpus. The value assigned to the *th*-column depends on the approach being used. The most simple one is in which a value of 1 is assigned to the column if the word is present in the document, 0 otherwise. This approach is known as the *one-hot encoding* [10]. Since this approach is hard discriminating, the value 1 assigned to the *th*-column in the document vector was replaced by the well known *TF-IDF* approach, which takes into account the frequency of the word in the document and a normalization of the frequency of the same word in the whole corpus. Also, more columns were considered in the building of the document vector by adding windows of n words (n-grams) [10]. This type of representations were good in the sense that they considered lexical information, and also some sort of syntactic information by using n-grams. Semantic approaches later gained ground in the state of the art.

Deerwester et al. [3] proposed the Latent Semantic Analysis (LSA) approach as an information retrieval technique in which documents were grouped according to their lexical and semantic content. The authors built a term-document matrix in which each row corresponded to a word in the corpus, and each column represented each document, forming vectors. The intersection between a row and a column in the matrix contained the frequency of the word given its corresponding document. From this matrix, the next step was to build a "semantic" space where words and related documents were placed in close positions in the vector space. For this, the authors used singular value decomposition (SVD) to reduce rows and highlight the most relevant associative patterns in documents, ignoring the less relevant ones. From this reduced semantic space, document vectors can be retrieved to obtain topics into the corpus.

Mikolov et al. [4] proposed two architectures to obtain continuous vectors of words, where the similarity relationships in the vector space are given by the semantic relationship between them, the well-known word embeddings. Authors used the concept of text window, known as n-gram, where a word was taken as target and the rest as context words. Then by using a neural network over the text, these n-grams are learned, capturing semantic information from text.

The first architecture defined as Continuous Bag of Words (CBOW) takes the context words as input,

trying to predict the target word at the output of the network; in the opposite case the second architecture, Skip-gram takes the target word to predict the context words. By this training approach, the generated vectors no longer depended on the size of the corpus vocabulary, since the columns of these vectors now depend on the definition of the model to be trained.

Pennington et al. [5] proposed Global Vectors (GloVe) as a way to improve some limitations of word embeddings such as local semantic information, since the generated vectors come from contexts of a target word in a given n-gram without taking into account how much semantic information the rest of words out of this one provide. In contrast, GloVe uses a word co-occurrence matrix to capture the semantic information of all words in a document, which denotes how all words contribute to the semantics of a particular target word. With this matrix, what is sought is to predict the co-occurrence probability of words in a given context. Authors used a weighted least squares model to produce this type of word representation. Given this co-occurrence scope, this representation is also based mainly on the lexical and semantic information of documents.

Proposals such as Mikolov et al. [4] and Pennington et al. [5] are good approaches to obtain word embeddings. Nevertheless, some tasks require not word but document embeddings for their solutions. Le and Mikolov [11] presented two variants of the word embeddings proposed by Mikolov et al. [4] by including a paragraph or document vector among the context words being trained. Such models are The Distributed Memory (PV-DM) and the Distributed Bag of Words (PV-DBOW), which can be considered as extensions of the CBOW and Skip-gram models respectively. We can find some other approaches in the literature to obtain document vectors such as works proposed by Pagliardini et al. [12], Kiros et al. [13] and Chen [14].

Peters et al. [15] proposed Embeddings for Language Models (ELMo), a representation of deeply contextualized word vectors modeling lexical and semantic features of words. The main difference of this representation against others such as the one of Mikolov et al. [4] is that each word is assigned to a representation that is based on a complete sentence. To obtain this type of embedding they resort to the use of a bidirectional LSTM (biLSTM) neural network. This type of language model has the particularity of having memory about the words that are before and after each other. By combining the internal states of the biLSTM network it is possible to obtain complex semantic representations, which are capable of solving problems such as word disambiguation.

Vaswani et al. [16] introduced the transformer model which became a milestone of modern language modeling techniques. Unlike previous proposals, this one moved away from convolutional and recurrence-based models to incorporate sequential attention mechanisms that highlight the relationship of words in a sentence to produce word and sentence vectors. Thanks to this proposal, transformers are able to speed up the learning process by incorporating lexical information, semantic information, and information about the position of words in sentences without having to perform inference with a recurrence-based approach such as ELMo. This set of algorithms is very good for solving problems such as word prediction, language modeling, question answering, among others. In this family of techniques, we can find several variants among which some of the most popular can be considered BERT [17], GPT-1 [18], GPT-2 [19], and GPT-3 [20].

In the area of computer vision, some works have proposed algorithms combining data in a multimodal way. For example, those that use text and images to train generative models. Such is the case of Reed et al. [21], who proposed a model for image generation from textual descriptions of images using generative adversarial networks (GANs). Also Li et al. [22] proposed a similar idea by using this type of neural network by adding a word-level discriminator to correlate a text with its corresponding images. A similar approach was presented by Li et al. [23], authors included semantic information from the text to map sub-regions of the images during the model training. It is worth to mention that although these works propose to generate images from text, our approach is different because we focus on obtaining a representation of text outside of the area of computer vision. According to our knowledge, there is not a similar approach as ours.

As described in the previous works, the existing representations of text generally focus partially on the components of text (lexical, syntactic, and semantic). Some of these representations, which seems to leaving aside the already available information from text, in which case, we consider taking into account all the components of text could provide improvements, in terms of performance, of learning algorithms. We look to obtain a text representation which can be used where lexical, syntactic or semantic components are required.

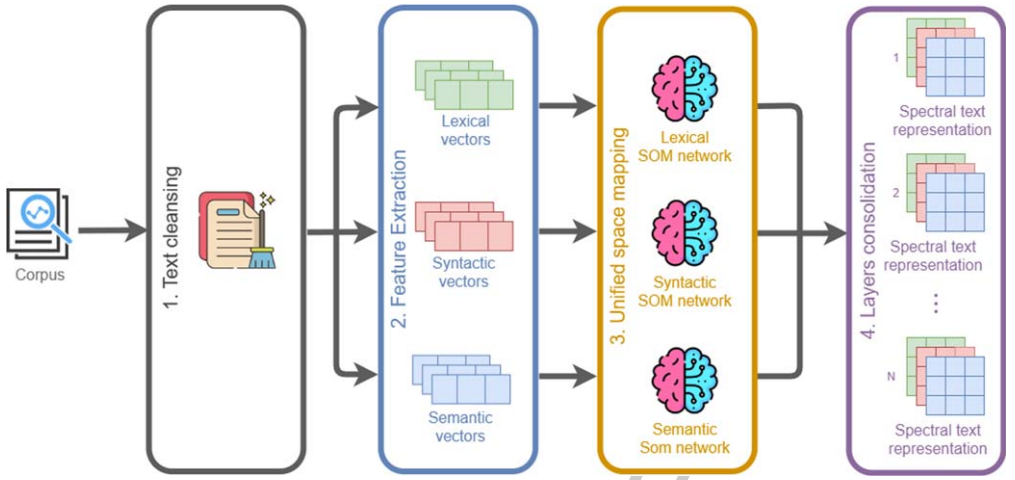

Fig. 1. Pipeline to obtain content spectral-based text representation composed of four stages 1) Text cleansing, where text pre-processing techniques are applied to obtain a clean corpus; 2) Feature extraction, where text is transformed into three sets of feature vectors, one for each text component; 3) Unified space mapping, where feature vectors serve as training data for Self Organizing Maps (SOM) networks to obtain content projections with a fixed size; and 4) Layers consolidation, where each the three projections of a single document are consolidated into a single image, which contains lexical, syntactic, and semantic information about the documents.

3. Text representation proposal

The research interest of this work was to address the gain of combining all or some of the three components of text. We show that these combinations exhibits a better performance in machine learning algorithms for text analysis. Our proposal does not assume prior knowledge about corpus that allow us to induce classes or partitions.

Let D be a corpus of unlabeled documents d, $\forall d \in D$ there exists a 2-dimensional matrix with fixed size that contains a projection of content of d, that we call *spectral-based text representation*. To obtain such representation, the text must pass in the pipeline of processes shown in Figure 1.

This pipeline is comprised of four main stages described as follows. First, the input corpus D is pre-processed to obtain a clean text from each document that serves as input to the next stage. In Subsection 3.1 the pre-processing techniques applied to D are described, where the used techniques vary depending on the component of text (lexical, syntactic, and semantic); this generates a set of pre-processed text for each component. In the second stage (feature extraction), these sets are transformed into vectors sets (one per each text component), which contain features of the content in every d. The size of these vectors depends on the component of text, this process is described in Subsection 3.2. In the third stage, every vector set is projected into a two-dimensional numerical space (matrix) according to the component of text (lexical, syntactic, and semantic), which denotes a specter that defines a pattern for each d, this process is described in Subsection 3.3. Finally, in Subsection 3.4 the three projections for a single d are consolidated into a single spectre (image).

3.1. Text cleansing

As mentioned, our proposal works over the three components of text. In this regard, according to the component in which we are working on, we apply some pre-processing techniques to D to generate three sets D^{lex}, D^{syn}, and D^{sem}, which correspond to each of component of text. The applied techniques are described below.

– **Lexical layer.** In this layer we are interested in the vocabulary used in D, which may reflect the type of information in documents, technical documents may use a specialized vocabulary, simpler documents may use a more generic one. The common pre-processing tasks for this component include: line breaks removal, special characters removal, extra spaces regularization, digit removal, punctuation symbols removal, stopwords removal, and lemmatization [24], which help us to reduce words to their minimum lexical value, this also helps us to reduce the size of the vocabulary. From this layer the set D^{lex} is produced.

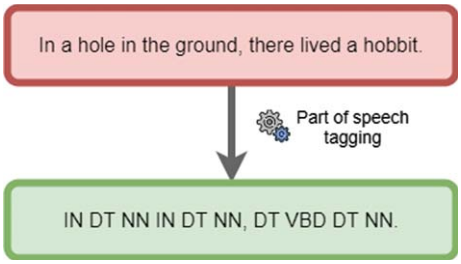

Fig. 2. Part-Of-Speech tagging example, where words of the input text are replaced with their corresponding part of speech tag.

– **Syntactic layer.** For this layer we want to extract those features that may remark the writing style used on each d. Thus, the used vocabulary is not as important for this component as in the lexical layer. We turn the original text into chains of part of speech (POS) tags, where we obtain the grammar category of each word for further feature extraction [25]. An example of this is shown in Figure 2. To obtain D^{syn}, the next pre-processing tasks were also applied: line breaks removal, special characters removal, extra spaces regularization, digit removal, punctuation symbols removal, stopwords removal.
– **Semantic layer.** Here we also use the pre-processing tasks of the lexical and syntactic layers to produce D^{sem}, these tasks are: line breaks removal, special characters removal, extra spaces regularization, digit removal, punctuation symbols removal, stopwords removal. Since we want to preserve the context of words to get as much semantic information as possible, we do not use any lemmatization neither POS tagging techniques in this layer.

3.2. Feature extraction

Given the pre-processed sets D^{lex}, D^{syn}, and D^{sem}, in this stage, we transform each of these sets into sets of feature vectors X^{lex}, X^{syn}, and X^{sem} respectively. Such sets abstract information about each component of text. In this regard, for each pre-processed set a different feature extraction process is applied. These used approaches are described below.

– **Lexical layer.** In this layer we abstract the words of each d. Commonly frequency-based approaches are applied for this task [26]. In these approaches, a feature matrix X is build from D, where each row corresponds to a document d and each column correspond to a word w in the vocabulary. As is expected, the size of X may grow to thousands of columns since the vocabulary may vary a lot. The value in the matrix corresponding to the intersection of row and columns represents a measure of frequency for a word w given a document d. To reduce the vocabulary of D we apply lemmatization. This task was performed in the text cleansing stage to obtain D^{lex}, which in consequence also reduces the number of columns in the resultant X^{lex}.

A common problem in this task is that stopwords may be highlighted given the frequent use of them in documents. Although we have removed most of them in the text cleansing stage, we want to highlight those words which frequency is low for each d; then, we remark the pattern of words usage in d. For this we use an information-based approach, where the value assigned to the intersection of a row and a column in matrix X^{lex} is given by the amount of information that w grants to d given its probability of occurrence ($p(w_j)$). Then, we build vectors of the form $\vec{x}_i = [I(w_0), I(w_1), \ldots, I(w_{j-1}), I(w_j)]$. The value computed for each column is given by the Shannon information content of an outcome w_j, such as Equation 1 describes [27].

$$I(w_j) = -log_2(p(w_j)) \tag{1}$$

where $I(w_j)$ is the amount of information that w_j gives to d, and w_j is a word in the vocabulary.
– **Semantic layer.** In this layer we desire document vectors close semantically, denoting similar meanings to create X^{sem}. We take advantage of the word embeddings approach for capturing semantic information about words and their contexts in every d.

In this regard, we apply the *Doc2Vec* algorithm [11]over D^{sem} by using the Distributed Memory Model of Paragraph Vectors (PV-DM) architecture. Nevertheless, also the Words version of Paragraph Vector (PV-DBOW) architecture could be used. The Doc2Vec algorithm obtains continuous fixed sized vectors from variable-length pieces of text (from a single word to paragraphs or entire documents) by concatenating a vector that represents d with its corresponding word vectors.
– **Syntactic layer.** During the text cleansing stage, we use a POS tagging algorithm to transform the

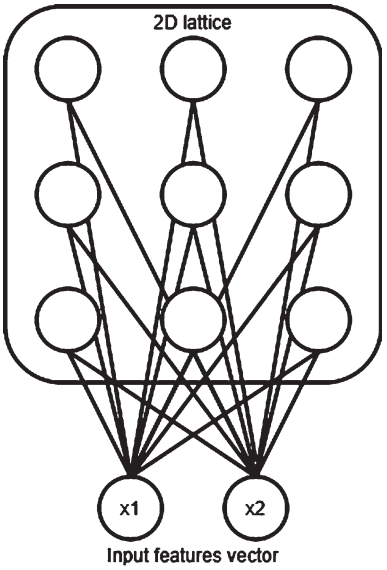

Fig. 3. Self-Organizing Map with a two-dimensional input vector and a two-dimensional lattice.

original words into chains of POS tags (D^{syn}). In this manner, we put aside the lexical and semantic information to focus on syntactic content that could give us information about how words are structured and highlight the writing style of the documents. Since the Doc2Vec algorithm makes use of the words surrounding a target word, we assume that it also stores somehow syntactic information about the content in d. Then, changing the vocabulary of D for its corresponding set of POS tags, it is possible that the Doc2Vec algorithm focuses more on syntactic rather than semantic information. Thus, from D^{syn} we obtain X^{syn} by using the Doc2Vec algorithm.

3.3. Unified space mapping

Given the nature of methods to obtain the sets of feature vectors X^{lex}, X^{sin}, and X^{sem}, such sets have a different number of dimensions, which is a problem if we want to put together the three components of text into a single representation. It is necessary to map them to a space in the same dimensions in that manner they hold their similarity features. To solve this problem we resort to the Self-Organized Map (SOM) approach [28]. In this two-layer neural network, the input layer can receive different-size vectors, and the output layer produces a feature map (SOM lattice) of a fixed size; as is illustrated in the Figure 3.

As can be seen, each neuron in the output layer has a vector of weights $\vec{\omega}$ with the same number of dimensions as the input layer. Thus, we are able to map a vector with any number of dimensions to a fixed-sized matrix (SOM lattice). The training of this kind of neural network is as follows: During training, $\vec{\omega}$ are adjusted by a competitive method in which one of the $\vec{x}$ is presented to the input layer, then a neuron called the best matching unit (BMU) is selected. After this, the neurons in the same lattice neighborhood are selected, and the $\vec{\omega}$ of these neurons are adapted to resemble $\vec{x}$ through a learning function. After this, another $\vec{x}$ is presented with a smaller BMU neighborhood size. This process is repeated during a defined number of iterations.

After the SOM training stage we can see how all $\vec{\omega}$ in the SOM lattice tend to fit the distribution of data in the input vectors $\vec{x}$, an example of this is shown in Figure 4. Such a property of this neural network is favorable to preserve the similarity features of X^{lex}, X^{sin}, and X^{sem} in the new space.

Thus, at this stage we train three SOM models, one per each component of text, using the vector sets X^{lex}, X^{sin}, and X^{sem} as training sets. The size of the

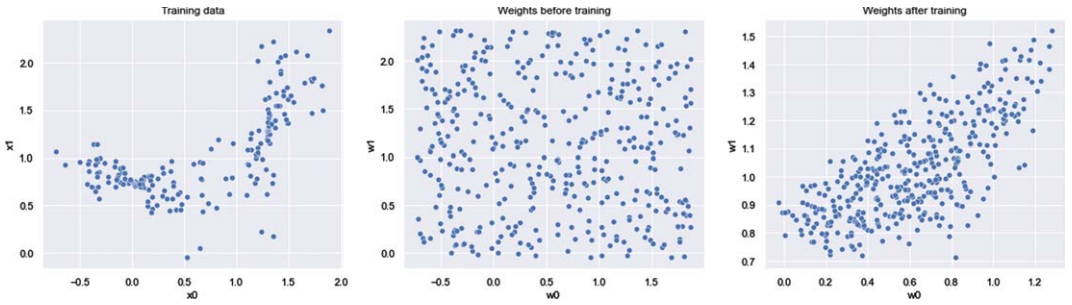

Fig. 4. Example of the distribution of the SOM's weights vectors $\vec{\omega}$ before and after training. On the left, a 2-dimensional synthetic training data set X is given, in the center the SOM's $\vec{\omega}$ before the training process, and on the right the SOM's $\vec{\omega}$ after the training process. Weight vectors $\vec{\omega}$ are fitted to X's distribution during the training.

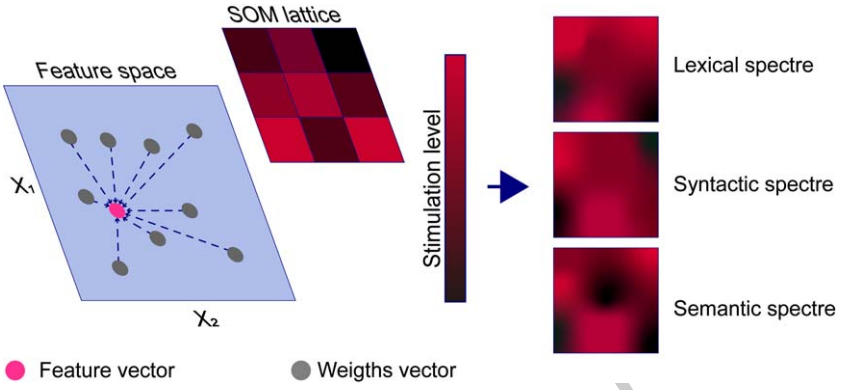

Fig. 5. Projection of a two-dimensional feature vector into a given previously trained SOM grid.

lattice of the three resulting models must be the same to carry out the projection of the training sets later.

Once the training of the three models has been performed, the next step is to make the projection of each vector in X^{lex}, X^{sin}, and X^{sem} in their corresponding lattice to obtain spectra, one per each layer of the same d, which contains lexical, syntactic, and semantic features of d. To obtain these projections, each vector in sets X^{lex}, X^{sin}, and X^{sem} are passed through its corresponding previously trained SOM model. By presenting each feature vector to its corresponding SOM model, each of the neurons in the SOM lattice is stimulated according to the similarity between the input vector against each of the weights vectors in the lattice, as is shown in Figure 5. The degree of stimulation for each neuron in the SOM lattice is computed by using the Equation 2.

$$f(\vec{x}, \vec{\omega}) = \frac{1}{\left(\sum_{i=0}^{n} |x_i - w_i|^p\right)^{\frac{1}{p}}} \quad (2)$$

where $\vec{x}$ is a feature vector of d in its corresponding component layer, $\vec{\omega}$ is the vector of weights for a given neuron in the SOM lattice, n is the number of dimensions of $\vec{x}$, and p is a type of distance definition (1 for Manhattan distance or 2 for Euclidean distance).

Since the training time of a SOM is directly proportional to the number of dimensions (d) of the training feature vectors and the size of the SOM lattice ($m \times m$), we can define the complexity of our proposal as a polynomial expression of the form $O(dm^2)$. Examples of the spectra obtained for some documents are shown in Figure 6, where each row corresponds to a given document and each column corresponds to their lexical, syntactic, and semantic components.

3.4. Layers consolidation

Up to this point, we have worked on each component of text from d separately, however, we desire a single representation containing the abstraction of the three components of text of d. For this, we consider each of these spectra as a channel of an image. Then, we concatenate the three spectra as a single image. In this manner, we pass a text to a representation, which can be treated as image or text.

4. Experiments and results

To determine if a machine learning technique is able to learn from the proposed text representation, this was tested in two case studies of text analysis: text classification and complexity reading score prediction. All the implementations were made by using the Keras[1] and Tensorflow tools[2]. Also, we used the POS tagger and lemmatization algorithms implemented in NLTK[3], as well as the Doc2Vec implementation of Gensim[4].

For both case studies, the tests were carried out on a server with 96 cores, 128 GB of RAM, 1 TB of hard disk, running CentOS 7 64-bit. Details about the used corpora and the test setup are next described.

In both cases, we transformed the test corpora into images of size 20×20 pixels by training its corresponding SOM networks and projecting the three document feature vectors. All these SOM models were trained with a learning rate of 0.01 and 1000

[1] keras.io [All webpages were visited on 10/30/2021]
[2] www.tensorflow.org
[3] www.nltk.org
[4] radimrehurek.com/gensim/models/doc2vec.html

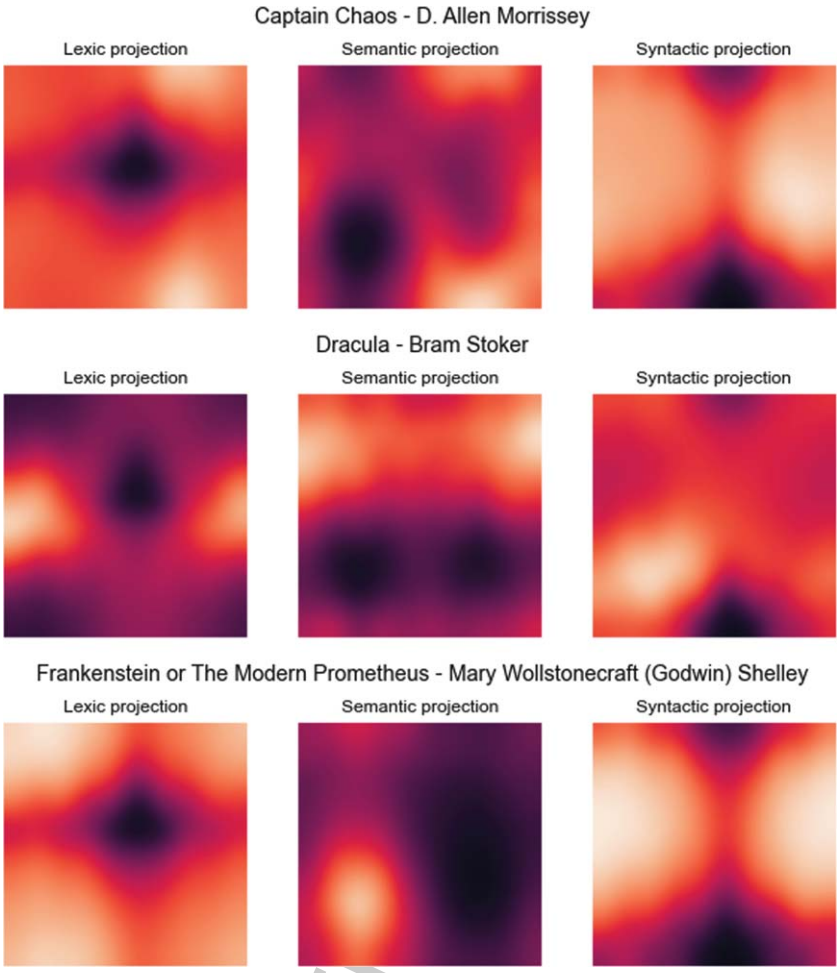

Fig. 6. Examples of the spectra obtained for three books, each row corresponds to a book and each column to its corresponding lexical, syntactic and semantic component.

epochs each. We used Euclidean distance to obtain the spectra from the SOM models in which case we set the parameter $p = 2$ for Equation 2. Also, the Doc2Vec models were trained to obtain vectors of size 100 by using windows of 5 words for both the syntactic and semantic components. The longest time registered to obtain the spectral representations of the documents was for case study 2 which took approximately 4050 seconds.

4.1. Case study 1: text classification

For this problem, we manually acquired books from the Project Gutenberg[5] to build the corpus. This corpus is made up of a total of 200 documents, as class label we considered the literary genre. The class labels collected were distributed as 48 Crime Fiction, 32 Demonology, 32 Horror Tales, 64 Humorous Stories, and 24 Western Stories documents.

Due to a text is mapped to a image (input matrix), we resort to a convolutional neural network as classification model with the following parameters in its architecture:

- A 1st convolution layer with 16 filters with size 3x3 pixels, 1x1 strides, and 0 pixels padding
- A ReLU activation function
- A max pooling layer of 2x2 pixels, 2x2 strides, and 0 pixels padding
- A 2nd convolution layer with 32 filters with size 3x3 pixels, 1x1 strides, and 0 pixels padding
- A ReLU activation function
- A max pooling layer of 2x2 pixels, 2x2 strides, and 0 pixels padding

[5] www.gutenberg.org

Table 1
Mean and standard deviation of accuracy for training and validation sets in text classification. By using all components of text (*Full*) better results were achieved

Used components	Training set		Validation set	
	μ	σ	μ	σ
Full	**0.8706**	0.0371	**0.5325**	0.0765
Lexical-Syntactic	0.7121	0.0376	0.4912	0.0687
Lexical-Semantic	0.8045	0.0402	0.5020	0.0707
Syntactic-Semantic	0.8579	0.0413	0.4820	0.0716
Lexical	0.4654	0.0270	0.4504	0.0626
Syntactic	0.6701	0.0495	0.4772	0.0703
Semantic	0.8067	0.0574	0.4926	0.0748

- A 3rd convolution layer with 64 filters with size 3x3 pixels, 1x1 strides, and 0 pixels padding
- A ReLU activation function
- A flatten layer
- A fully connected layer with 128 neurons
- A ReLU activation function
- A dropout with a probability of 0.4
- A fully connected layer with 6 neurons one per class in the test corpus
- A softmax output layer to transform the output into a probability distribution of getting a class

For training, a total of 100 iterations of 150 epochs each were performed by using our representation proposal. Likewise, we applied the k-fold strategy with $k = 5$ to have 80% of the data as a training set, and 20% as a validation set for each series of experiments. We computed the accuracy as metric to guide the model during the training process for a better classification. A better classification model or network architecture could be used in this case, but this is out of the scope of this work.

In the same way, to determine if by using all the components of text in our approach gave better results rather than using combinations of them. This series of experiments were repeated taking as input the spectra of three components in a single image, two components, and a single component.

Given the number of executions per experiment series, and following the central limit theorem, we assumed our results fit a normal distribution. Then, the parameters of results per combination of component spectra for the training and validation are shown in Table 1, where the better results were obtained by using all three text components as a whole. *Full* means we used the three components together.

We conducted a t-test to determine if there are significant differences between the performance of using all components of text and the performance exhibited by using them with one or two them.

Table 2
t-test results over the classification results. In all cases the p-value $< .05$, which reflects that there is a significant difference of accuracy between using all components of text and using one or two of them. Confidence intervals of the difference between means also reflect a positive difference in all cases showing also that higher accuracy values are reached by using all components simultaneously

Component comparison	Training set		Validation set	
	p-val	CI 95%	p-val	CI 95%
Full vs. Lexical	<.001	[0.4, 0.41]	<.001	[0.04, 0.06]
Full vs. Syntactic	<.001	[0.2, 0.21]	<.001	[0.02, 0.03]
Full vs. Semantic	<.001	[0.06, 0.07]	.042	[0.0, 0.02]
Full vs. Lexical-Syntactic	<.001	[0.15, 0.16]	.014	[0.0, 0.02]
Full vs. Lexical-Semantic	<.001	[0.06, 0.07]	<.001	[0.02, 0.04]
Full vs. Syntactic-Semantic	<.001	[0.01, 0.02]	<.001	[0.01, 0.03]

Let μ_s and μ_k be the average performance of using all the components in a single image and the k^{th} benchmark combination respectively. The hypothesis to be contrasted were:

$$H_0 : \ \mu_s - \mu_k = 0$$
$$H_1 : |\mu_s - \mu_k| > 0$$

We defined a significance level of 0.05. The values of μ_s and μ_k were defined in terms of the accuracy from experiments.

The results of these hypothesis tests are shown in Table 2.

These tests demonstrated that there is a significant difference on accuracy values between using all components of text and using combinations of two or one of them since in all cases p-value $< .05$. Also, it is worth to mention that in all cases the confidence interval of the difference between means is positive, which reflects that using all components of text is statistically better than using one or two of them. Additional result analyses were performed where we also included some more classification metrics and distribution analysis, these results can be seen in an additional web repository[6].

4.2. Case study 2: Complexity reading score prediction

In this scenario, we used the *CommonLit Readability Prize* corpus obtained from the competition website[7]. This corpus was used to determine if a machine learning algorithm can determine the

[6]https://github.com/ctvdata/textrespresentation

[7]www.kaggle.com/c/commonlitreadabilityprize

reading level of several text passages from books for grades 3-13 classroom use. One of the main goals in this competition was to incorporate cohesion and semantics in the trained model, which fits our point of view from our text representation proposal. This corpus is composed of 2834 text passages, where each one is associated with a complexity reading score determined by CommonLit[8] and the Georgia State University. The scores in this corpus are in the interval $[-3.7, 1.7]$, where the higher the score, the easier the passage can be read. Our interest on this problem was not to win the competition but to determine if better results are achieved by using the proposed text representation, with all components of text.

In this case, we resort to a neural network model for making regressions of complexity scores given a text passage encoded according our proposal. The used architecture was a multilayer perceptron network with the following definition:

- A flatten layer to turn the input matrix into a vector
- A fully connected layer with 600 neurons
- A ReLU activation function
- A dropout with a probability of 0.2
- An output fully connected layer with a single neuron to get a score $y \in \mathbb{R}$, which determines the ease of reading for the text passage

The training consisted on a total of 50 iterations with 100 epochs each. As in the previous case study, we used the k-fold strategy with a value of $k = 5$ to obtain an 80% of the corpus as a training set and 20% as a validation set for each series of experiments. In this case study, we used the Root Mean Squared Error (RMSE) as metric to guide the model in the learning process, where a lower value of RMSE reflects a better fit of the model to the data. As in the previous case study, a better regression model or architecture could be used, but this is also out of the scope of this work. Also, we repeated each series of experiments by using all spectra of the components of text in the documents in a single representation and combinations of one or two of them.

By assuming normality in our results, the results of these series of experiments are shown in Table 3. Here, the best outcomes are achieved once again by using all components of text (*Full*) in both training and validation sets.

Also the statistical significance was computed for these results, a t-test was conducted between the results of using the three components and combinations of one or two of them. In this case, the null and alternative hypotheses were established in similar manner as in the previous case study. Table 4 shows the obtained results of the tests.

Table 3
Mean and standard deviation of RMSE for training and validation sets in text reading score prediction. By using all components of text (*Full*) better results were achieved

Used component	Training set		Validation set	
	μ	σ	μ	σ
Full	**0.8714**	0.0094	**0.8758**	0.0287
Lexical-Syntactic	0.9182	0.0078	0.9179	0.0269
Lexical-Semantic	0.8863	0.0081	0.8849	0.0252
Syntactic-Semantic	0.9062	0.0076	0.9199	0.0252
Lexical	0.9613	0.0070	0.9552	0.0252
Syntactic	0.9732	0.0061	0.9812	0.0231
Semantic	0.9301	0.0071	0.9386	0.0247

Table 4
t-test results over the reading score prediction results. In all cases the p-value $< .05$, which reflects that there is a significant difference on RMSE between using all components of text and using combinations of one or two of them. Confidence intervals of the difference between means also reflect a negative difference in all cases, showing also that lower RMSE values are reached by using all components simultaneously

Component comparison	Training set		Validation set	
	p-val	CI 95%	p-val	CI 95%
Full vs. Lexical	<.001	[-0.09, -0.09]	<.001	[-0.08, -0.07]
Full vs. Syntactic	<.001	[-0.1, -0.1]	<.001	[-0.11, -0.1]
Full vs. Semantic	<.001	[-0.06, -0.06]	<.001	[-0.07, -0.06]
Full vs. Lexical-Syntactic	<.001	[-0.05, -0.05]	<.001	[-0.05, -0.04]
Full vs. Lexical-Semantic	<.001	[-0.02, -0.01]	<.001	[-0.01, -0.0]
Full vs. Syntactic-Semantic	<.001	[-0.04, -0.03]	<.001	[-0.05, -0.04]

Since the p-value $< .05$ in all cases, we demonstrated that there is a significant difference between using the spectra of all components simultaneously rather than using one or two of them for this case study. By taking into account that a lower value of RMSE reflects a better fit of the model, we can state that the confidence intervals between the means also reflect this enhancement by showing negative values in all cases. Additional results analysis were performed where we also included some more regression metrics and distribution analysis. These results can also be reached in the web repository[9].

[8] www.commonlit.org

[9] https://github.com/ctvdata/textrespresentation

5. Conclusions

The use of machine learning algorithms in text analysis has been beneficial in many scenarios. Although traditional text representations (binary, bag of words, etc.) have proved to produce good results, few years ago, *new* vector text representations have outperformed traditional ones. Most of the machine learning methods rely on the use of vectors representation to abstract text into a numerical form they can manipulate to find hidden patterns in text. In this sense, a text representation must abstract relevant features about the information in text to achieve good learning results. Commonly, these newer proposals focus on semantic information.

In this work we proposed a text representation integrating the spectra of all components of text (lexical, syntactic, and semantic). Our hypothesis is that with this representation is possible to abstract the content of text for learning algorithms. We assume the content of text is rich on language variety phenomena over types, sources, and writing styles, which can be captured into a single representation.

To demonstrate the goodness of our proposal, this was applied for text classification and complexity reading score prediction tasks. In the experiments different combinations of components of text were tested. Over these results statistical significance were computed, demonstrating the proposed text representation is suitable for those tasks. From these experiments series our main intention was to demonstrate that our proposal is feasible from a computational perspective. We consider as future work over this proposal exploring the goodness of having the text components in isolation by treating these content spectra with image analysis techniques (e.g. image segmentation, pixel detection, pixel correction, etc.) to ground this concept into a linguistic perspective. We consider also, integrating other types of text representations from the state of the art to our proposal by consolidating it as a multimodal representation in order to verify if the same improvement behavior showed in this work is again fulfilled (e.g. TF-IDF, probabilistic representations, or BERT-based embeddings).

References

[1] A. Torfi, R.A. Shirvani, Y. Keneshloo, N. Tavaf and E.A. Fox, Natural language processing advancements by deep learning: A survey, *arXiv preprint arXiv:2003.01200*, (2020).

[2] D.W. Otter, J.R. Medina and J.K. Kalita, A survey of the usages of deep learning for natural language processing, *IEEE Transactions on Neural Networks and Learning Systems* **32**(2) (2020), 604–624.

[3] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Harshman, Indexing by latent semantic analysis, *Journal of the American Society for Information Science* **41**(6) (1990), 391–407.

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

[5] J. Pennington, R. Socher and C.D. Manning, Glove: Global vectors for word representation, In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, (2014), 1532–1543.

[6] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E.D. Trippe, J.B. Gutierrez and K. Kochut, A brief survey of text mining: Classification, clustering and extraction techniques, *arXiv preprint arXiv:1707.02919*, (2017).

[7] A.B. Dieng, F.J.R. Ruiz and D.M. Blei, Topic modeling in embedding spaces, *Transactions of the Association for Computational Linguistics* **8** (2020), 439–453.

[8] F. Almeida and G. Xexéo, Word embeddings: A survey, *arXiv preprint arXiv:1901.09069*, (2019).

[9] M. Sahlgren, The distributional hypothesis, *Italian Journal of Disability Studies* **20** (2008), 33–53.

[10] C. Zong, R. Xia and J. Zhang, Text Data Mining, *Springer* (2021).

[11] Q. Le and T. Mikolov, Distributed representations of sentences and documents, In *International Conference on Machine Learning* 1188–1196. PMLR, (2014).

[12] M. Pagliardini, P. Gupta and M. Jaggi, Unsupervised learning of sentence embeddings using compositional n-gram features, *arXiv preprint arXiv:1703.02507*, (2017).

[13] R. Kiros, Y. Zhu, R.R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba and S. Fidler, Skip-thought vectors, In *Advances in neural information processing systems*, (2015), 3294–3302.

[14] M. Chen, Efficient vector representation for documents through corruption, *arXiv preprint arXiv:1707.02377*, (2017).

[15] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations, *arXiv preprint arXiv:1802.05365*, (2018).

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, In *Advances in neural information processing systems*, (2017), 5998–6008.

[17] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805*, (2018).

[18] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, Improving language understanding by generative pre-training. (2018).

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, *OpenAI blog* **1**(8) (2019), 9.

[20] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, *arXiv preprint arXiv:2005.14165*, (2020).

[21] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schieleand and H. Lee, Generative adversarial text to image synthesis, *Proceedings of The 33rd International Conference on Machine Learning* **48** (2016), 1060–1069.

[22] B. Li, X. Qi, T. Lukasiewicz and P.H.S. Torr, Controllable text-to-image generation, *arXiv preprint arXiv:1909.07083*, (2019).

[23] L. Li, Y. Sun, F. Hu, T. Zhou, X. Xi and J. Ren, Text to realistic image generation with attentional concatenation generative adversarial networks, *Discrete Dynamics in Nature and Society* **2020** (2020).

[24] V. Balakrishnan and E. Lloyd-Yemoh, Stemming and lemmatization: a comparison of retrieval performances. (2014).

[25] S.G. Kanakaraddi and S.S. Nandyal, Survey on parts of speech tagger techniques, In *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pages 1–6. IEEE, (2018).

[26] M. Ikonomakis, S. Kotsiantis and V. Tampakas, Text classification using machine learning techniques, *WSEAS Transactions on Computers* **4**(8) (2005), 966–974.

[27] D.J.C. MacKay and D.J.C. Mac Kay, *Information theory, inference and learning algorithms*, Cambridge University Press, (2003).

[28] T. Kohonen, The self-organizing map, *Proceedings of the IEEE* **78**(9) (1990), 1464–1480.