A Word Embeddings Model for Sentence Similarity

Victor Mijangos, Gerardo Sierra and Abel Herrera

National Autonomous University of Mexico
Language Engineering Group, Faculty of Engineering
Mexico City, Mexico
{vmijangosc,gsierram}@iingen.unam.mx, abelherrerac1@gmail.com

Abstract. Currently, word embeddings (Bengio et al, 2003; Mikolov et al, 2013) have had a major boom due to its performance in different Natural Language Processing tasks. This technique has overpassed many conventional methods in the literature. From the obtained embedding vectors, we can make a good grouping of words and surface elements. It is common to represent top-level elements such as sentences, using the idea of composition (Baroni et al, 2014) through vectors sum, vectors product or through defining a linear operator representing the composition. Here, we propose the representation of sentences through a matrix containing the word embedding vectors of such sentence. However, this involves obtaining a distance between matrices. To solve this, we use a Frobenius inner product. We show that this sentence representation overtakes traditional composition methods.

1 Introduction

Word embeddings methods, based on the proposal of neural language models [2], have over-passed traditional approaches to natural language processing in a lot of tasks [10], [3]. The distributional space model that word embeddings proposes seems to codify better the proper features of the language. In general terms, the distributional models have been used for representing words in a vector space. Nevertheless, most of the time we look for representing high-level linguistic elements. For example, here we want to find for sentences similarity through its representation on a vector space.

For representing such elements, [1] summarizes three compositional methods that can allow us to represent high-level linguistic terms:

1. Vectors sum.
2. Vectors multiplication.
3. A linear operator through representation of the compositionality of a word.

Vectors sum and vectors multiplication are easy to implement. But to determine a linear operator representing the compositional of a word becomes a hard labor. The problem with these compositional approaches is that they assume

that language is compositional. Given this, we can find many counterexamples. Therefore, we believe that a perspective that compare the similarity between the words that make a sentence, rather than the sentences themselves, can give good results in determining similarity between sentences.

2 Previous works

Similarity between sentences has been an essential problem for text mining, question answering, text summarization and another tasks. What we want is that given two sentences, we can determine their similarity through semantic proximity. Nevertheless, this task shows big complications. As we have mentioned before, most of the time the assumption of compositionality is given. So, most of the methods are based on this approach to represent linguistic items in vector spaces [1], [7].

Some of the methods that have been used to determine similarity measures for sentences are compared in (Achananuparp, 2008). One of these methodologies is the word overlap measure. This method seeks to find related concepts through comparing the overlapping words between definitions (sentences) of both concepts. In (Metzler, 2005), a method of word overlap is evaluated from a simple word overlap fraction that determines the proportion of words that appear in the two sentences to compare them. Then this proportion is normalized through the length of the sentences. Another method based on word overlap used to determine similarity between sentences is the Inverse Document Frequency (IDF) overlap, where the proportion of words is compared in two sentences from their IDF weights.

Based on these ideas, (Banerjee et al., 2003) extends the concept of word overlap to the distinction between multi-word terms. They assume that the simple word overlap methods do not take into account the elements that are composed of more than one word and that may be important for the similarity between sentences. Therefore, they estimate the overlap between multi-word terms and single word terms.

Other methods are based on Term Frequency-Inverse Document Frequency (TFIDF). For example, (Allan, 2003) is based on search thematically similar sentences. It is based on the sum of the TFIDF values of the words in both sentences, since it is assumed that this measure weights the thematically relevant words. Another way is to create vectors based on the idea of Bag of Words (BoW) and compare the distance between vectors sentences. However, BoW lose relevant information between the sentences, such as order, and have now been overtaken by the word embeddings representations.

In (Landauer, 1997) it is proposed that the similarity between sentences is given from a linear combination of vectors of semantic similarity and the similarity in words order.

Meanwhile, in (Achananuparp, 2008a) a methodology of similarity between sentences for question-answering systems is proposed. They propose an hybrid approach, combining semantic similarity and syntactic patterns; for this, they

take into account the similarity between words, word order and Part of Speech tags. Also, they integrate a Support Vector Machine to determine information about the types of sentences.

As reported by Achananuparp (2008), the method with the highest accuracy is proposed by (Achananuparp, 2008a), followed by other methods that rely in similar concepts. However, such methods require a high cost, since they are based on principles of word order, labeling Parts of Speech and semantic similarity. Meanwhile, the methods of word overlap are generally not as efficient, as they do not capture the paraphrase or syntactic level elements, rather remaining at the lexical level.

Socher (2011) proposes the analysis of similarity between paraphrases through a recursive auto-encoder (RAE). They represent the sentence as an ordered list of word vectors, $s = \{x_1, .., x_m\}$ obtained with a word embedding algorithm. Given a sentence s the binary parse tree has the form of branching triplets ($p \mapsto c_1 c_2$), where p is a partner and c_i is a children, such that c_i is a word vector or a non-terminal node. Then $p = f(W_e[c_i; c_j] + b)$, where $[c_i; c_j]$ is the concatenation of the corresponding word vectors and W_e is the encoding matrix to learn.

The error function is given by:

$$E_{rec}(p) = || \ [c_1; c_2] - [c_1'; c_2'] \ ||^2 \tag{1}$$

where $[c_1'; c_2'] = f(W_d p + b)$. This process is recursively applied until the tree is fully constructed. To assign a similarity between two sentences, a euclidean distance is computed between all words and phrases vectors. This euclidean distance fills a similarity matrix. To determine a similarity between paraphrases, the authors use a pooled matrix. Nevertheless, as the authors said, the pooled matrix loses some of the information of the original matrix.

Another approximation to sentence similarity is showed by Kartsaklis (2012). They propose a tensor-based method. They create tensors through grammatical information and Frobenius algebras. Nevertheless, this method requires the grammatical information and, also, does not use the matrix representation for computing the similarity but a $\mathbb{R}^n$ representation.

Based on this ideas, Kim (2015) proposes a tensor-based composition for the word embedding algorithm. They used the compositional methods explained in (Milajevs, 2014) for integrating the compositional function into the neural network of the word embeddings. They proved the typical mean and point-wise multiplication techniques; additionally, they proved the sum and the concatenation of these two techniques. Furthermore, they proposed tensor-based techniques; nevertheless, they used a projection function to obtain a vector in $\mathbb{R}^n$ from the matrix representation.

A comparison of different methods is made by Milajevs (2014). They show the simple vector addition and multiplication methods as well as the Frobenius algebras based methods. However, for the computation of similarity between sentences they use a vector representation in all cases combined with a cosine distance.

Our proposal seeks to be more simple, and at the same time to capture elements that are beyond the lexical level. At the same time, the method proposed

here does not lose information of the similarity between words. This proposal seeks to characterize a sentence from the words that compose it, and to determine a matrix space where a geometry can be defined through a Frobenius inner product. Thus, the method proposed here becomes simple compared with the categorical compositional distributional and projection methods. Furthermore, it has good performance.

3 Theoretical framework

The methodology proposed here consists of two main tools: word embeddings and Frobenius inner product. In the following section we describe these tools.

3.1 Word embeddings

In the area of Computational Linguistics, it is common to hear about language models. These language models can be seen as Markov process of order r. Overall, Markov models allow us to see the likelihood of a chain as the product of the states that make up the chain. The language models are simple models that attempt to determine the likelihood of a linguistic element from the n above items. Thus we can understand them as we show below.

Definition 1 (n-grams models) *Giving a string of words* $w_{1,n} = \{w_1, ..., w_t\}$ *the language model determines the probability of the chain from the probability of the states of the chain. Such that:*

$$P(w_{1,t}) = \prod_{i=1}^{t} P(w_i | w_{i-n+1} ... w_{i-1}) \tag{2}$$

where $n \in \mathbb{N}$ *is the lenght of the window.*

Perhaps the simplest model within these is the bi-gram model. It takes $r = 2$ such that we have a simple Markov process, then equation (2) becomes:

$$P(w_{1,t}) \approx \prod_{i=1}^{t} P(w_i | w_{i-1}) \tag{3}$$

Based on these models, [2] proposes a model to determine the probability of a linguistic element given its context. To do this, it is proposed the use of deep learning. Therefore, these new models are called 'neural language model'. As [2] noted, this idea is based on three main points:

1. Representing each word in a vocabulary through a distributed feature vector (a vector with entries in $\mathbb{R}$).
2. Expressing a joint probability function of words sequences in terms of the features vectors of the words in the sequence.

3. To learn simultaneously the feature vectors and the parameters of the probability function.

The probability function, in this case, is of the type expressed in equation (2). Now, however, a neural network is used to predict subsequent words in the string. In this case, the vectors representing the word are learnt by a learning machine. In general, the model is based on the ideas of distributional models; the vectors of similar words should be similar, and the context of words plays an important role.

We have a vocabulary $\mathfrak{C} = \{w_1, ..., w_t\}$ of finite lenght. We want to learn a model $f(w_{i-n+1}...w_i) = P(w_i|w_{i-n+1}...w_i)$ such that $\sum_{j=1}^{t} f_j(w_{i-n+1}...w_{i-1}) = 1$; this means that we look for f to be a probability measure. We can define f as:

$$f_j(w_{i-n+1}...w_{i-1}) = g_j(v(w_{i-n+1})...v(w_{i-1})) \tag{4}$$

where

- $v : \mathfrak{C} \to \mathbb{R}^m$ is the function determining a vector in $\mathbb{R}^m$ for each word in the vocabulary. In general terms, $v(\cdot)$ is represented as a matrix of size $|\mathfrak{C}| \times m$.
- A function g (a neural network) that maps a sequence of input vectors, taking into account its context, to a conditional probability distribution for the next word w_i. The output vector generated by g is a probability vector such that the ith entry estimates the probability of $P(w_i|w_{i-n+1}...w_{i-1})$.

To determine the probability of each word given the $n+1$ previous words, the Softmax regression is used. The Softmax regression is a form of asymmetric probability of neighbors (Hinton, 2002) where $\delta = \langle \cdot, \cdot \rangle$ is the inner product, such that:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{exp(\langle v_i, h(v_{i-n+1}...v_{i-1}) \rangle)}{\sum_{k=1}^{t} exp(\langle v_k, h(v_{i-n+1}...v_{i-1}) \rangle)} \tag{5}$$

where $v_k = v(w_k)\ \forall k \in \{1, ..., t\}$ and $h : \mathbb{R}^{m \times (n+1)} \to \mathbb{R}^m$ is a function that maps the word vectors of the context to a sole vector in $\mathbb{R}^m$.

Therefore, the model proposed in (2) now depends on the equation (5). However, what we want is a vector space representation of sentences.

In general terms, the method of word embeddings is based on stochastic neighbor embedding, watching the elements in the context of a word as neighbors. Therefore it seeks to maximize the equation (5) while a loss function, that often tends to be the Kullback-Liebler divergence (Hinton, 2002), is minimized. So we want to minimize:

$$KL(p||q) = \sum_{i=1}^{t} p_i \log \frac{p_i}{q_i} \tag{6}$$

However, since the Kullback-Leibler divergences represents a high computational cost, the use of negative sampling has been proposed [6], which streamlines the process of word embeddings.

Then we have a word w and its context, $N(w)$. To generate the vectors of dimension m, where each coordinate represents a feature that is learnt by a neural network, we follow the next steps:

1. We randomly generated a matrix $V \in \mathbb{R}^{t \times m}$ of input vectors and a different matrix $V' \in \mathbb{R}^{m \times t}$, where t is the number of words.
2. Given the context, we determine $P(w_i, N(w_i))$ with softmax regression. We choose the function h as follows:

$$h := \frac{1}{n} \sum_{k=1}^{n} v'_k \tag{7}$$

where $v' = v'(w)$ is the word vector of the matrix $(V')^t$.
3. We upgrade from a hidden layer to the output of the neural network, by using a stochastic gradient descent with a learning rate η, such that:

$$v'_{i+1}(w_j) = v'_i(w_j) - \eta \nabla \epsilon(w_j) \tag{8}$$

where ϵ is the loss function. We use a negative sampling such that we define the loss function as follows.

$$\nabla \epsilon(w_j) = \begin{cases} 1 - P(w_j|N(w_j))h(N(w_j)) & \text{if } w_j = w_o \\ 0 - P(w_j|N(w_j))h(N(w_j)) & \text{if } w_j \neq w_o \end{cases} \tag{9}$$

where w_o is the actual objective word in the iteration.
4. To upgrade the input to the hidden layers, given $e_k \in N(w_j)$ we have $\forall m' \in 1, ..., m$

$$v_{i+1}(w_k) = v_i(w_k) - \eta \sum_{j=1}^{n} \epsilon(w_j) \cdot v'_{m',j} \tag{10}$$

where $v'_{m',j}$ represents the m'th entry of the matrix V'.

The algorithm then iteratively runs until the cost function is less than a given range. Also, it can be run a determined number of iterations.

This way, what the algorithm does is try to approximate the distributions between the matrices V and $(V')^t$ from the observations of the contexts in which a word occurs.

Currently, the methods based on word embeddings have surpassed previous models in most of the tasks of natural language processing. Also they have the advantage of not requiring a dimensionality reduction because the dimension is chosen a priori. Therefore we work with vectors having low dimensionality unlike other methods. It is important to point out that the distributional assumptions [5], [9], are still presented, implying that word embeddings are capturing the similarity of the words from the idea that similar words appears in similar context (Goldberg, 2014).

The method of word embeddings generates word vectors. From these word vectors we can perform a composition process, which commonly consists of the

sum vector (it has also been used despite other methods described above). However, what we propose is to characterize a sentence from a matrix where each row vector is a vector representation of the words that compose such a sentence. However, before this, we are faced with reinventing the classic methods of distance between vectors. For this, we use the Frobenius inner product and related concepts explained below.

3.2 Frobenius distance

To calculate the distance between two matrices, we first have to define a geometry in the space of matrices of $m \times n$. For this, we require an inner product. That is, a function that, given u, v, and w vectors and a scalar $\lambda \in \mathbb{R}$, the next properties are met:

Positive-definite $\langle v, v \rangle \geq 0$.
Non-degenerative $\langle v, v \rangle = 0$ iff $v = 0$.
Bilineal $\langle \lambda u + w, v \rangle = \lambda \langle u, v \rangle + \langle w, v \rangle$.
Conjugated symmetry $\langle u, v \rangle = \overline{\langle v, u \rangle}$.

Given this, the following theorem gives us an inner product into the space of matrices of $m \times n$ [4], [8].

Theorem 1 *The equation*

$$\langle X, Y \rangle = trace(X^t \cdot Y) = \sum_{i=1}^{n} \langle X_i, Y^i \rangle \tag{11}$$

defines a inner product in the spaces of $\mathbb{R}^{m \times n}$.

It is easy to see that given the properties of the trace and the inner product over the column vectors of a matrix, the equation (11) is positive-definite. We can also see that it is bilinear, as we have:

$$\begin{aligned} \langle \lambda X + Z, Y \rangle &= trace((\lambda X + Z)^t \cdot Y) \\ &= \sum_{i=1}^{n} \langle \lambda X_i + Z_i, Y^i \rangle \\ &= \lambda \langle X_i, Y^i \rangle + \langle Z_i, Y^i \rangle \\ &= \lambda \langle X, Y \rangle + \langle Z, Y \rangle \end{aligned}$$

Finally, it satisfies the property of conjugated symmetry as we have:

$$\begin{aligned} \langle X, Y \rangle &= \sum_{i=1}^{n} \langle X_i, Y^i \rangle \\ &= \sum_{i=1}^{n} \langle (Y^i)^t, (X_i)^t \rangle \\ &= \langle Y, X \rangle \end{aligned}$$

Since equation 11 indeed defines an inner product, then we can move on to define a norm on the space of matrices of $m \times n$. This norm is known as the Frobenius norm and is defined as (see (Tarazaga, 2001)):

$$||A|| = \sqrt{\langle A, A \rangle} = \sqrt{trace(A \cdot A^t)} \tag{12}$$

The norm defines a metric given by $||A - B||$ where A and B are matrices. Nonetheless, we seek a way to standardize this distance. Given the properties of the norm and the inner product, it is clear that the Cauchy-Schwartz inequality is satisfied and therefore we can define a distance function as follows:

$$cos(A, B) = \frac{\langle A, B \rangle}{||A|| \cdot ||B||} \tag{13}$$

$$= \frac{trace(A, B^t)}{\sqrt{trace(A \cdot A^t) \cdot trace(B \cdot B^t)}} \tag{14}$$

So the distance between two matrices holds that $|cos(A, B)| \leq 1$. So we now have a normalized distance between matrices that can be applied to the matrices generated from the vectors of the words composing each sentence.

4 Proposal

Based on what has been showed, we propose to calculate the similarity of two sentences from their matrices created by embedding vectors of words and the use of a Frobenius based distance to determine the similarity between the two matrices in the space. We start with the following elements:

- $Q = \{s_1, ..., s_m\}$ a set of sentences.
- $W = \{w_1, ..., w_k\}$ a set of words obtained from the sentences in Q.

Now, we choose a sentence set, this set will be the training set, then the words are split in order to obtain the vector representation used to generate the matrix. However, a large sample number is required to obtain word embeddings. In consequence, we used the Corpus del Español Mexicano Contemporáneo (CEMC) (DE, 1987s) that has around two millions of words.

So, the set of words W is done from CEMC corpus. It allows that a sentence not in Q can be compared even if its words are not in the training set of sentences.

The proposal consists of five steps: words representation in a vector space, sentences representation, petition representation and the compute of similarity between sentences. These steps will be explained below:

Word representation. A vector space of dimension n is generated (the dimension is 800) from the CEMC using word embeddings. We use word2vec [7] [6]. The window equals to 5 and the minimum occurrence is 1. With these parameters all the words are included and the algorithm has more range when we ask for a query. In this step, the vector space $W \subset \mathbb{R}^m$ is obtained, each vector is the representation of one word of CEMC.

Sentence Representation. The sentence representation in Q is done from matrices where each line is one vector of a word in W. To do this, first of all, the sentence is divided in its words, where a vector $v \in W$ is the representation of the word. So, the method obtains a matrix of $r \times n$, where n is the dimension of W (the chosen dimension is equals to 800) and r is the number of the sentence words. Finally, the method obtains a set $S = \{s_1, ..., s_m\}$ where each $s_i, i = 1, ..., m$ is the matrix representation of the sentence i.

Petition Representation. The petition is a sentence; the method will look for the Q element most similar. The petition is called q_0, not necessarily in Q. It is replaced by the vectors representation of its words, so it forms a matrix s, of $l \times n$ dimension.

Sentence similarity. Finally, the objective is to determine a distance between s', the matrix representation of q_0, and all the elements of S. This function will give us the most similar sentence to s'. The reader can notice that the matrices are not of the same dimension but have the same number of columns. In the method we propose the use of the Frobenius inner product to compute this distance. So, the similarity function between the matrices is given by equation 13 and equation 11. Then the sentence most similar to q_0 is given by

$$\arg\max_i \delta(s', s_i), s_i \in S \tag{15}$$

where δ is the distance between the matrices.

The process is described at Figure 1.

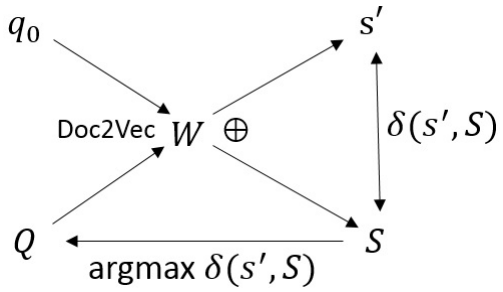

Fig. 1. Process to compute the similarity between sentences.

The advantages of our proposal over other methods reside in the computation of the similarity through the matrix representation. We do not need to make any map into $\mathbb{R}^n$ for computing a metric between sentences. The Frobenius inner product allows us to determine a primal geometry over the space of matrices. With this geometry we can calculate the similarity between points in this vector space (these points are matrices). But what happens in a deep level is that the Frobenius inner product computes a standard inner product over the row vectors

representing the words of the sentences. Then, the Frobenius inner product can bee seen as follows:

$$\langle A, B \rangle = \sum_{i=1}^{r} \langle a_i, b_i \rangle \tag{16}$$

where a_i is the ith word vector composing the sentence represented by the matrix A and b_i the ith word vector of the sentence represented by the matrix B.

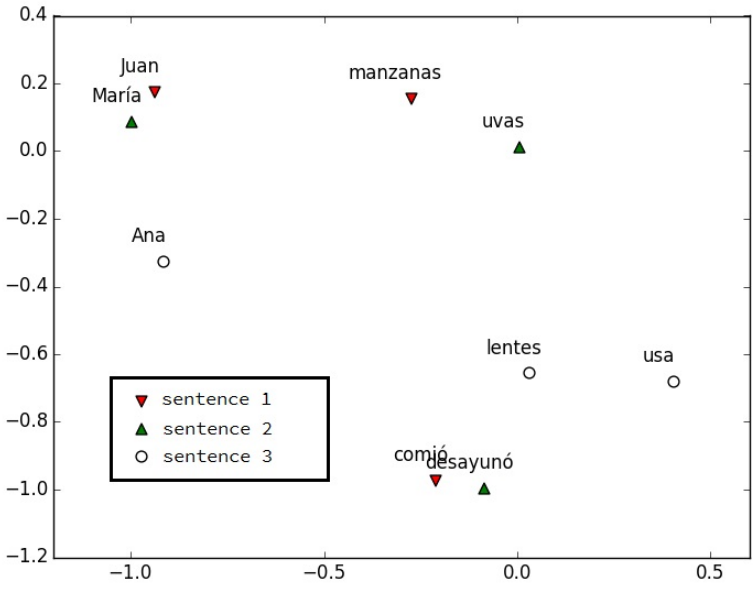

Fig. 2. Plot of the words composing the similar sentences 'Juan comió manzanas' ('John ate apples') and 'María desayuna uvas' ('Mary had grapes for breakfast'), and the disimilar sentence 'Ana usa lentes' ('Ana wears glasses').

Take a sentence like 'Juan comió manzanas' ('John ate apples'). This sentence is similar, for one hand, to 'María desayunó uvas' ('Mary had grapes for breakfast') since the verbs 'comió' y 'desayunó' are similar; both verbs reflect the action of 'eat', past tense and third singular person. Also, the direct objects 'uvas' and 'manzanas' ares similar because they both are fruits. For the other hand, a sentence like 'Ana usa lentes' ('Ana wears glasses') has a completely different meaning. Here, the verbs 'wears' and 'ate' are from different semantic groups, such as 'glasses' and 'apples'. If we plot the three sentences we can

see that the words composing them are grouped differently depending on its meaning (see Figure 2). Computing the Frobenius cosine between the two similar sentences 'Juan comió manzanas' and 'María desayunó uvas' we have 0.62. Whereas for 'Juan comió manzanas' and 'Ana usa lentes' the Frobenius cosine equals 0.39.

Our method tries to make the calculation of similarity between sentence simpler than than other methods like the proposed in (Milajevs, 2014) or (Socher, 2011) by computing the metric in the original matrix space. This implies that there is no information loss and we do not need to map the representation into another simpler vector space (generally $\mathbb{R}^n$).

5 Results

For evaluation, a corpus of Spanish paraphrases was used; this corpus has 144 sentences with paraphrases. The corpus always has an original document and two paraphrase levels: low-level paraphrases and upper-level paraphrases. The low-level paraphrases has lexical and syntactic changes and, in general, formal level changes. The upper-level paraphrases has changes at discursive level, semantic and more elaborated changes. Then, for the evaluation process, the method only uses low level paraphrases, and they are compared to the original texts. More information of this corpus is at (Mota).

The baseline is the paraphrase representation from word vectors sum, the vectors were obtained from the words embedding method explained above. The cosine and inner product are compared at both cases, then the accuracy is obtained. Using the cosine distance, the highest accuracy is achieved by our method. Unfortunately, only by a difference of 0.03; this improvement is not significant. However, with the inner product, our method achieve a 0.62 accuracy, against the accuracy of 0.55 obtained by the vectors sum method (see table 1).

Table 1. Results comparation.

Method	**Inner product**	**Cosine**
Matrix representation	**0.6206**	0.5862
$\mathbb{R}^n$ representation	0.0689	**0.5517**

6 Conclusions and Future work

The proposed method has a better performance compared to the traditional $\mathbb{R}^n$ vector representation. The $\mathbb{R}^n$ representation method is commonly used to represent linguistic elements; now, the information from that method can be amplified using a matrix representation. This way, the a priori conception of compositionality is not assumed. In the matrix representation method, each word represents

a column of a matrix and we can compare each word with the words of another sentence. So, a matrix representation is more adequate than the representation through vectors in $\mathbb{R}^n$.

The advantage over other methods, such as the one proposed in (Socher, 2011) is that our proposal does not lose information. The Frobenius inner product is a natural geometric form in the matrix space. This inner product compares the distance between all the words of both sentences to give us a similarity measure of the sentences.

However, a lot of work must be done. The matrix representation method must be compared with other methods. In this paper we prove this method for Spanish language. Nevertheless, a more complete set of languages can show that this is a good language-independent method.

We want to point out that, as the angle within matrices makes no much sense, the Frobenius inner product improve the result of the experiment. Nevertheless, we use just two distances to compare. For later works, it is necessary to define metrics that capture better the topology of the data. Also, the word embeddings methods need big size corpora. So a bigger corpus than CEMC is needed for a question-answering system application.

The sentence representation can be improved. We see that, by the definition of the Frobenius inner product, the word order between the sentences improves the results. This is given because it is expected that both sentence were composed by the same syntactic structure. Nevertheless, this happens rarely. Other elements as syntactic information and morpho-syntactic information can be integrated. The contribution of these elements must be proved.

References

1. Baroni, M., Bernardi, R., Zamparelli, R.: Frege in space: A program of compositional distributional semantics. Linguistic Issues in Language Technology 9 (2014)
2. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. The Journal of Machine Learning Research 3, 1137–1155 (2003)
3. Campr, M., Ježek, K.: Comparing semantic models for evaluating automatic document summarization. In: Text, Speech, and Dialogue. pp. 252–260. Springer (2015)
4. Chang, K.C., Pearson, K., Zhang, T., et al.: Perron-frobenius theorem for nonnegative tensors. Commun. Math. Sci 6(2), 507–520 (2008)
5. Harris, Z.: Distributional structure. Papers on Syntax pp. 3–22 (1954)
6. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
8. Noutsos, D.: On perron–frobenius property of matrices having some negative entries. Linear Algebra and its Applications 412(2), 132–153 (2006)
9. Sahlgren, M.: The distributional hypothesis. Italian Journal of Linguistics 20, 33–54 (2008)
10. Sateli, B., Witte, R.: Supporting wiki users with natural language processing. Proceedings of the 8th Annual International Symposium on Wikis and Open Collaboration pp. 379–423 (2012)