EFFECTIVE VECTOR REPRESENTATIONS FOR VARIABLE LENGTH SYMBOL SEQUENCES

Gustavo Lado and Enrique Carlos Segura

Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires

ABSTRACT

Machine learning techniques have demonstrated their versatility and have been successfully applied to a wide variety of problems. However, one of their major limitations is the treatment of sequential information. In general the input and output for these methods is expressed as fixed-dimension vectors, but in many problem domains, as in natural language processing, the information is represented by variable-length sequences. In most cases, it is possible to use some methods that transform these variable length sequences into fixed dimension vectors, but each of these methods has its own disadvantages. In this paper we propose an alternative to obtain vector representations of fixed dimension from sequences of symbols of variable length and their potential applications for natural language processing..

KEYWORDS

Neural Networks, Natural Language Processing, Sequential Learning, Deep Architectures

1. INTRODUCTION

One of the main topics of interest in the area of machine learning is natural language processing, but despite the excellent results that have been obtained there is still a barrier that is difficult to overcome [1]. Most machine learning techniques are designed to work with instantaneous information represented in the form of vectors; natural language, however, is always presented as sequential information. Whether we consider words as sequences of letters, sentences as sequences of words, or documents as sequences of sentences, in all these cases we may think that information is presented as a sequence of symbols, and in order to effectively use these machine learning techniques we need, in some way, to transform this sequential information into a vector representation. But what kind of such representation we want to obtain? [2]

For example, it is desirable that the vector representation be directly related to the symbols forming the sequence, not only indicating which symbols are present, or their quantity, but also in their order [3]. Ideally the vector representation will have enough information about the sequence so as to make it possible its reconstruction.

It is also desirable that the vector representation obtained have the smallest possible dimension [4]. Some degree of redundancy may be acceptable for error correction, but given the nature and possible applications of this method every extra dimension in the representation can carry a computational load in later stages.

An important point for the vector representation is to be consistent across all valid sequences. For example, similar sequences should have similar vector representations, so that the vector distance between two different representations could be used to measure the similarity of the sequences to which they correspond.

Ideally the representations could be so consistent as to end up supporting vector operations with constraints but responding to a certain degree of compositionality [5]. For example, it might be useful if arithmetic operations could be performed on the vector representations to obtain a representation close to the result of doing the same operations on the original sequences.

And finally it is desirable that the method be effective, i.e., that it be possible to encode any valid sequence easily. This means that it is possible to accept restrictions on what is considered a valid sequence, for example, in the set of possible symbols or in the maximum length, but once these restrictions are accepted the codification should be possible without consuming many computational resources. Furthermore the method has to be efficient [6]. This means that the training should be performed relatively quickly even for large datasets, and ideally must have an acceptable degree of generalization for any valid sequence.

2. BACKGROUND AND MOTIVATION

Currently, there are several methods that offer similar solutions. We will consider next some of the most commonly used to evaluate how they behave in relation to all the properties described above.

One of the most used is known as Bag of Words [7]. With this method, each sentence or document is represented by a vector with as many dimensions as words contain a dictionary. Each position in this vector corresponds to the number of times that a specific word of the dictionary appears in the sequence. This type of vectors have the advantage of being easily generated with great consistency between sequences. But they present the problem of discarding a large amount of information and the result is usually of such a large dimension that it is rarely used directly without going through a later stage of dimensionality reduction.

Another of the first models to consider vector representations for sequence processing were the Simple Recurrent Neural Networks [8]. In this case, the previous state of the hidden layer of a neural network is used as part of the input during training to try to predict the next item in the sequence. This method can effectively generate vector representations in the hidden layer, even with a low dimension, and the result is completely consistent for any valid sequence. However, the same recurring nature of the architecture generates problems with the back-propagation training, making the method only able to be effectively used for small datasets.

An alternative to try to deal with this problem are the so-called Long-Short Term Memory networks [9]. They use a type of unit with several internal connections to be able to decide which information is propagated at any time. This allows them to use multiple hidden layers to work with much larger data sets. But the same use of this type of units not only makes the training more complex, but also makes it impossible to obtain a single vector representation with all the corresponding information for each sequence.

There is also another not-so-known model called Recursive Auto-Associative Memory (RAAM) [10] which is, in a way, a generalization of Elman's simple recurrent network model [8]. A RAAM network is composed of an auto-encoder capable of compressing a pair of patterns to only one of smaller dimension. This new compressed patterns can be recursively fed back into the network, allowing it to learn to encode and decode complex data structures such as lists and trees.

This would make it ideal for working on language problems where information is defined by grammars with an inherently recursive structure.

However, this same mechanism also has some disadvantages [11]. When a single network is used to contain all the information for encoding and decoding a set of trees the model capacity is severely limited, and becomes less robust and more difficult to train.

3. MODEL AND METHODOLOGY

Trying to avoid the disadvantages described and maintaining the desirable properties already mentioned, it is proposed the use of a model inspired by the RAAM networks but consisting of a series of auto-encoders organized in successive stages, with each stage responsible of learning only the patterns corresponding to their level.

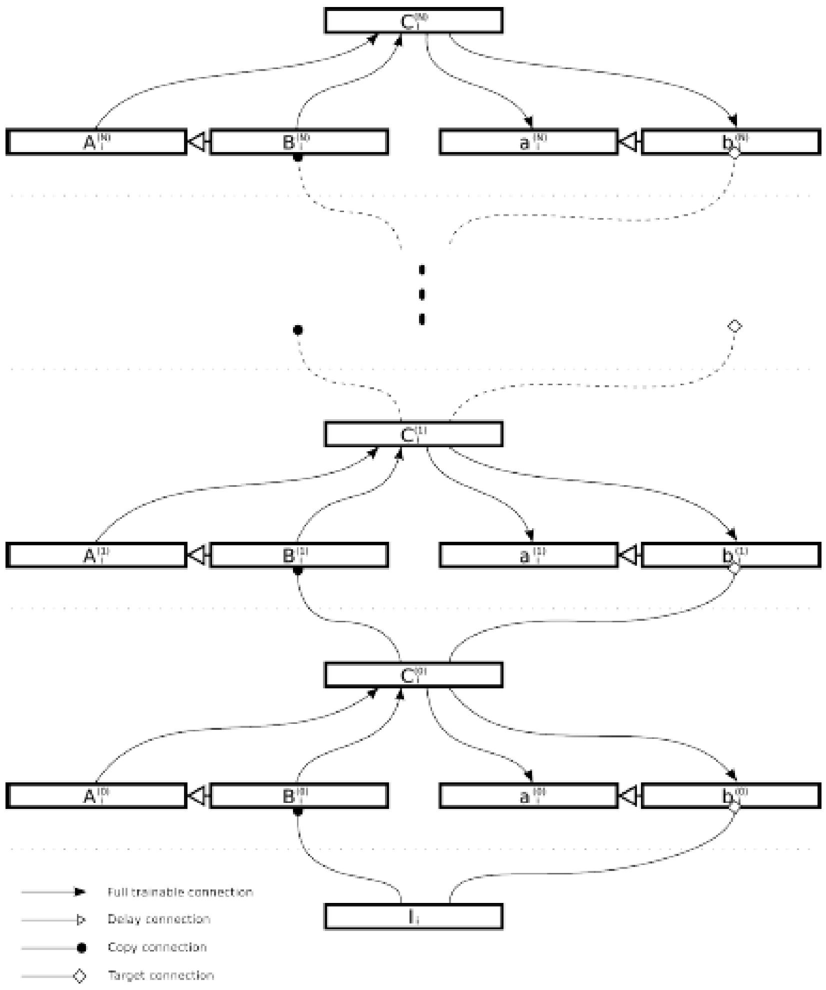

Figure 1. Diagram of the model.

The patterns encoded at one level, instead of being fed back to the same stage, are passed through a copy connection to an upper layer. In the same way, the decoded patterns in one stage are not fed to the same network but are passed to the lower stage. To construct the pairs of patterns corresponding to each stage a type of delayed connection is used which is responsible for the coordination.

In this way, each stage should only learn the regularities corresponding to its level, that is, the first level learns about regularities between pairs of letters, the next about regularities between pairs of pairs of letters and so on [12]. This increases the capacity of the model since the network at each stage should only learn a part of the set of patterns that previously had to learn a single network, making it also less prone to failures.

Figure 1 shows a diagram of the model. The stages are separated by dotted lines. Each stage has an auto-encoder similar to the one on a RAAM network, where the input is formed by layers *A* and *B*, the hidden part by layer *C*, and the output by the layers *a* and *b*. In all these layers the position within the sequence is indicated by the subscript *i* and the corresponding stage with a superscript *(n)*.

The input sequence will be defined by $I = [I_0, I_1, \dots I_L]$ wherein each I_i is a vector representing a symbol. At the bottom, layer I_i indicates the pattern at position *i* in the input sequence.
At each stage *j* the encoding and decoding corresponding to the position *i* within the sequence are described by:

$$[A_i^{(j)}, B_i^{(j)}] \rightarrow [C_i^{(j)}] \rightarrow [a_i^{(j)}, b_i^{(j)}]$$

In turn, each stage *j* will produce a sequence $C^{(j)} = [C_0^{(j)}, C_1^{(j)}, \dots C_M^{(j)}]$ in which each $C_i^{(j)}$ is also a vector [13]. One of the objectives of this architecture is that each stage encodes a sequence of less or equal length, that is to say, to produce sequences that comply with $|C^{(j)}| >= |C^{(j+1)}|$ until finally $|C^{(N)}|=1$, where N is the number of stages.

Trainable, copy, delayed and target connections are indicated with different types of lines in the same figure. The copy and delay connections work together to define the pairs of patterns that each stage must learn. For the first stage the input patterns to the network are defined by:

$$[A_i^{(0)}, B_i^{(0)}] = [I_i, I_{i+1}]$$

For the successive stages it is possible to choose between two ways of selecting the pattern pairs depending on the level of redundancy that is desired. For example it is possible to switch between two different methods defined by:

$$[A_i^{(j)}, B_i^{(j)}] = [C_{2i}^{(j-1)}, C_{2i+1}^{(j-1)}] \text{ if } j \text{ is even}$$

$$[A_i^{(j)}, B_i^{(j)}] = [C_i^{(j-1)}, C_{i+1}^{(j-1)}] \text{ if } j \text{ is odd}$$

Note that this makes $|C^{(j)}| = 2|C^{(j+1)}|$ if *j* is even and $|C^{(j)}| = |C^{(j+1)}|+1$ if *j* is odd, fulfilling the required $|C^{(j)}| >= |C^{(j+1)}|$ condition. Also note that in cases where *j* is odd is met $A_{i+1}^{(j)} = B_i^{(j)}$, this introduces the level of redundancy in the patterns at the time of decoding that makes the model more robust [14]. In each case the targets for $[a_i^{(j)}, b_i^{(j)}]$ are equal to the values of $[A_i^{(j)}, B_i^{(j)}]$ previously determined.

4. RESULTS

In order to demonstrate its properties a model consisting of 7 stages capable of coding letter sequences was used, where the letters are represented by bipolar patterns of dimension 8. The number of units used in each stage was chosen depending on the amount of combinations of pairs of patterns produced by the previous stage. Finally the vector produced by the final stage has a size of 128 bipolar values. These vectors are represented in the figures with black and white squares corresponding to the positive and negative values.

The model was trained with a set of 1000 random words from the English language. In order to demonstrate some of the desired properties, it was necessary to ensure that 9 of the words were related to “every” and “sing” as shown in the examples below. It is important to note that vector representations do not have, neither are intended to have, any information about the meaning of the words. It is a purely syntactic, non-semantic representation.

The first results consist of some pairs of words whose vectors have the smallest mutual distances of the whole set. In this case the values of the distances between the pairs of vectors are not being included because the vector representations are sufficiently similar between them (Figure 2).

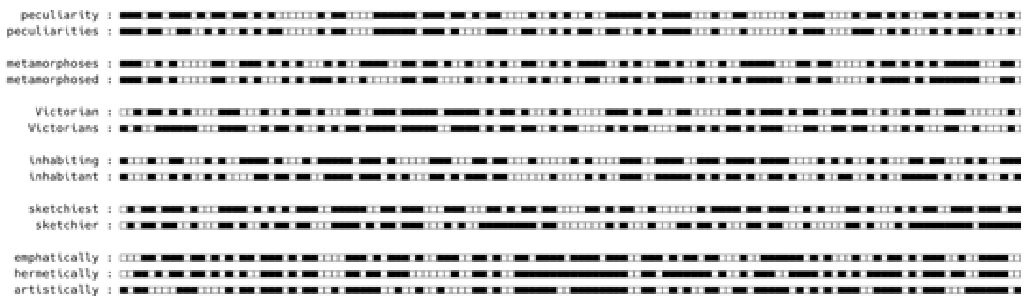

Figure 2. Words from the dataset with the similar vector representations.

In particular, it is interesting to show how similar words obtain similar vector representations, and how these representations can be used to identify words belonging to a group or family [5,13]. Also, comparing three words with different root but same syntactic category, it can be seen how it would be possible to use this type of representation to identify to which category a new word belongs.

Another interesting aspect to note is the ability to detect typographical errors [15]. As part of the training set, the words "every", "ever" and "fever" were included as all valid and similar to each other, and compared to three types of errors (replacement, omission and addition of one letter) with respect to "every" (Figure 3).

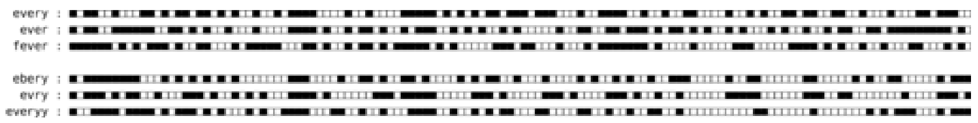

Figure 3. Vector representations from words similar to “every”.

In this case we compare the distances between the representations of some valid words similar to "every" and other vector representations obtained from variations with typographical errors.

dist(every,ever) = 13.227	dist(every,ebery) = 12.867
dist(ever,fever) = 13.239	dist(every,evry) = 12.982
dist(every,fever) = 13.631	dist(every,everyy) = 13.695

A potential application of the information obtained with this method is the possibility to identify an unknown word either as a poorly written word or as a new word. If it is a known word but poorly written, its vector representation should be sufficiently similar to the original word (ie, low vector distance between them).

song :
sings :
sing :
SINGING :
ringing :
sinking :
swinging :

Figure 4. Vector representations from words similar to “singing”.

In the next test (Figure 4) it is shown how it would be possible to infer information of an unknown word from known words [16]. In this case the model was trained with some words as "sing", "sings", "song" and some gerunds as "sinking", "ringing" and "swinging", and it was analyzed what type of representation would be obtained for an unknown word, but associated to these two groups, like "singing".

dist(sing,sings) = 14.344 dist(sinking,singing) = 12.912

dist(sing,song) = 14.152 dist(ringing,singing) = 13.536

dist(sings,song) = 15.059 dist(swinging,singing) = 15.301

In this case it can be seen that the distance between two similar but known words is usually greater than the distance with a new word which is associated with known words. However, the real measure is to see how adequate the representation of the new word is in relation to the known groups of words [5,11].

dist(sinking-sink+sing, singing) = 10.781

dist(ringing-ring+sing, singing) = 10.269

dist(swinging-swing+sing, singing) = 11.723

These representations obtained by performing arithmetic operations on the vectors were not only very similar to the representation of the new word, but also very similar to each other, showing that the representations generated by this method are consistent for different sequences.

5. CONCLUSIONS AND FUTURE WORK

In this paper we explored the possibilities of the proposed method applied to sequences of letters, that is to say, words; but it is clear that this same principle can be applied to sequences of words, and even sequences of sentences. The case of word coding is simple enough to easily show the properties of the model and can also be used as a first step in more complex coding. However working exclusively with a natural language is not enough to show its true flexibility.

A good demonstration of the capability of the model would be to test it against randomly generated sequences from regular and context-free grammars and measuring its properties against different levels of entropy.

Another aspect to consider is the memory capacity of the model, specifically what is the minimum number of units needed at each stage for it to be able to reconstruct all known sequences without errors, and the number of stages needed with the proper methods of pattern pairing in order to have an adequate level of redundancy.

REFERENCES

[1] Dietterich, T.G., 2002, August. Machine learning for sequential data: A review. In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR) (pp. 15-30). Springer Berlin Heidelberg.

[2] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Learning representations by back-propagating errors. In Neurocomputing: foundations of research, James A. Anderson and Edward Rosenfeld (Eds.). MIT Press, Cambridge, MA, USA 696-699.

[3] Socher, R., Lin, C.C., Manning, C. and Ng, A.Y., 2011. Parsing natural scenes and natural language with recursive neural networks. In Proceedings of the 28th international conference on machine learning (ICML-11) (pp. 129-136).

[4] Sutskever, I. and Hinton, G.E., 2007, March. Learning Multilevel Distributed Representations for High-Dimensional Sequences. In AISTATS (Vol. 2, pp. 548-555).

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[6] LeCun, Y.A., Bottou, L., Orr, G.B. and Müller, K.R., 2012. Efficient backprop. In Neural networks: Tricks of the trade (pp. 9-48). Springer Berlin Heidelberg.

[7] Wallach, H.M., 2006, June. Topic modeling: beyond bag-of-words. In Proceedings of the 23rd international conference on Machine learning (pp. 977-984). ACM.

[8] Elman, J. L. 1990. Finding structure in time. Cognitive science, 14(2), 179-211.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 1997), 1735-1780.

[10] Pollack, J.B., 1990. Recursive distributed representations. Artificial Intelligence, 46(1), pp.77-105.

[11] Levy, S., Melnik, O. and Pollack, J., 2000, February. Infinite RAAM: a principled connectionist basis for grammatical competence. In Proceedings of the 22nd annual meeting of the cognitive science society (pp. 298-303).

[12] Hinton, G.E., 2007. Learning multiple layers of representation. Trends in cognitive sciences, 11(10), pp.428-434.

[13] Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. science, 313(5786), pp.504-507.

[14] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol, P.A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, 11(Dec), pp.3371-3408.

[15] Li, Y., Cohn, T. and Baldwin, T., 2016. Learning robust representations of text. arXiv preprint arXiv:1609.06082.

[16] Ravuri, S. and Stolcke, A., 2016, March. A comparative study of recurrent neural network models for lexical domain classification. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on (pp. 6075-6079). IEEE.

AUTHORS

Gustavo Lado is a graduated student from the University of Buenos Aires in Computing Science.

His previous research interests include the simulation of the human visual system with artificial neural networks. Part of this work was recently published in the book "La Percepción del Hombre y sus Máquinas".

He is currently working in neural networks applied to the natural language processing and cognitive semantics as part his Ph.D.

Enrique Carlos Segura was born in Buenos Aires, Argentina. He received the M.Sc. degrees in Mathematics and Computer Science in 1988 and the Ph.D. degree in Mathematics in 1999, all from University of Buenos Aires.

He was Fellow at the CONICET (National Research Council, Argentina) and at the CONEA (Atomic Energy Agency, Argentina). He had also a Thalmann Fellowship to work at the Universitat Politécnica de Catalunya (Spain) and a Research Fellowship at South Bank University (London). From 2003 to 2005 he was Director of the Department of Computer Science of the University of Buenos Aires, where he is at present a Professor and Resarcher.

His main areas of research are Artificial neural Networks -theory and applications- and, in general, Cognitive Models of Learning and Memory.