An application of Spectral Clustering on Named Entity Recognition

Ali Josue Limon; Michele Cerú
ajl649@nyu.edu ; mc3784@nyu.edu

December 6, 2016

Abstract

This study explores the peformance of spectral clustering in a Spanish NER task. Our approach comprises two main steps. Firstly, we represent each Entity in two different ways: 1) by the embedding vector formed from the entity and 2) by the embedding vector formed from the context and the entity. Then, we use spectral clustering over the words embeddings to analyze weather the words cluster according to their labels or not. In order to train the word embedding, we used Word2Vec algorithm over a huge Spanish Corpus with the skip-gram and CBOW models. Our results suggest that

1 Introduction

Named Entity Recognition (NER) is one of the important parts of NLP. It aims to find and classify expressions of special meaning in texts written in natural language. These expressions can belong to different predefined classes such as Person, Organization and Location.

The main algorithms that are currently used for NER in English involves Neural Networks that are trained in a supervised way using large corpuses of labeled data. However, in most languages and domains, there is only a very small amount of supervised training data available, and consequently supervised algorithms are hard to train. Specifically for Spanish, NER is a challenging problem because of the lack of resources and labeled data. Compared with supervised and semi-supervised methods, unsupervised approach for entity recognition can overcome the difficulties on requirement of a large amount of labeled data.

In this paper we investigate whether spectral clustering techniques can be successfully applied to NER in Spanish. In order to do so, we transform the original input, defined by the the word2Vector embeddings of entities and context, into a set of orthogonal eigenvectors. Then, We work in the space defined by the

first few eigenvectors, using standard clustering techniques in the reduced space.

The paper is organized as follows. In Section 2, we provide a explanation of the data used to accomplish the study. Section 3 describes the architecture to create the word embeddings. Section 4 presents our framework for NER and Section 5 explains the metric used to evaluate results. In Section 6 the experiments and results are discussed. Finally, Section 7 presents the conclusions.

2 Data

The experimentation of this work will be carried on a corpora that consist of sentences extracted from news articles. The corpora corresponds to the *CoNLL 2002* Shared Task Spanish data, the original source being the *EFE Spanish Newswire Agency*. The data consists of two columns separated by a single space. The first item on each line is a word and the second the named entity tag. In this dataset, there are four types of entities: person names (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC) and other (O). This dataset contains 11,752 sentences, 369,171 words and 26,706 Name Entities.

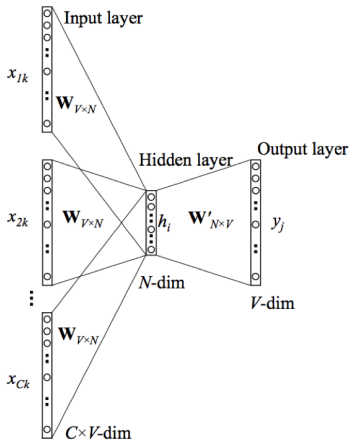
In order to create the word embedding, we added to the *CoNLL 2002* dataset an unannotated corpus of the Spanish Billion Words Corpus [reference] with a total of 1,420,665,810 raw words, 46,925,295 sentences and 3,817,833 unique tokens. Both Corpuses are preprocessed by replacing all non-alphanumeric characters with whitespaces, all numbers with the token "DIGITO" and all the multiple whitespaces with only one whitespace.

3 Word2Vector Embedding

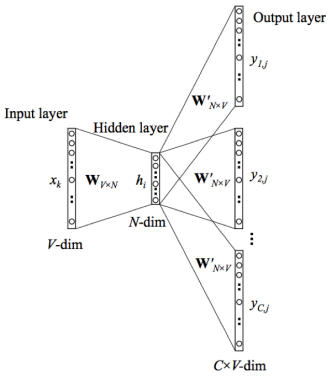
Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus.

The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace. That is, it detects similarities mathematically. Word2vec creates vectors that are distributed numerical representations of word features, features such as the context of individual words. It does so without human intervention.

Word2Vec works in two ways, either using context to predict a target word (a method known as continuous bag of words, or CBOW), or using a word to predict a target context, which is called skip-gram. For both approaches a neural network is trained with a single hidden layer to perform a certain task depending of the approach.



(a) CBOW



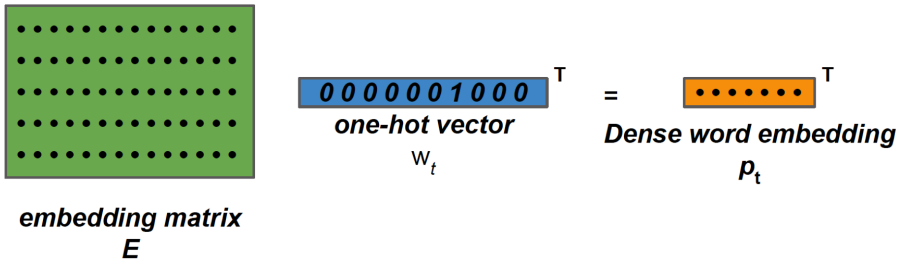
(b) Skip-gram

Figure 1: Word2Vector Algorithm

4 Problem Formulation

Our first goal is to estimate a vector for each entity that captures a large number of precise syntactic and semantic word relationships. For this purpose, we creat an embedding matrix, which will transform the entity's words into a dense word embedding. To do so, we make use of a simple one-hot vector that represent every word as an $\mathbb{R}^{|V|}$ vector with all 0s and one 1 at the index of that word in the $|V|$ vocabulary created by the embedding Figure 2.

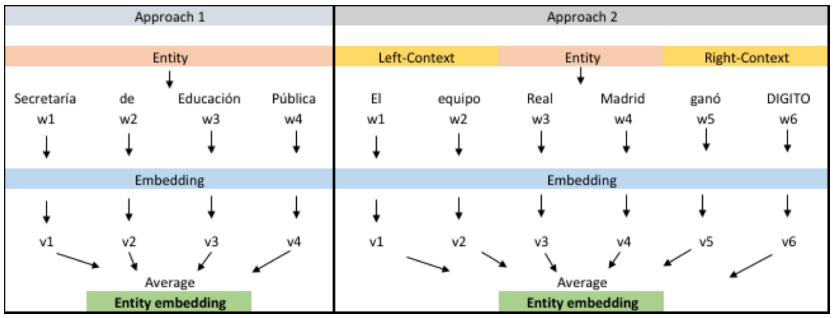
Figure 2: Dense Word Embedding



In this work, each entity can be represented in two different ways. The first one takes into account the embedding of the entity and the second, in addition to the entity, incorporates the embedding of the surrounded words (context).

As each entity can be composed by one or more words, we use the average of the embeddings for each word to create the final entity embedding. Figures 3 shows the two approaches used to compute the final entity embedding.

Figure 3: Entity embedding under both approaches



Finally, the clustering is done using spectral cluster algorithm on the entity embeddings. Results are compared using skip-gram and CBOW algorithms for the word embeddings and using different kernels for the spectral clustering.

5 Evaluation

To compere the different models we ran, we used the Adjusted Rand index [6], that evaluate the accuracy of the predicted labels of the clustering given the ground truth of the label. This index looks at each pair of sample, and evaluate the consistency of the prediction given the true label. Calling C the ground truth clustering and C' the predicted one, there are four possible set of pairs:

S_{11} = {pairs belonging to the same cluster both in C and C'}
S_{00} = {pairs belonging to different cluster both in C and C'}
S_{10} = {pairs belonging to the same cluster both C but to different ones in C'}
S_{01} = {pairs belonging to the same cluster both C' but to different ones in C}

Calling with n_{11} the number of sample in the first set (and analogously for the other sets), the Rand index is defined as:

$$R = \frac{n_{11} + n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}} = \frac{n_{11} + n_{00}}{\binom{n}{2}}$$

where n is the total number of sample. This index range from 0 (when C and C' don't agree on any pair of points) and 1 (when C and C' agree on every pairs). The Adjusted Rand index is defined by rescaling it considering the value that a random clustering would have, doing so the index value is 0 for random clusters.

6 Experiments and Discussion

Parameter of the model:

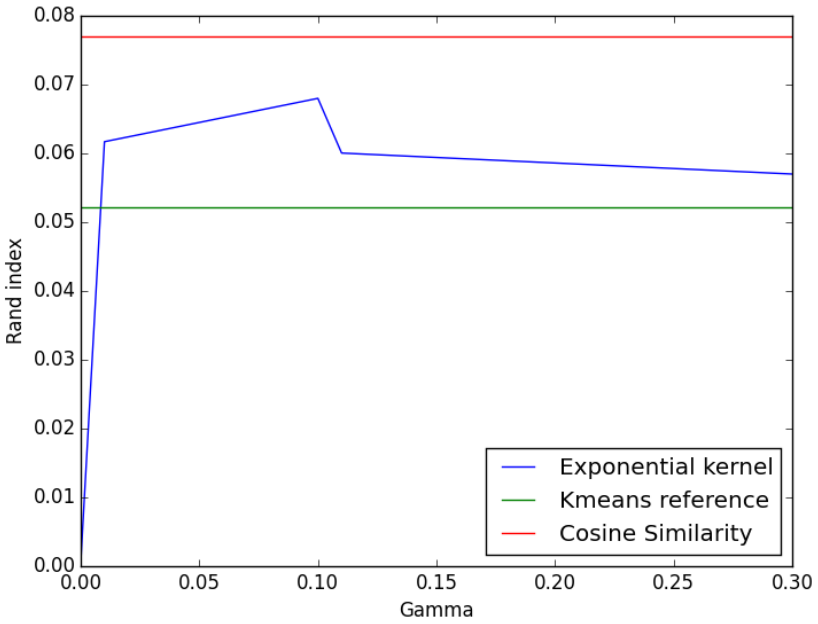


Figure 4: The blue line show the rand index for the exponential kernel model trained for different value of γ. The red line is the best model (cosine similarity) and the green line is the kmeans clustering, reported here as a bench mark

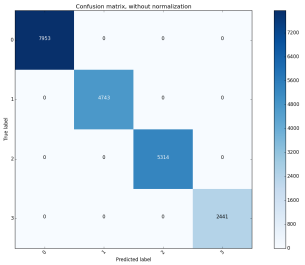


Figure 5: True label distribution

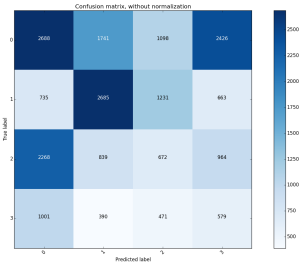


Figure 6: Kmeans Clustering

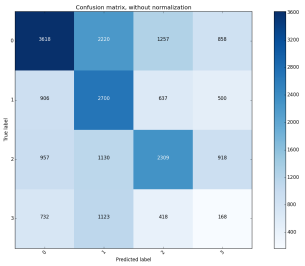


Figure 7: Spectral(exponential kernel)

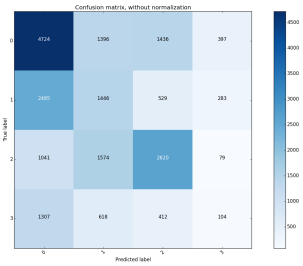


Figure 8: Spectral(Cosine similarity)

- Size of the word embedding space: d_E.
- technique used for the word embedding training: cbow or skip-gram.
- Number of words embedding considered around the target word: n_w.
- Kernell used for the spectral clustering: cosine similarity
  1. Cosine similarity
  2. Knearest neighbours
  3. $k(x, y) = exp(-\gamma||x - y||_1)$

Training the model with different combination of this parameter we have found that the best model is the one that uses an embedding $d_E = 100$ trained with cbow, an average over $n_w = 3$ words, and the cosine similarity as the kernel for spectral clustering. The spectral clustering has also a comparable Rand index value for small values of γ. In the plot (**??**) we reported the results for this two model compared with the Rand index obtained applying kmeans clustering to the embedding vectors.

7 Conclusions

References

[1] http://www.cnts.ua.ac.be/conll2002/ner/

[2] http://www.cims.nyu.edu/∼bandeira/TenLecturesFortyTwoProblems.pdf

[3] Popescu, M., & Hristea, F. (2011). State of the art versus classical clustering for unsupervised word sense disambiguation. Artificial Intelligence Review, 35(3), 241?264. http://dx.doi.org/10.1007/s10462-010-9193-7.

[4] http://crscardellino.me/SBWCE/

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

[6] Rand, William M.: Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association, 66(336):846? 850, 1971.