Procesamiento de Lenguaje Natural

Introducción

Representaciones de documentos

Procesamiento de Lenguaje Natural

Embeddings

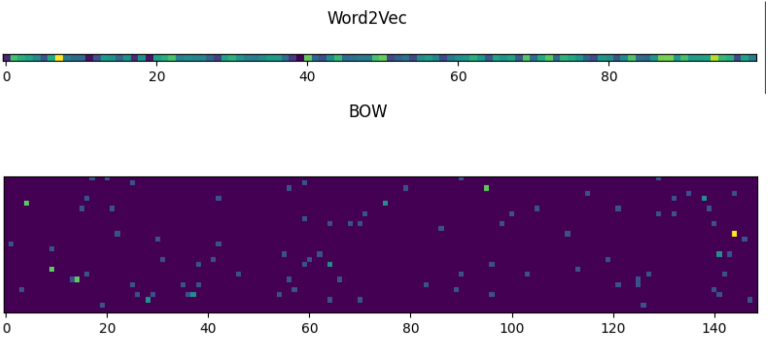
Mauricio Toledo-Acosta
mauricio.toledo@unison.mx

Departamento de Matemáticas
Universidad de Sonora

Section 1

Introducción

Sparsity problem



Bengio, 2003

Journal of Machine Learning Research 3 (2003) 1137–1155 Submitted 4/02; Published 2/03

A Neural Probabilistic Language Model

Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Réjean Ducharme DUCHARME@IRO.UMONTREAL.CA
Pascal Vincent VINCENTP@IRO.UMONTREAL.CA
Christian Jauvin JAUVINC@IRO.UMONTREAL.CA
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal, Montréal, Québec, Canada

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

The original paper

Antecedentes, 1991

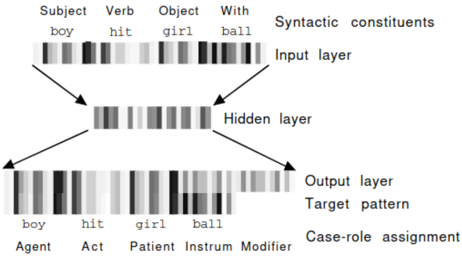


Figure 2: **Snapshot of basic FGREP simulation.** The input and output layers of the network are divided into assemblies, each holding one word representation at a time. Each unit in an input assembly is set to the activity value of the corresponding component in the lexicon entry. The input layer is fully connected to the hidden layer and the hidden layer to the output layer. Connection weights are omitted from the figure. If the network has successfully learned the task, each output assembly forms an activity pattern identical to the lexicon representation of the word filling that role. The correct role assignment is shown at the bottom of the display. This pattern forms the output target for the network. Grey-scale values from white to black are used in the figure to code the unit activities, which vary within the range [0,1].

Antecedentes, 1991

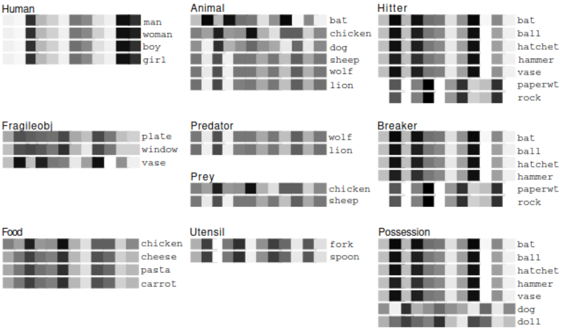


Figure 3: **Final representations.** The representations for the synonymous words {**man**, **woman**, **boy**, **girl**}, {**fork**, **spoon**}, {**wolf**, **lion**}, {**plate**, **window**}, {**ball**, **hatchet**, **hammer**}, {**paperwt**, **rock**} and {**cheese**, **pasta**, **carrot**} have become almost identical.

Word2Vec

Sentence	**He poured himself a cup of coffee**
Target	**himself**

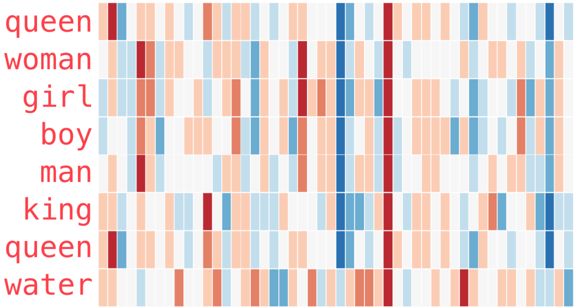
- Continuous Bag-Of-Words

input	*He, poured, a, cup*
output	*himself*

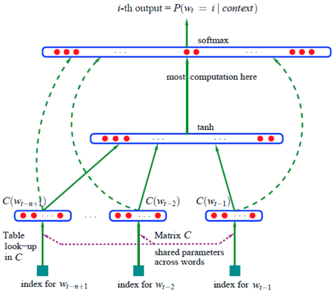
- Skip-gram model

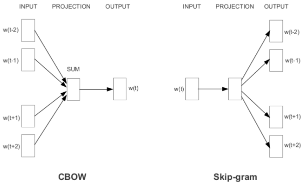
input	*himself*
output	*He, poured, a, cup*

Original Paper


queen
woman
girl
boy
man
king
queen
water

Diferencias entre los enfoques de NPLM y Word2Vec





GloVe, 2014

- *In contrast to word2vec, GloVe seeks to make explicit what word2vec does implicitly: Encoding meaning as vector offsets in an embedding space – seemingly only a serendipitous by-product of word2vec – is the specified goal of GloVe.*
- There are no vectors for OOV words.

GloVe, Original paper

FastText, 2017

Enriching Word Vectors with Subword Information

Piotr Bojanowski* and **Edouard Grave*** and **Armand Joulin** and **Tomas Mikolov**
Facebook AI Research
{bojanowski,egrave,ajoulin,tmikolov}@fb.com

- Extension of the continuous skipgram `word2vec` model (2013), which takes into account subword information.
- Each word is represented as a bag of character *n*-grams. A vector representation is associated to each character n-gram.
- Words being represented as the sum of these representations.

Original Paper

FastText, 2017

- **Example:** Consider *where* and $n = 3$, it will be represented by the character n-grams:

$$\langle \text{wh}, \ \text{whe}, \ \text{her}, \ \text{ere}, \ \text{re} \rangle$$

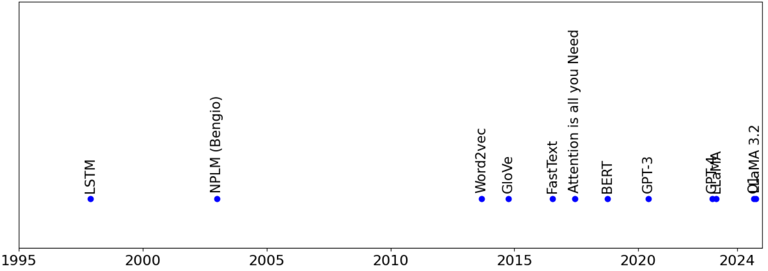
and the special sequence $\langle \text{where} \rangle$.

$$\langle \text{her} \rangle \neq \text{her}$$

- `FastText` computes valid representations for OOV words (out-of-vocabulary) by taking the sum of its n-grams vectors.

Original Paper, Vectors

Timeline


LSTM
NPLM (Bengio)
Word2vec
GloVe
FastText
Attention is all you Need
BERT
GPT-3
GPT-4
LLaMA
O1
LLaMA 3.2
1995
2000
2005
2010
2015
2020
2024

Section 2

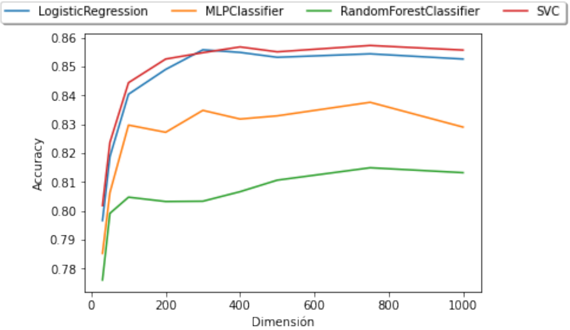
Representaciones de documentos

¿Cómo representamos documentos?

- Promedio de vectores de palabras (centroide).
- Promedio pesado de vectores de palabras.
- Usando redes neuronales.
- Usando embeddings de documentos:
  - `doc2vec`: Le and Mikolov in 2014 introduced the Doc2Vec algorithm, which usually outperforms such simple-averaging of Word2Vec vectors. The basic idea is: act as if a document has another vector, which contributes to all training predictions, and is updated like other word-vectors, but we will call it a doc-vector.

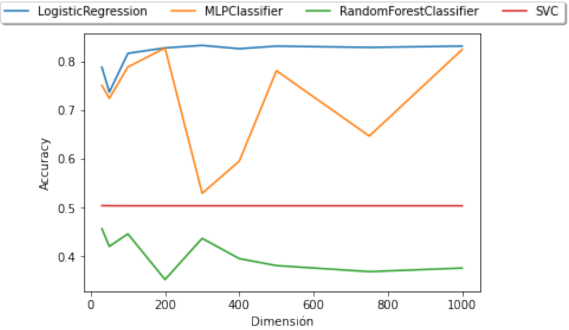
    Original paper Gensim's doc2vec
  - `Bert`-based embeddings.

Efecto de la dimensión



La representación de cada documento está dada por el promedio de cada vector de `word2vec`.

Efecto de la dimensión



La representación de cada documento está dada por el promedio de cada vector de `word2vec`. Esta representación se reescala para tener norma 1.

¿Aún son vigentes estos modelos?