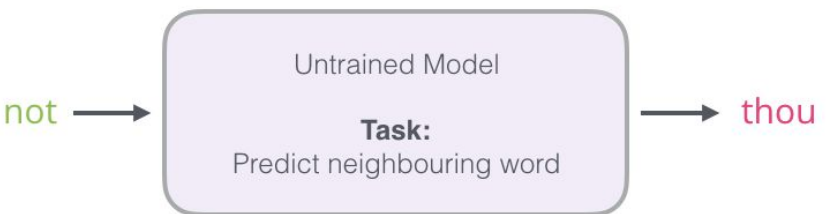
Introducción al Procesamiento de Lenguaje Natural

Clase 5. Inferencia en word embeddings

Repaso…

Modelando con skipgram => PROBLEMA

Change Task from



To:

not
thou

Untrained Model

Task:
Are the two words neighbours?

0.90

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

Problema!
Todos ejemplos positivos…

OVERFITTING

factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

Negative sampling

input word	output word	target
not	thou	1
not		0
not		0
not	shalt	1
not	make	1

Negative examples

Fuente: https://jalammar.github.io/illustrated-word2vec/

Negative sampling

Pick randomly from vocabulary (random sampling)

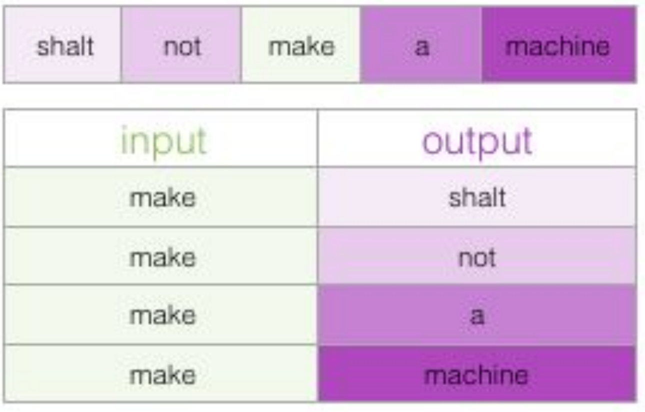
input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	make	1

Word	Count	Probability
aardvark		
aarhus		
aaron		
taco		
thou		
zyzzyva		

Fuente: https://jalammar.github.io/illustrated-word2vec/

La fórmula mágica de w2vec

Skipgram



Negative Sampling

input word	output word	target
make	shalt	1
make	aaron	0
make	taco	0

Fuente: https://jalammar.github.io/illustrated-word2vec/

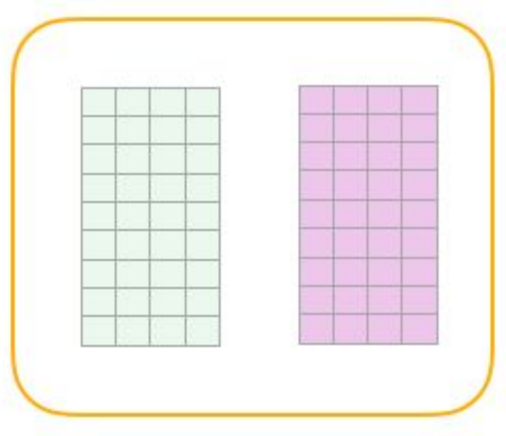
Intuición sobre el proceso de training e inferencia en word2vec

Parámetros a estimar (matrices)

dataset

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

model



factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

Parámetros a estimar (matrices)

dataset

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

Embedding

Context

vocab_size

aardvark
aarhus
aaron
...
...
...
...
...
zyzzyva

embedding_size

aardvark
aarhus
aaron
...
...
...
...
...
zyzzyva

embedding_size

factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

Look up embeddings

Embedding

aardvark
aarhus
aaron
…
not
…
…
…
zyzzyva

Context

aardvark
aarhus
aaron
…
taco
…
thou
…
zyzzyva

not

aaron

taco

thou

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

input word	output word	target	input • output
not	thou	1	0.2
not	aaron	0	-1.11
not	taco	0	0.74

¿A a qué operación es equivalente la tercera columna?

Por ende… ¿qué está midiendo?

=> Este paso mide el dot.product de la input_word con cada término del contexto.

=> Mide la SIMILITUD entre cada par de vectores

Para ponerle un

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

input word		output word		target	input • output
not		thou		1	0.2
not		aaron		0	-1.11
not		taco		0	0.74

¿A a qué operación es equivalente la tercera columna?

Por ende… ¿qué está midiendo?

=> Este paso mide el dot.product de la input_word con cada término del contexto.

=> Mide la SIMILITUD entre cada par de vectores

Para ponerle una notación más linda:

$$\mathrm{Sim}(\mathrm{w},\mathrm{c}) \approx \mathrm{w} \cdot c$$

factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

input word	output word	target	input • output	sigmoid()
not	thou	1	0.2	0.55
not	aaron	0	-1.11	0.25
not	taco	0	0.74	0.68

factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

input word	output word	target	input • output	sigmoid()
not	thou	1	0.2	0.55
not	aaron	0	-1.11	0.25
not	taco	0	0.74	0.68

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

factor~data
EIDAES_UNSAM

Fuente: https://jalammar.github.io/illustrated-word2vec/

¿Cómo calcula SkipGram P(+|w, c)?

- Tenemos muchas palabras de contexto
- => Asumimos independencia y las probabilidades se multiplican

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

Fuente: https://jalammar.github.io/illustrated-word2vec/

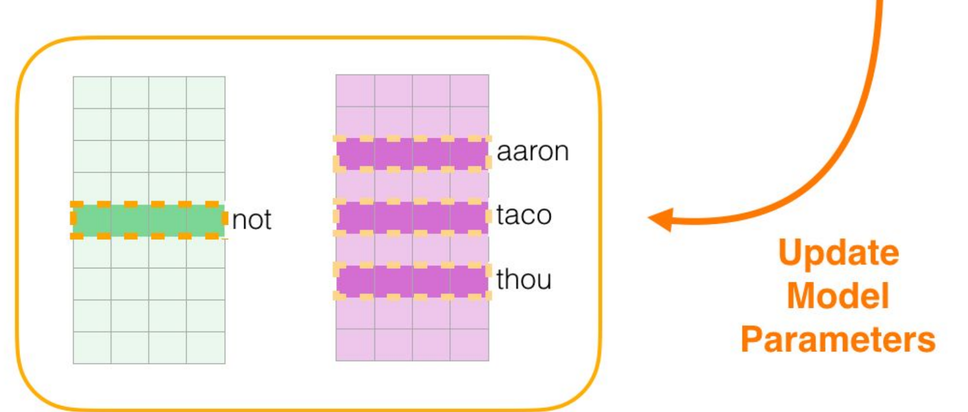
Veamos una iteración de training…

input word		output word		target	input • output	sigmoid()	Error
not		thou		1	0.2	0.55	0.45
not		aaron		0	-1.11	0.25	-0.25
not		taco		0	0.74	0.68	-0.68

Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



Update Model Parameters

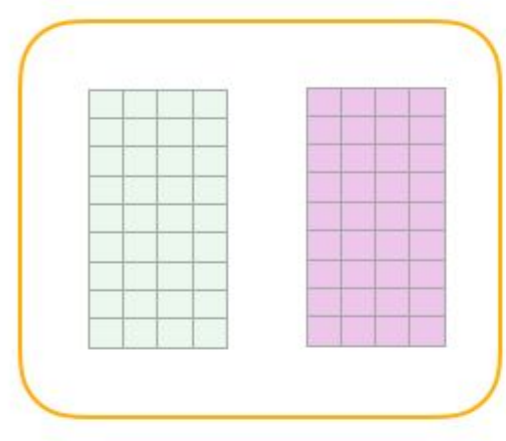
Fuente: https://jalammar.github.io/illustrated-word2vec/

Veamos una iteración de training…

dataset

input word	output word	target
not	thou	1
not	aaron	0
not	taco	0
not	shalt	1
not	mango	0
not	finglonger	0
not	make	1
not	plumbus	0
...	...	...

model



Fuente: https://jalammar.github.io/illustrated-word2vec/

Un poquito más formalmente

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log\sigma(c_{pos}\cdot w)+\sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

- Dados
  - un conjunto de ejemplos positivos y negativos
  - un set inicial de embeddings (embedding + context)
- El objetivo del proceso de entrenamiento es ajustar esos vectores tal que
  - se maximice la similaridad (dot.product) entre las target_word y sus contextos positivos => (w, c_pos)
  - se minimice la similaridad entre los pares (w, c_neg)
- ¿Cómo? Stochastic Gradient Descent (queda para la materia de Deep Learnng?

factor~data
EIDAES_UNSAM

Resumen de inferencia en word2vec

- Comienza con V vectores aleatorios de dimensión d como embeddings iniciales
- Entrena un clasificador basado en la similitud de embeddings:
  - Toma un corpus y pares de palabras que coocurren como ejemplos positivos
  - Toma pares de palabras que no coocurren como ejemplos negativos
  - Entrena el clasificador para distinguirlos ajustando lentamente todos los embeddings para mejorar el rendimiento del clasificador
  - Descarta el código del clasificador y conserva los embeddings

Otros métodos para construir embeddings

- word2vec fue pionero (2013) pero hoy hay métodos mejores

- GloVe: trabaja directamente sobre la matriz de co-ocurrencias

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning

Introduction

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Getting started (Code download)

- Download the latest latest code (licensed under the Apache License, Version 2.0). Look for "Clone or download"
- Unpack the files: unzip master.zip
- Compile the source: cd GloVe-master && make
- Run the demo script: ./demo.sh
- Consult the included README for further usage details, or ask a question

Download pre-trained word vectors

- Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses/pddl/1.0/.
  - Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors, 822 MB download): glove.6B.zip
  - Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip
  - Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip
  - Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, & 200d vectors, 1.42 GB download): glove.twitter.27B.zip
- Ruby script for preprocessing Twitter data

Citing GloVe

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. [pdf] [bib]

Highlights

1. **Nearest neighbors**

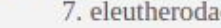
   The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

   0. *frog*
   1. frogs
   2. toad
   3. litoria
   4. leptodactylidae
   5. rana
   6. lizard
   7. eleutherodactylus

3. litoria

4. leptodactylidae

5. rana


7. eleutherodactylus

Referencias

arXiv:1310.4546v1 [cs.CL] 16 Oct 2013

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships. In this paper we present several extensions that improve both the quality of the vectors and the training speed. By subsampling of the frequent words we obtain significant speedup and also learn more regular word representations. We also describe a simple alternative to the hierarchical softmax called negative sampling.

An inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases. For example, the meanings of "Canada" and "Air" cannot be easily combined to obtain "Air Canada". Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.

1 Introduction

Distributed representations of words in a vector space help learning algorithms to achieve better performance in natural language processing tasks by grouping similar words. One of the earliest use of word representations dates back to 1986 due to Rumelhart, Hinton, and Williams [13]. This idea has since been applied to statistical language modeling with considerable success [1]. The follow up work includes applications to automatic speech recognition and machine translation [14, 7], and a wide range of NLP tasks [2, 20, 15, 3, 18, 19, 9].

Recently, Mikolov et al. [8] introduced the Skip-gram model, an efficient method for learning high-quality vector representations of words from large amounts of unstructured text data. Unlike most of the previously used neural network architectures for learning word vectors, training of the Skip-gram model (see Figure 1) does not involve dense matrix multiplications. This makes the training extremely efficient: an optimized single-machine implementation can train on more than 100 billion words in one day.

The word representations computed using neural networks are very interesting because the learned vectors explicitly encode many linguistic regularities and patterns. Somewhat surprisingly, many of these patterns can be represented as linear translations. For example, the result of a vector calculation vec("Madrid") - vec("Spain") + vec("France") is closer to vec("Paris") than to any other word vector [9, 8].

arXiv:1301.3781v3 [cs.CL] 7 Sep 2013

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good reasons - simplicity, robustness and the observation that simple models trained on huge amounts of data outperform complex systems trained on less data. An example is the popular N-gram model used for statistical language modeling - today, it is possible to train N-grams on virtually all available data (trillions of words [3]).

However, the simple techniques are at their limits in many tasks. For example, the amount of relevant in-domain data for automatic speech recognition is limited - the performance is usually dominated by the size of high quality transcribed speech data (often just millions of words). In machine translation, the existing corpora for many languages contain only a few billions of words or less. Thus, there are situations where simple scaling up of the basic techniques will not result in any significant progress, and we have to focus on more advanced techniques.

With progress of machine learning techniques in recent years, it has become possible to train more complex models on much larger data set, and they typically outperform the simple models. Probably the most successful concept is to use distributed representations of words [10]. For example, neural network based language models significantly outperform N-gram models [1, 27, 17].

1.1 Goals of the Paper

The main goal of this paper is to introduce techniques that can be used for learning high-quality word vectors from huge data sets with billions of words, and with millions of words in the vocabulary. As far as we know, none of the previously proposed architectures has been successfully trained on more

factor~data
EIDAES_UNSAM