Word Embeddings

Parte 2

Bibliografía

Jurafsky and Martin 3rd edition. Cap 6. https://web.stanford.edu/~jurafsky/slp3/

Word Embedding basados en predicción de contextos

Se obtienen word embeddings al optimizar una función de objetivo para predecir palabras a partir de contextos.

Es decir, se minimiza una función de pérdida formulada para predicciones en el texto.

<u>Existen varios métodos:</u>

CBOW

Skip-gram

SGNS

FastText (subword information!)

...

Vamos a ver SGNS (Mikolov et al., 2013)

SGNS (skip-gram with negative samipling)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.

Instead of **counting** how often each word *w* occurs near "*apricot*"

- Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?

We don't actually care about this task

- But we'll take the learned classifier weights as the word embeddings

Big idea: **self-supervision**:

- A word c that occurs near apricot in the corpus cats as the gold "correct answer" for supervised learning
- No need for human labels
- Bengio et al. (2003); Collobert et al. (2011)

Extraído de https://web.stanford.edu/~jurafsky/slp3/

SGNS (skip-gram con negative samipling)

Approach: predict if candidate word *c* is a "neighbor"

1. Treat the target word *t* and a neighboring context word *c* as **positive examples**.
2. Randomly sample other words in the lexicon to get negative examples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Skip-Gram Classifier

(assuming a +/- 2 word window)

…lemon, a [tablespoon of apricot jam, a] pinch…
c1 c2 [target] c3 c4

Goal: train a classifier that is given a candidate (**w**ord, **c**ontext) pair
(apricot, jam)
(apricot, aardvark)
…

And assigns each pair a probability:

$P(+|w, c)$

$P(-|w, c) = 1 - P(+|w, c)$

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Skip-Gram Training data

...lemon, a [tablespoon of apricot jam, a] pinch...

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

For each positive example we'll grab k negative examples, sampling by frequency

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Skip-Gram Training data

…lemon, a [tablespoon of apricot jam, a] pinch…

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Extraído de https://web.stanford.edu/~jurafsky/slp3/

How Skip-Gram Classifier computes $P(+|w, c)$

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

This is for one context word, but we have lots of context words.
We'll assume independence and just multiply them:

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

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

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Loss function for one *w with* c_{pos}, c_{neg1} ...c_{negk}

Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

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

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Skip-gram classifier: summary

A probabilistic classifier, given

- a test target word *w*
- its context window of *L* words $c_{1:L}$

Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings).

To compute this, we just need embeddings for all the words.

Extraído de https://web.stanford.edu/~jurafsky/slp3/

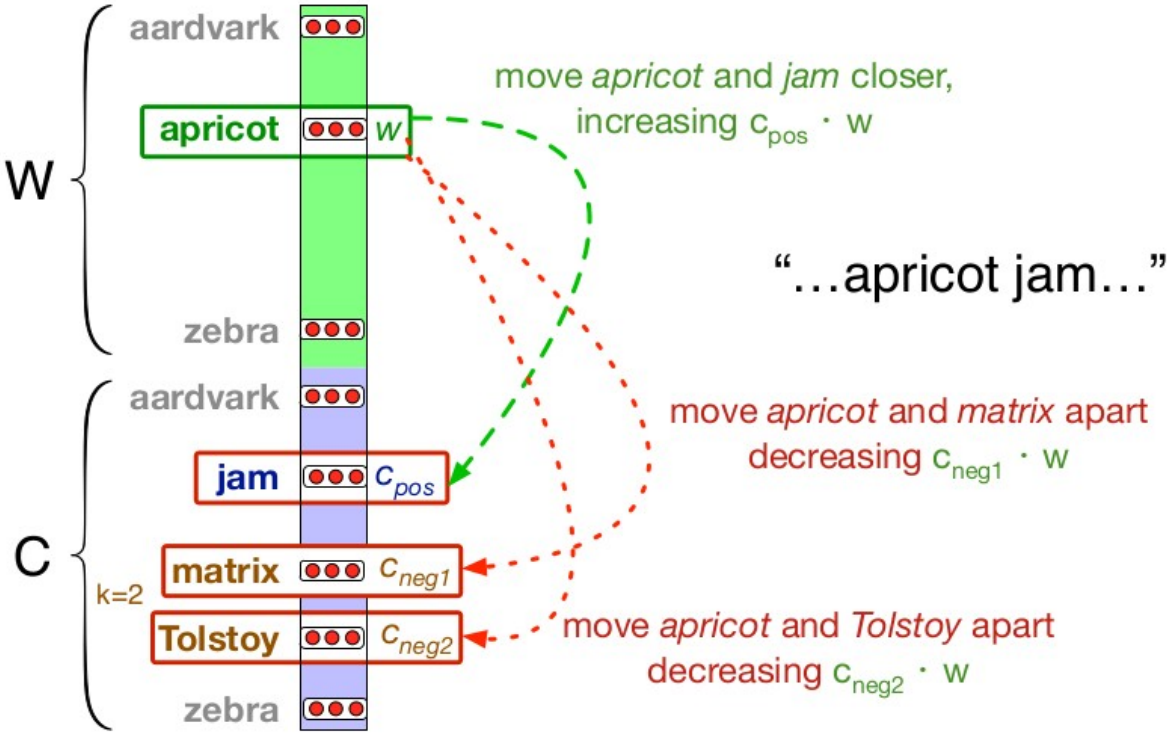
Learning the classifier

How to learn?

- Stochastic gradient descent!

We'll adjust the word weights to

- make the positive pairs more likely
- and the negative pairs less likely,
- over the entire training set.



Extraído de https://web.stanford.edu/~jurafsky/slp3/

Reminder: gradient descent

- At each step
  - Direction: We move in the reverse direction from the gradient of the loss function
  - Magnitude: we move the value of this gradient $\frac{d}{dw} L(f(x;w),y)$ weighted by a **learning rate** η
  - Higher learning rate means move *w* faster

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

Extraído de https://web.stanford.edu/~jurafsky/slp3/

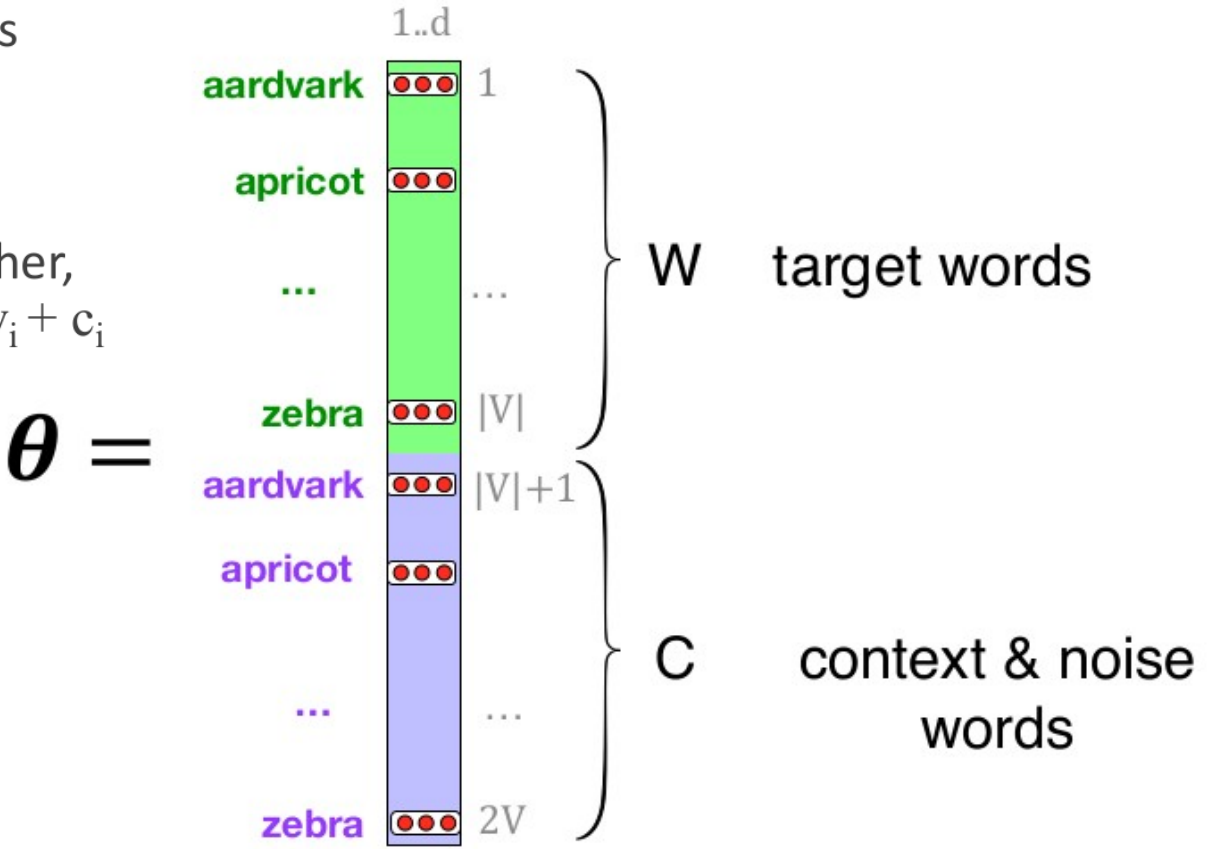
Two sets of embeddings

SGNS learns two sets of embeddings

Target embeddings matrix W

Context embedding matrix C

It's common to just add them together, representing word *i* as the vector $w_i + c_i$



$\boldsymbol{\theta} =$

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random d-dimensional vectors as initial embeddings

Train a classifier based on embedding similarity

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the embeddings.

Extraído de https://web.stanford.edu/~jurafsky/slp3/

fastText (Bojanowski et al., 2016)

Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. (2016). Enriching Word Vectors with Subword Information

Extiende skipgram con subword information

- **Vector de cada palabra a partir de la suma de vectores de ngramas sus caracteres**
- **Se mejoran las representaciones de palabras con baja frecuencia**
- **Vectores para palabras fuera de vocabulario (oov)**

En el sitio de fastText (https://fasttext.cc/) pueden encontrar:

- **Vectores y modelos entrenados para varios idiomas (157)**
- **Código para entrenar/utilizar modelos entrenados**
- **Clasificadores de textos**

Evaluación de Word Embeddings

Dos tipos de evaluaciones:

Extrínseca: en una tárea final probar distintos embeddings y comparar resultados

Intrínseca: tareas específicas para evaluar la calidad de los embeddings respecto a un conjunto de referencia (ej. Word Similarity, Word Analogy, Concept Categorization, etc.)

Vamos a ver Word Similarity y Analogy

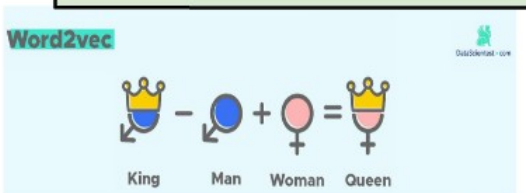
Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. C. J. (2019). Evaluating word embedding models: methods and experimental results. APSIPA transactions on signal and information processing, 8. https://arxiv.org/pdf/1901.09785.pdf

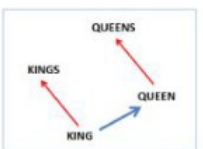
Evaluación de Word Embeddings

Luego de construir colecciones de vectores de palabras se suele querer evaluarlos. Hay dos grandes categorías.

INTRÍNSECOS

- Analogías
  - *hombre* es a *mujer* lo que *rey* a ____
- Similitud
  - *¿Cuánto se parece una manzana a un durazno?*
  - *¿Y una silla a una mesa?*
  - *¿Y un camión a una luciérnaga?*
- Otros...





EXTRÍNSECOS

- Recuperación de información
- Chatbots
- Question Answering
- Traducción automática
- Análisis de sentimientos
- Análisis de emociones

Extraído de https://eva.fing.edu.uy/pluginfile.php/404731/mod_resource/content/2/VectoresDePalabras.pdf

Word Similarity

Compare to human scores on word similarity-type tasks:

- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Word Similarity

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

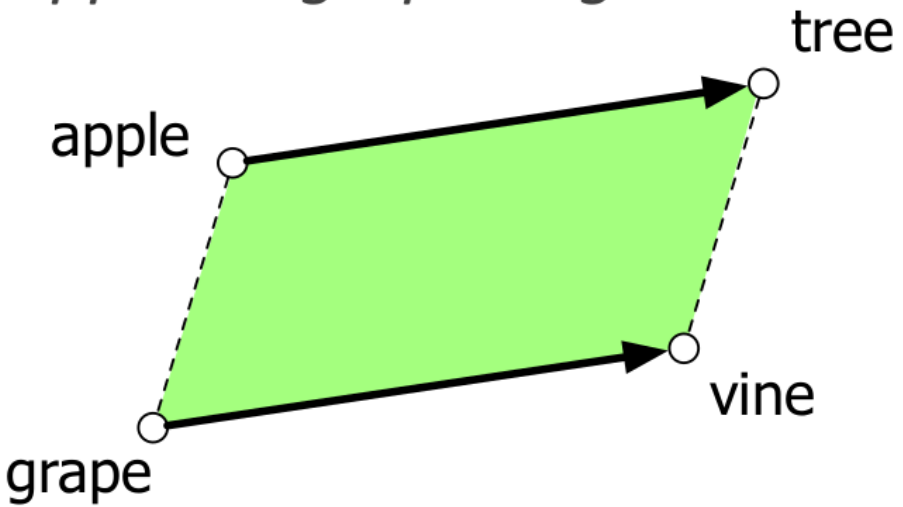
Extraído de https://web.stanford.edu/~jurafsky/slp3/

Word Analogy

The classic parallelogram model of analogical reasoning
(Rumelhart and Abrahamson 1973)

To solve: *"apple is to tree as grape is to _____"*

Add $\overrightarrow{tree} - \overrightarrow{apple}$ to $\overrightarrow{grape}$ to get $\overrightarrow{vine}$



Extraído de https://web.stanford.edu/~jurafsky/slp3/

Word Analogy

The parallelogram method can solve analogies with both sparse and dense embeddings (Turney and Littman 2005, Mikolov et al. 2013b)

$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$

$\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}}$ is close to $\overrightarrow{\text{Rome}}$

For a problem a:a*::b:b*, the parallelogram method is:

$$\hat{b}^* = \underset{x}{\text{argmax}}\ \text{distance}(x, a^* - a + b)$$

Extraído de https://web.stanford.edu/~jurafsky/slp3/

Caveats with the parallelogram method

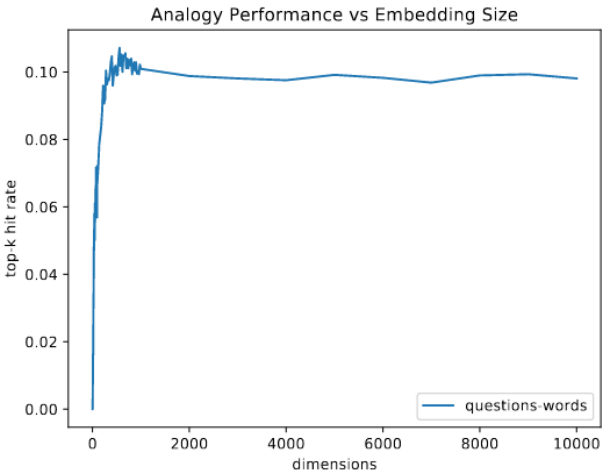
It only seems to work for frequent words, small distances and certain relations (relating countries to capitals, or parts of speech), but not others. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)

Understanding analogy is an open area of research (Peterson et al. 2020)

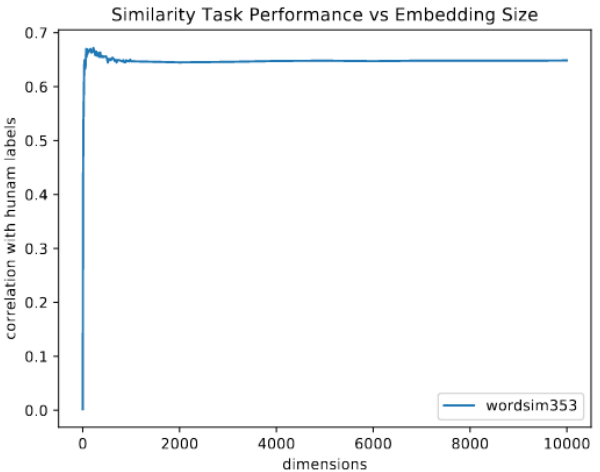
Extraído de https://web.stanford.edu/~jurafsky/slp3/

Dimensión de Word Embeddings

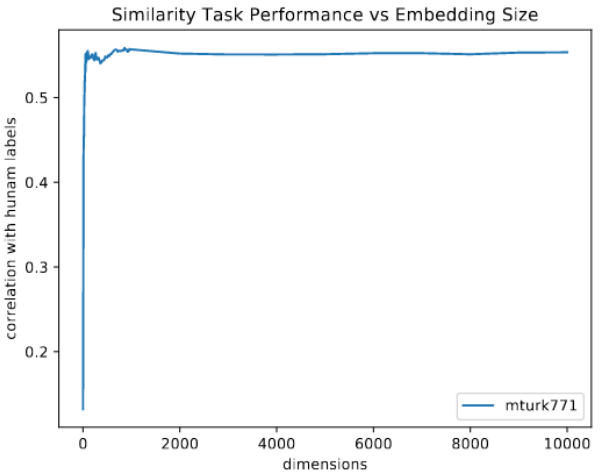
Yin, Z., & Shen, Y. (2018). On the dimensionality of word embedding. Advances in neural information processing systems, 31.



(a) Google Analogy Test



(b) WordSim353 Test



(c) MTurk771 Test

Figure 3: GloVe: over-parametrization does not significantly hurt performance

Extraído de https://proceedings.neurips.cc/paper/2018/file/b534ba68236ba543ae44b22bd110a1d6-Paper.pdf

Algunos Recursos

Gensim https://radimrehurek.com/gensim/

Librería para entrenamiento y manejo de word embeddings. Vectores entrenados.

GloVe https://nlp.stanford.edu/projects/glove/

Sitio de GloVe. Código, vectores entrenados con distintos coprpus.

FasText https://fasttext.cc/

Sitio de fastText. Código, modelos entrenados. Modelos para distintos idiomas.

Representaciones Contextualizadas

Hasta ahora cada representación contempla todos los sentidos de la palabra.

No hay tratamiento de homonimia o polisemia en este punto.

Idea de embeddings contextualizados: un vector para cada ocurrencia de palabra.

Ocurrencias distintas tienen vectores distintos. Representa el significado de esa ocurrencia de la palabra.

Vamos a volver sobre esto.

En la que viene empezamos con redes neuronales