

Word Embeddings

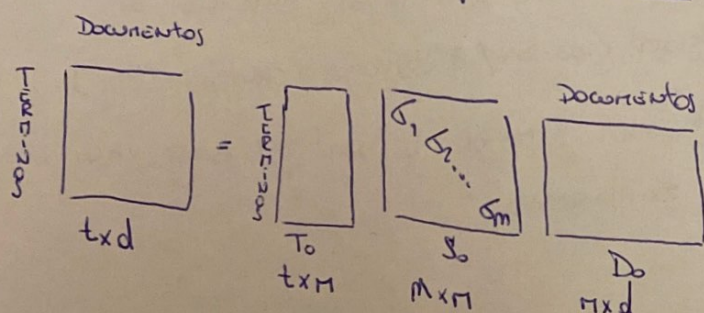
6-1-word-embedding

8/

- Dado un gran corpus de palabras, poder determinar que palabras esta cerca de cual en un espacio vectorial.
- Cada palabra la represento como un vector.

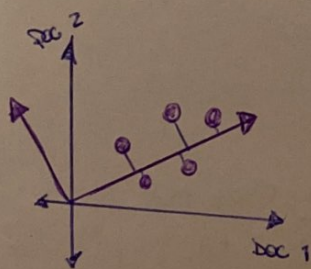
⑦ LATENT SEMANTIC ANALYSIS (LSA - 1990)

- ↳ ES REDUCIR UNA FUNCIÓN DE COSTO.
- ↳ APLICA Singular Value Disposition (SVD)



TRUNCANDO:
Bajo Dimensiones y cada palabra queda representada por un vector.

- SVD
- De queda donde esta x se mueve hacia las dimensiones que mejor ajusta los datos.
 - ES UN METODO DE Reducción de Dimensionalidad. (Reducir dimensiones no útiles)
 - USA COSINE para buscar distancia entre palabras



• Busca el vector que mejor ajusta.

Antes de usar SVD se puede aplicar una TRANSFORMACIÓN

TF-IDF

$$tf-idf(t,d) = tf(t,d) \cdot idf(t)$$

LOG-ENTROPY

$$\log Ent(t,d) = \log (tf(t,d) + 1) \cdot w_p$$

LSA - ALTERNATIVO

↳ Aplicar SVD sobre la matriz de TERM-CONTEXT (matriz con ventanas)

LSA - TIPS

- Nro. estándar de Dimensiones $K=300$
- El "K" óptimo depende del número de tópicos distintos existentes en los textos y de la tarea a realizar.
- Muchas veces se "descarta" la primera dimensión o hasta las primeras 50 (el primero son cosas generales, vector general / stopwords / centro de masa).
- Cosine entre palabras no es absoluto. A mayor "K" las palabras van a estar más distantes entre sí \Rightarrow MENOR SIMILARIDAD.

② WORD2VEC (SKIP-GRAM)

- Red Neuronal (Multiplicación de Matrices)
- Al igual que con LSA, voy bajando dimensiones y si empiezo a ver solapamiento. Tengo que reducir mientras estoy tirando dimensiones poco útiles y sin tirar las útiles (existe un "K" óptimo).

Función Objetivo

$$O = \sum \log p(w_{t+j} | w_t)$$

$$= \log P(\text{Argentino} | \text{choripan}) + \log P(\text{consiste} | \text{choripan})$$

"Maximizar el \log (Probabilidad de cada uno de los outputs)"

Función SOFT-MAX

Argentino

$$P(w_k | w_j) = \frac{\exp(c_k \cdot v_j)}{\sum_{i \in V} \exp(c_i \cdot v_j)}$$

choripan

"Argentino choripan consiste"

-1 +1

$$P(w_k | w_j) = \frac{\exp(c_k \cdot c_j)}{\sum \exp(c_i \cdot c_j)}$$

Obs

Muy Costoso
Computacionalmente

NEGATIVE SAMPLING

en vez de usar el
logaritmo de la
probabilidad,

funcion
Sigmoidea $\sigma(x) = \frac{e^x}{1+e^x}$

Nueva funcion objetivo
a MAXIMIZAR

$$O = \sum_{k=1}^{|V|} \sum_{j=1}^{|V|} \#(w_k, w_j) \left[\log(\bar{c}_k \cdot \bar{v}_j) + \sum_{i=1}^K \mathbb{E}_{w_i \sim P(w)} [\log \sigma(-\bar{w}_i \cdot \bar{v}_j)] \right]$$

La solución optima
Cumple:

con $\sigma(x) = \frac{e^x}{1+e^x}$

$$\bar{c}_k \cdot \bar{v}_j = \text{PMI}(w_k, w_j) - \log(k) \quad (\text{El PMI de sus palabras con un correimiento})$$

Correimiento

El "SKIP-GRAM" factoriza el PMI en 2 matrices (con un correimiento)

$$C \cdot V = X^{\text{PMI} - \log(k)}$$

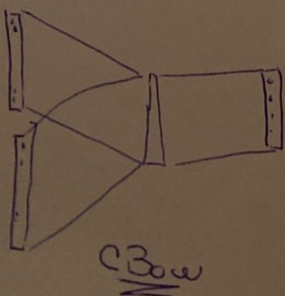
Factorización en matrices
mas pequeñas.

PMI matriz grande por lo general

ES un
tipo de
Factorización
de la matriz
del PMI.

③ WORD2Vec CBow \Rightarrow OTRA VARIANTE

Por medio de las palabras del
Contexto, se predice la
palabra central!



• Depende el caso porque o no
ser mejor.

Word Analogy Task

analogías entre vectores.

- la mejor técnica de Word embeddings se la le ha probado y experimentación. A priori no hay una predefinida.

LSA - 30 años VS Word2vec - 7 años

- funciona mejor con corpus mas pequeños

- Mejor desempeño con corpus mas grandes.
(gigantes $\approx 10^8$ palabras)

Métricas de rendimiento

- Menor dependencia a la dimensionalidad


- ⇒ Silhouette coefficient
- ⇒ Spearman correlation

Entrenar \Rightarrow Reducir función de costo

LSA no conoce a priori que costo estoy reduciendo.

Word2vec tampoco una conocida (la matriz del PPM - Cte)

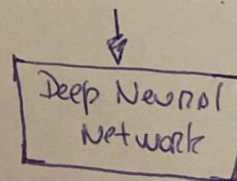
- Existen Word2vec pre-entrenados (Facebook, Stanford, Google, etc.)

VER que palabras aparecen juntas en textos y las representa vectorialmente] 

Embeddings

Aplicaciones

→ Word Embedding como features: pasar palabras a números.



→ Predicción

→ Como método para investigar corpus

→ Como métrica de similitud semántica externa