Grau en Intel.ligència Artificial

Processament del Llenguatge Humà
11. Word Vectors

Outline

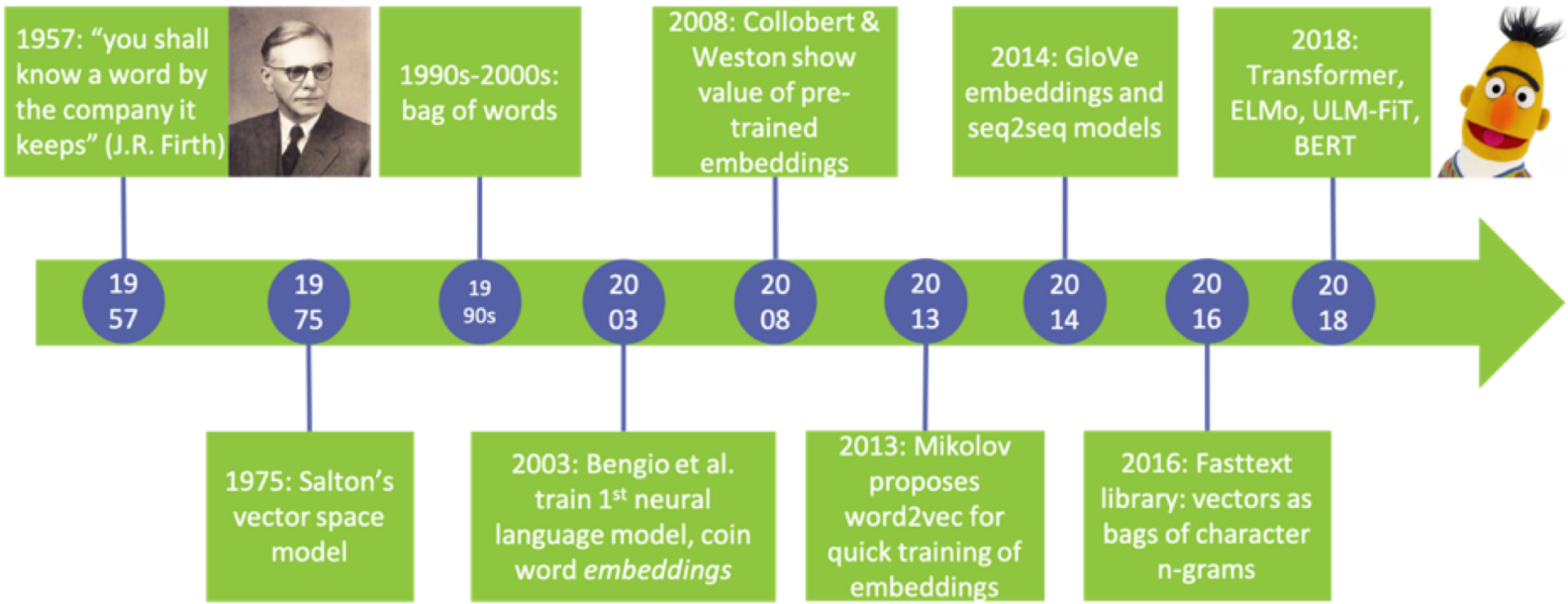
Pregunta

¿Qué sabes sobre Vectores de Palabras o Incrustaciones de Palabras?

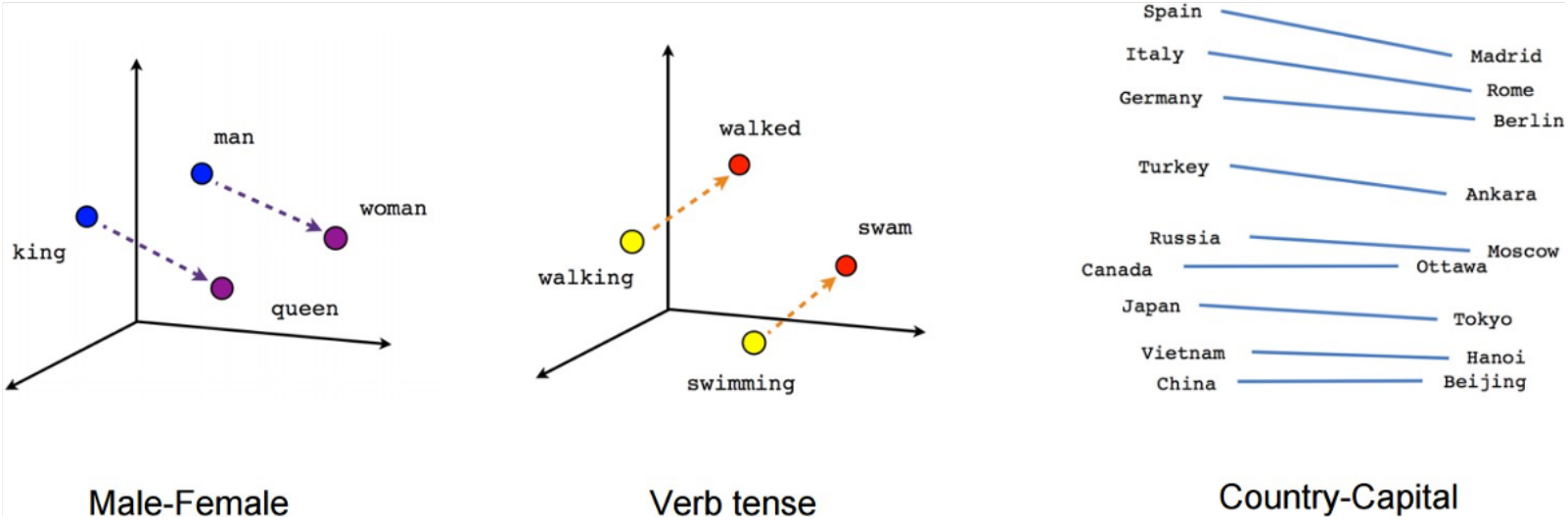
Un Vector de Palabras es una representación numérica de una palabra

- Los Vectores de Palabras permiten realizar operaciones aritméticas en un texto:
  - Ejemplo: *time* + *flies*
- Los Vectores de Palabras también se han denominado:
  - Representación Semántica de Palabras
  - Representación de Vectores de Palabras

Línea de tiempo



Vectores de palabras



Hipótesis de la Distribución y Contextualidad

(Frege, 1884)

Nunca preguntes por el significado de una palabra en aislamiento, sino solo en el contexto de una oración.

(Wittgenstein, 1953)

Para una gran cantidad de casos... el significado de una palabra es su uso en el lenguaje.

(Firth, 1957)

Conocerás una palabra por las compañías que mantiene.

(Harris, 1954)

Las palabras que ocurren en contextos similares tienden a tener un significado similar.

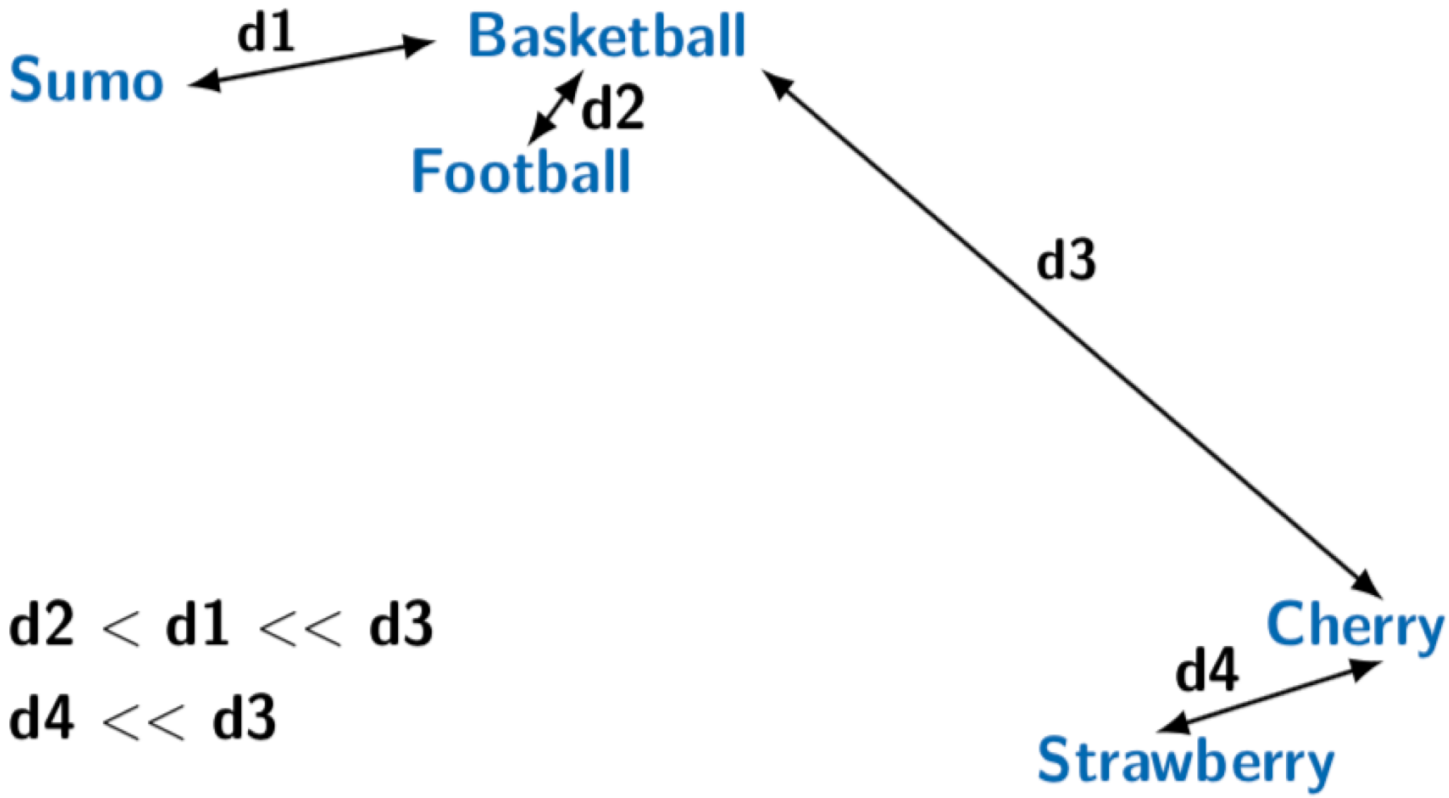
Los Vectores de Palabras permiten procesar oraciones con Machine Learning

Las oraciones son secuencias de símbolos:
Los Vectores de Palabras (incrustaciones de palabras) son representaciones vectoriales de palabras, la unidad "natural" para resolver tareas de procesamiento del lenguaje natural.

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Los Vectores de Palabras permiten procesar oraciones con Machine Learning

Las representaciones vectoriales pueden ayudarnos a encontrar **significados similares**... pero necesitamos definir un concepto de **distancia**.



Outline

Cómo representar una palabra: vectores One-Hot

- **Vector One-Hot** (dim == tamaño del vocabulario)
  - Vector muy grande (incluso millones de palabras)
  - Representaciones dispersas y ortogonales
  - No proporciona información sobre cómo se relacionan las palabras
  - No permite una distancia vectorial útil
  - Utiliza una gran cantidad de memoria (si no se usan matrices dispersas)
  - Codificación habitual de variables categóricas para modelos lineales y SVM con los núcleos estándar

$$
\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \dots \end{bmatrix} \quad \text{to} \quad (1)
$$
$$
\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix} \quad \text{be} \quad (3)
$$
$$
\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & \dots \end{bmatrix} \quad \text{or} \quad (2)
$$
$$
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & \dots \end{bmatrix} \quad \text{not} \quad (5)
$$
$$
\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \dots \end{bmatrix} \quad \text{to} \quad (1)
$$
$$
\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix} \quad \text{be} \quad (3)
$$

Outline

Vectores y Documentos

- **Matriz Documento-Término:** cantidad de veces que un término (fila) aparece en un documento (columna)
- Originalmente se definió como un medio para encontrar documentos similares en la tarea de recuperación de información de documentos.
- Podemos usar vectores de documentos para encontrar otros documentos similares.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Vectores y Documentos (II)

- **Matriz Término-Documento:** cantidad de veces que un término (fila) aparece en un documento (columna)
- Las palabras similares tienen vectores similares porque tienden a ocurrir en documentos similares.

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

- Problemas:
  - Es difícil obtener resultados significativos para palabras frecuentes (como “el”, “eso”, ...)
  - ’good’ aparece con frecuencia en diferentes contextos.
- Solución:
  - **tf-idf** (frecuencia de término-frecuencia inversa de documento)

Outline

Vectores TF-IDF

- TF-IDF es una representación numérica de documentos basada en la importancia de los términos dentro de ellos.
- La Frecuencia del Término (TF) mide la frecuencia de un término en un documento.
- La Frecuencia Inversa del Documento (IDF) mide la importancia de un término en todo el corpus.
- La puntuación TF-IDF combina tanto TF como IDF para determinar la relevancia de un término en un documento.

Vectores TF-IDF (II)

$$\text{Frec. del Término (TF)}: \quad \text{TF}_{ij} = \frac{n_{ij}}{n_{\text{total}}}$$

$$\text{Frec. Inversa del Doc. (IDF)}: \quad \text{IDF}_i = \log\left(\frac{N}{n_i}\right)$$

$$\text{Puntuación TF-IDF}: \quad \text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_i$$

donde:

- n_{ij} es la frecuencia del término i en el documento j.
- n_{total} es el número total de términos en el documento j.
- N es el número total de documentos en el corpus.
- n_i es el número de documentos que contienen el término i.

Vectores TF-IDF (III)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Outline

Más allá de One-Hot: Tipos de vectores de palabras

- Basados en el conocimiento humano.
- Basados en palabras de contexto: "Debes conocer una palabra por las compañías que mantiene" (J.R. Firth, 1957).
  - Ejemplo:
    - I will go to the **cinema** on Sunday.
    - Pop-up **cinema** to enjoy films about local cuisine.
    - Concerning eyesight, photography, **cinema**, television.
  - Tipos:
    - Métodos basados en recuento (conteo de co-ocurrencias)
    - Métodos de predicción directa/aprendizaje profundo
    - Híbridos (vectores GloVe)

Outline

Vectores de palabras basados en el conocimiento humano

Basados en recursos lingüísticos creados por humanos, como WordNet, un tesauro que contiene listas de conjuntos de sinónimos e hipónimos (relaciones "es un").

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
            ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Pregunta

¿Qué problemas puedes imaginar con este enfoque?

Vectores de palabras basados en el conocimiento humano (continuación)

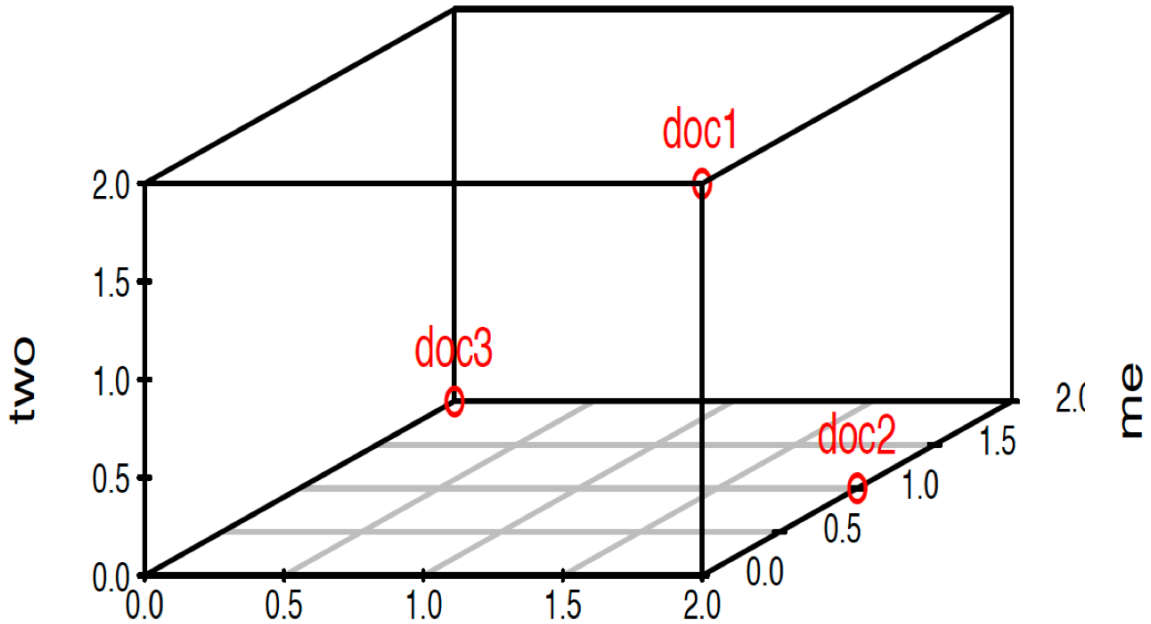
Problemas:

- No hay una forma directa de calcular la similitud entre palabras (para crear un vector de palabras).
- Falta de matices: relaciones binarias (por ejemplo, sinónimos solo en algunos contextos).
- Número limitado de palabras.
- Imposible mantenerlo actualizado.
- Subjetivo.
- Costoso en términos de trabajo humano para crear y adaptar.
- Sin embargo, puede usarse para complementar otras representaciones vectoriales.

Outline

Basado en palabras de contexto: métodos de conteo

- ¿Cómo lo hacemos? Necesitamos una colección de documentos y, utilizando estos documentos, podemos utilizar diferentes métodos...
- Comenzando por **frecuencia de término**... contando el número de palabras que aparecen en un documento.



Basado en palabras de contexto: métodos de conteo (II)

doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

	two	tea	me	you
doc1	2	2	0	0
doc2	0	2	1	1
doc3	0	0	2	2

Basado en palabras de contexto

Basado en conteo + SVD (aproximación de rango reducido)

- Contar el número de co-ocurrencias de palabras:
  1. Matriz de co-ocurrencia de palabras/ palabras basada en ventana.
  2. Información Mutua Puntual.

Word-Word Matrix

Context: ± 7 words

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

Resulting word-word matrix:

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Outline

Información Mutua Puntual (PMI)

- PMI es una medida de la asociación entre dos palabras basada en su co-ocurrencia en un corpus.
  - PMI captura en qué medida la co-ocurrencia observada de dos palabras se desvía de lo que se esperaría si fueran independientes.
  - Proporciona una medida de la fuerza y dirección de la asociación entre palabras.
  - Los valores de PMI positivos indican una asociación más fuerte de lo esperado, mientras que los valores de PMI negativos indican una asociación más débil de lo esperado.

$$\text{PMI}(w_1, w_2) = \log\left(\frac{P(w_1, w_2)}{P(w_1)P(w_2)}\right)$$

- $P(w_1, w_2)$ es la probabilidad conjunta de que las palabras w_1 y w_2 co-ocurran juntas.
- $P(w_1)$ y $P(w_2)$ son las probabilidades individuales de que las palabras w_1 y w_2 ocurran de forma independiente.

Información Mutua Puntual Positiva (PPMI)

- PPMI es una versión modificada de PMI que aborda algunas de sus limitaciones, especialmente el manejo de eventos de baja frecuencia y el problema de los valores negativos.
  - PPMI solo considera valores positivos y asigna pesos más altos a las co-ocurrencias más significativas.
  - PPMI mide la fuerza de la asociación entre dos palabras en función de sus probabilidades de co-ocurrencia en un corpus.

$$\text{PPMI}(w_1, w_2) = \max(\log(\frac{\text{cooc}(w_1, w_2) \cdot N}{\text{freq}(w_1) \cdot \text{freq}(w_2)}), 0)$$

- $\text{cooc}(w_1, w_2)$ es el recuento de co-ocurrencia de las palabras w_1 y w_2 en una matriz de co-ocurrencia.
- $\text{freq}(w_1)$ y $\text{freq}(w_2)$ son las frecuencias de las palabras w_1 y w_2 en el corpus.
- N es el número total de co-ocurrencias en la matriz.

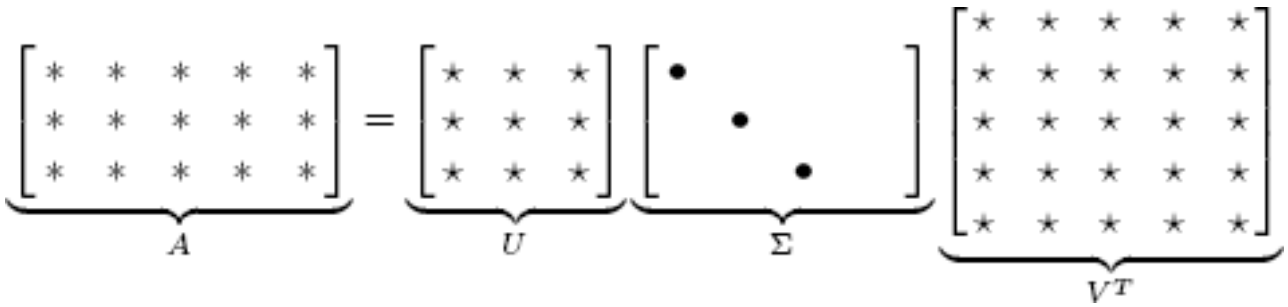
PPMI: Ejemplo

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

Descomposición de Valor Singular (Singular Value Decomposition, SVD)

Basado en conteo + SVD

- Contar la co-ocurrencia de palabras: dos opciones
  - Matriz de co-ocurrencia de palabras/documentos
  - Matriz de co-ocurrencia de palabras en ventana
- Descomposición de Valor Singular (SVD) $X = USV^T$ para reducir la dimensionalidad (rango). Las filas de U son las representaciones de palabras.



Descomposición de Valor Singular (Singular Value Decomposition, SVD) (II)

Problemas:

- Las palabras funcionales (the, you, is, ...) tienen un gran impacto.
- Soluciones: modificar los recuentos brutos (log tf-idf) o eliminar las palabras funcionales.
- Matriz de alta dimensionalidad.
- Costo cuadrático de SVD.
- Soluciones: algoritmos adaptativos.

Basado en palabras de contexto: Predicción directa

- Representaciones continuas en el espacio o incrustaciones de palabras.
- Pequeño vector de números reales (dimensión 200-400).
- La similitud lingüística o semántica se puede medir con la distancia euclidiana o la similitud del coseno.
- Las diferencias de vectores capturan las relaciones entre palabras.
- Elección estándar para modelos de aprendizaje profundo.

```
(12424, 100)
```

	0	1	2	3	4	5	6	7	8	9	...	90	91	92	93
shall	-0.002272	0.015870	0.018349	0.022802	0.028364	-0.040064	-0.013263	0.136607	0.019667	0.033407	...	0.037663	-0.087140	0.073169	-0.028257
unto	0.034425	-0.102070	0.018051	0.017960	0.172954	-0.115672	-0.012632	0.096919	-0.049203	-0.040344	...	0.106373	-0.075703	0.013888	-0.134224
lord	0.051990	-0.113865	0.007226	0.031754	0.052963	-0.094523	-0.067664	0.001706	-0.112827	-0.078586	...	-0.041636	0.053685	0.041299	-0.026255
thou	-0.152183	-0.073681	-0.091472	0.022033	0.008415	-0.048438	-0.041181	0.082019	0.004648	0.044870	...	0.101531	-0.018404	-0.070462	-0.041363
thy	-0.257579	-0.023008	0.053303	0.013690	-0.083293	0.034279	0.078811	0.079851	-0.015215	-0.111211	...	-0.064527	0.112085	0.061625	0.026398

5 rows × 100 columns

Outline

Word2Vec: CBOW

- Predicción directa / Métodos de aprendizaje profundo: Word2vec (Mikolov, Google 2013)
  - **Continuous Bag-of-Words (CBOW)**: predicción de una palabra utilizando las palabras de contexto (bag-of-words).

Word2Vec: CBOW (II)

FUN WITH FILL-INS

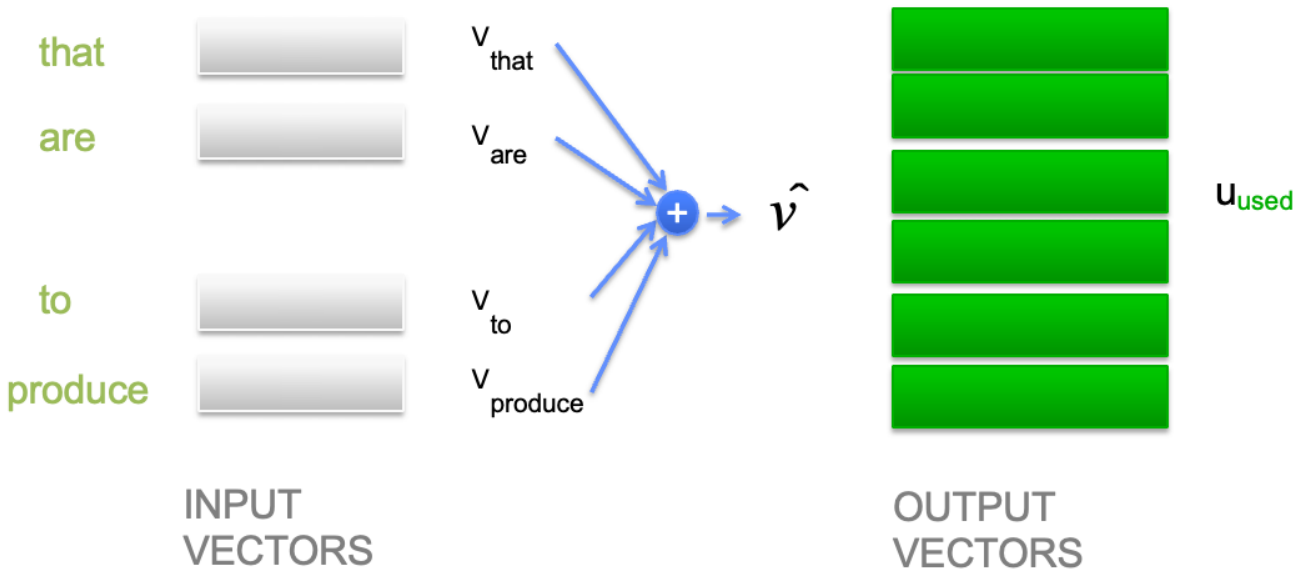
First Grade Sight Words

Choose the sight word from the Word List that will complete each sentence below.
Hint: Words can be used more than once.

Word List: are, good, now

1. Plums _____ in a tree.
2. Are the plums _____ now?
3. The plums are hard. They _____ not good.
4. Sun is good for plums. Rain is _____ for plums.
5. Are the plums good _____?
6. The plums _____ soft.
7. _____ the plums are good!

Word2Vec: CBOW (III)



CBOW

is a group of related models that are used to produce word embeddings

Ecuaciones de CBOW

- Continuous Bag-of-Words (CBOW)
- W es el vocabulario de palabras.
- Vectores de entrada: v_w para cada $w \in W$.
- Vectores de salida: u_w para cada $w \in W$.

El vector de salida 'predicho' es la suma de todos los vectores de entrada de contexto:

$$u_w = \sum_{\text{palabras de contexto}} v_w$$

Utilizamos el producto escalar para calcular el score (similitud de palabras):

$$\text{score}(w) = s_{w,c} = u_w \cdot v_c$$

Y la función softmax para obtener las probabilidades:

$$p(w|c) = \frac{e^{s_{w,c}}}{\sum_{w' \in W} e^{s_{w',c}}}$$

Ecuaciones de CBOW (II)

La elección estándar para la función de loss es la entropía cruzada de la probabilidad estimada $p(w)$ respecto a la probabilidad real $q(w)$:

$$\begin{aligned} \mathrm{CE}(q,p) &= E_q[-\log p(w)] \\ &= E_q[-\log p(w) + \log q(w) - \log q(w)] \\ &= E_q[\log p(w)] + E_q[-\log q(w)] \\ &= D_{KL}(q||p) + H(q) \end{aligned}$$

En nuestro caso, es equivalente a minimizar el logaritmo negativo de la probabilidad del vector de palabras objetivo dado el contexto:

$$\text{minimizar} \ -\log p(w_c|w_{\text{contexto}})$$

Outline

Word2Vec: Skip-gram

- Predicción directa / Métodos de aprendizaje profundo: Word2vec (Mikolov, Google 2013)
  - **Arquitectura de skip-gram continuo**: predicción de las palabras de contexto utilizando la palabra actual.

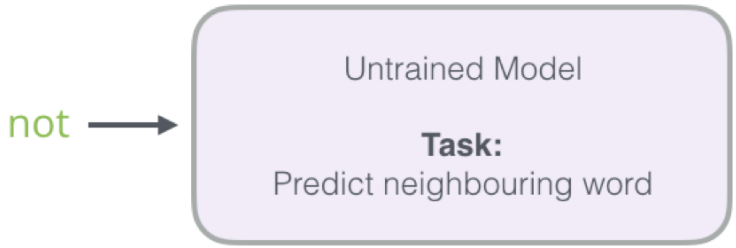
Pasos detallados: entrenamiento de skip-gram con muestreo negativo

- Echemos un vistazo a cómo lo utilizamos para entrenar un modelo básico que predice si dos palabras aparecen juntas en el mismo contexto.

Pasos preliminares

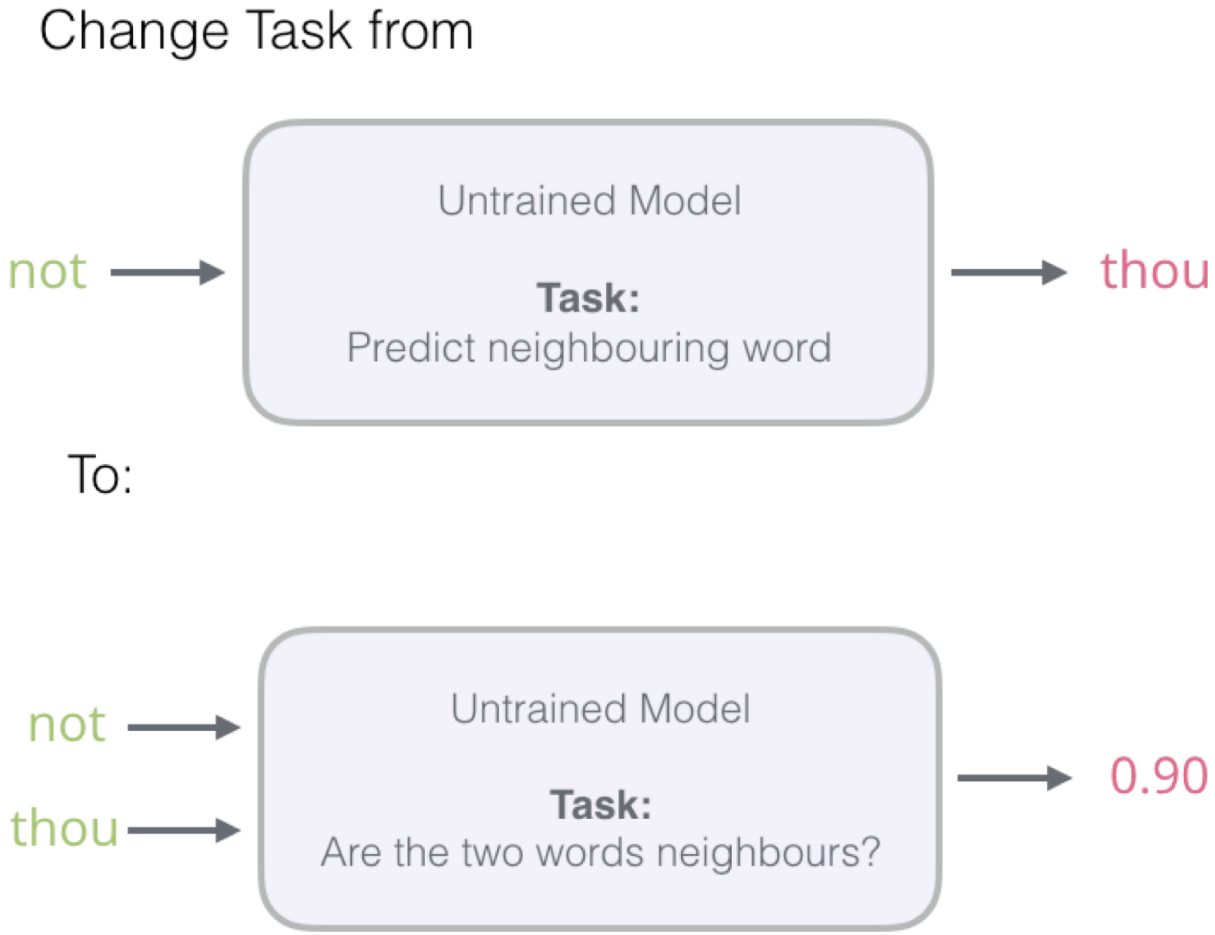
- Comenzamos con la primera muestra de nuestro conjunto de datos.

input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine
a	not
a	make
a	machine
a	in
machine	make
machine	a
machine	in
machine	the
in	a
in	machine
in	the
in	likeness



Nota sobre la eficiencia del muestreo negativo

- Tomamos la característica y se la proporcionamos al modelo no entrenado, pidiéndole que prediga si las palabras están en el mismo contexto o no (1 o 0).



Ejemplos negativos

- Ahora esto se puede calcular a gran velocidad, procesando millones de ejemplos en minutos. Pero hay una falla que debemos solucionar. Si todos nuestros ejemplos son positivos (objetivo: 1), nos exponemos a la posibilidad de un modelo astuto que siempre devuelva 1, logrando una precisión del 100

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

Ejemplos negativos (II)

- Para cada muestra en nuestro conjunto de datos, agregamos ejemplos negativos. Estos tienen la misma palabra de entrada y una etiqueta 0.

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

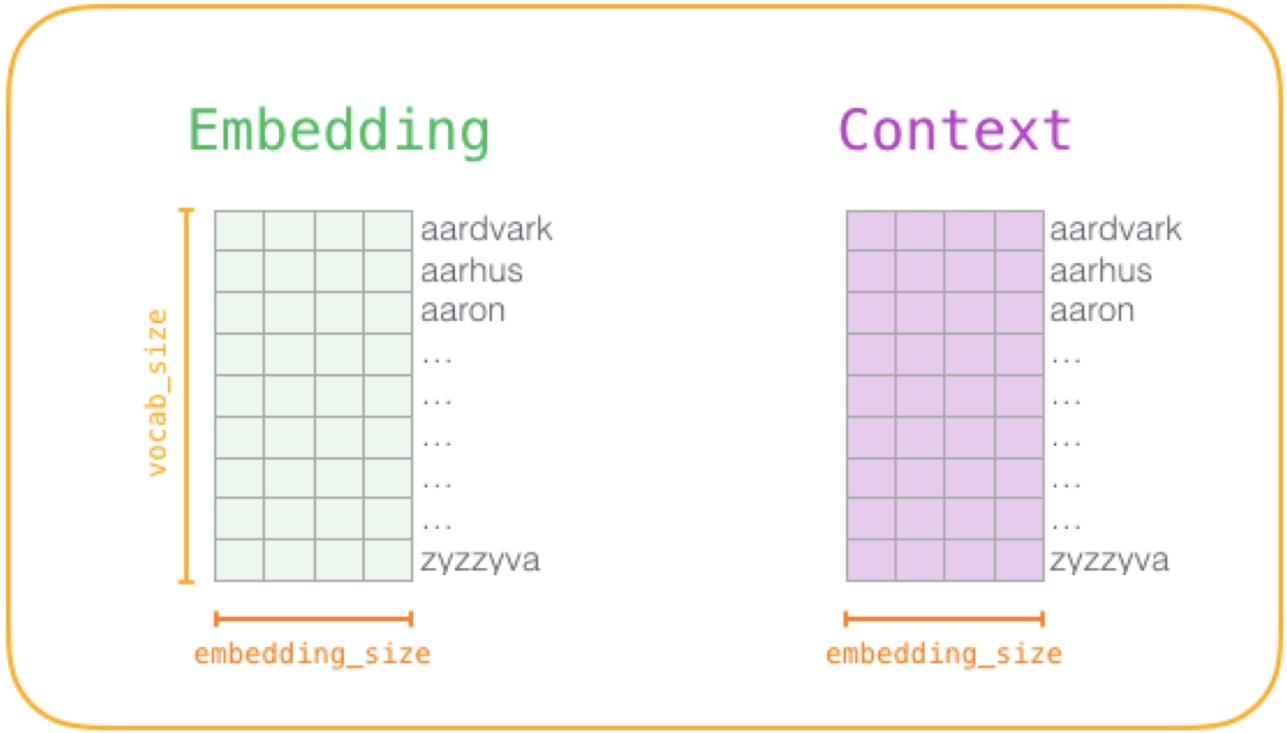
Negative examples

- Estamos contrastando la señal real (ejemplos positivos de palabras vecinas) con el ruido (palabras seleccionadas aleatoriamente que no son vecinas). Esto conduce a un gran equilibrio entre eficiencia computacional y estadística.

Proceso de entrenamiento

- Ahora que hemos establecido las dos ideas centrales de skip-gram y muestreo negativo, podemos examinar más de cerca el proceso de entrenamiento real de Word2Vec.
- Antes de que comience el proceso de entrenamiento, procesamos el texto con el que entrenamos el modelo. En este paso, determinamos el tamaño de nuestro vocabulario (lo llamaremos vocab_size) y qué palabras pertenecen a él.
- Al comienzo de la fase de entrenamiento, creamos dos matrices: una matriz de embeddings y una matriz de contexto. Estas dos matrices tienen un embedding para cada palabra de nuestro vocabulario (entonces vocab_size es una de sus dimensiones). La segunda dimensión es la longitud que queremos que tenga cada embedding (embedding_size: 300 es un valor común).

Proceso de entrenamiento



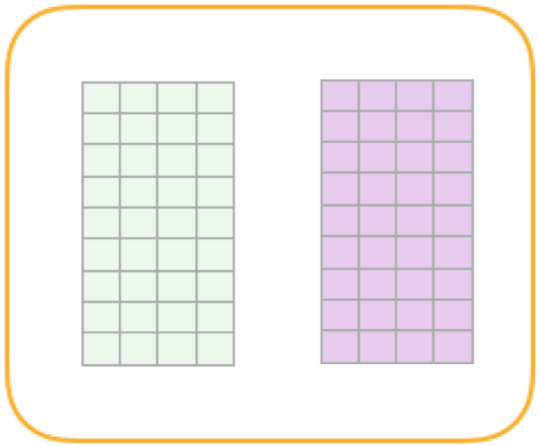
Proceso de entrenamiento: Paso a paso

1. Al comienzo del proceso de entrenamiento, inicializamos estas matrices con valores aleatorios. Luego, comenzamos el proceso de entrenamiento. En cada paso de entrenamiento, tomamos un ejemplo positivo y sus ejemplos negativos asociados. Veamos nuestro primer grupo:

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
...	...	...

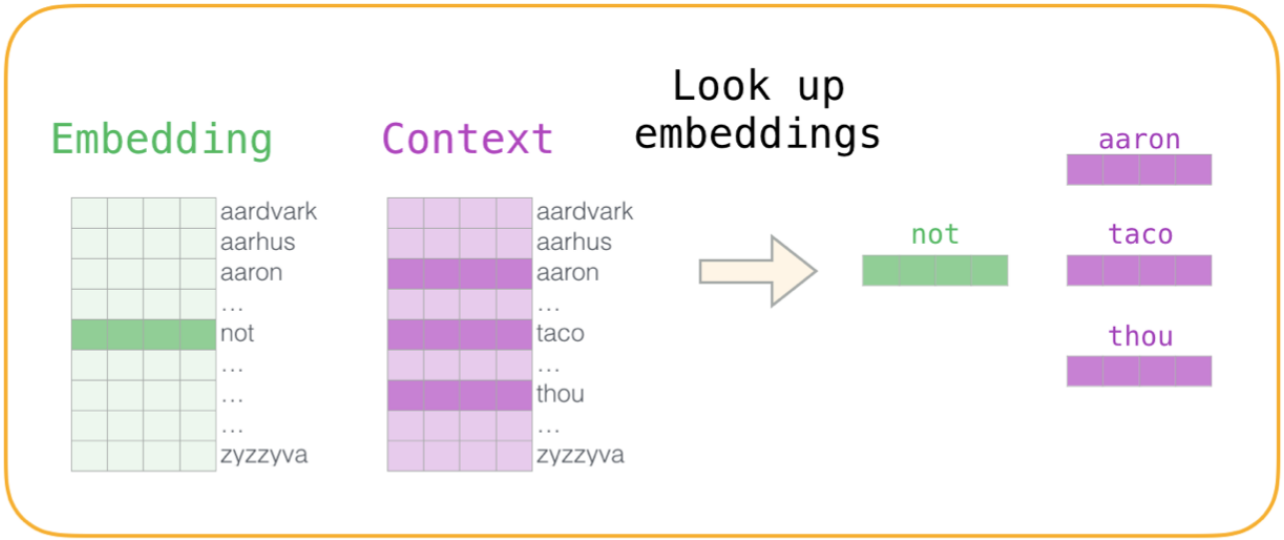
model



Proceso de entrenamiento: Paso a paso (II)

- Ahora tenemos cuatro palabras: la palabra de entrada "not" y las palabras de salida/contexto: "thou" (el vecino real), "aaron" y "taco" (los ejemplos negativos).

2. Procedemos a buscar sus embeddings: para la palabra de entrada, buscamos en la matriz de embeddings. Para las palabras de contexto, buscamos en la matriz de contexto (aunque ambas matrices tienen un embedding para cada palabra de nuestro vocabulario).



Proceso de entrenamiento: Paso a paso (III)

3. Luego, realizamos el producto punto del embedding de entrada con cada uno de los embeddings de contexto. En cada caso, eso daría como resultado un número que indica la similitud de los embeddings de entrada y contexto.
4. Ahora necesitamos una forma de convertir estos puntajes en algo que se parezca a probabilidades: necesitamos que todos sean positivos y tengan valores entre cero y uno. Esta es una gran tarea para la sigmoidal, la operación logística. Y ahora podemos tratar la salida de las operaciones sigmoidales como la salida del modelo para estos ejemplos.

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

Proceso de entrenamiento: Paso a paso (IV)

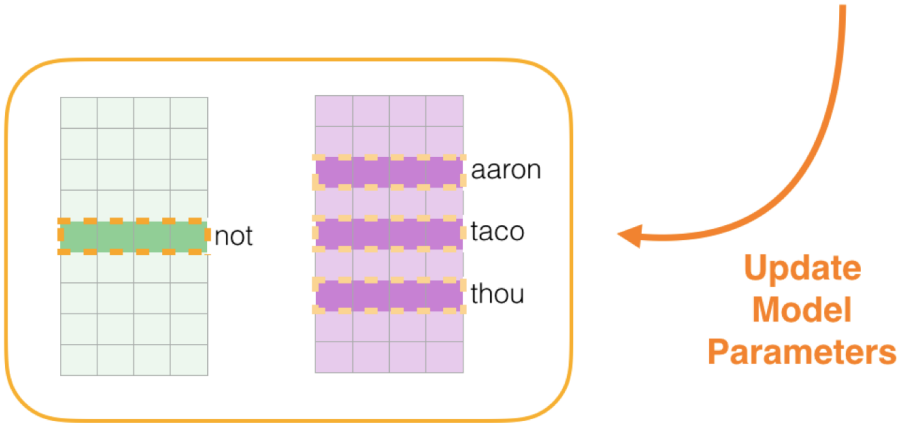
5 Ahora que el modelo no entrenado ha realizado una predicción y dado que tenemos una etiqueta objetivo real para comparar, calculemos cuánto error hay en la predicción del modelo. Para hacer eso, simplemente restamos los puntajes sigmoidales de las etiquetas objetivo.

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

Proceso de entrenamiento: Paso a paso (V)

6 Aquí viene la parte de "aprendizaje" de "aprendizaje automático". Ahora podemos usar este puntaje de error para ajustar los embeddings de "not", "thou", "aaron" y "taco" para que la próxima vez que hagamos este cálculo, el resultado esté más cerca de los puntajes objetivo.

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



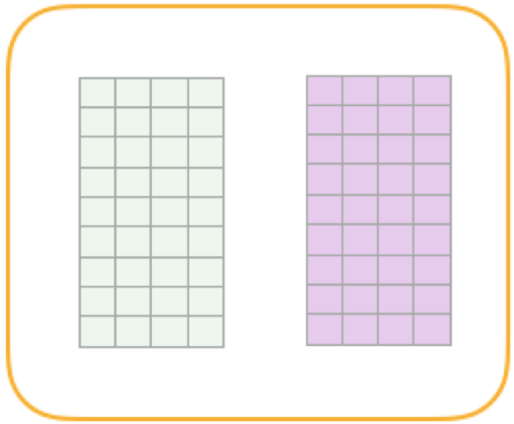
Proceso de entrenamiento: Paso a paso (VI)

7. Esto concluye el paso de entrenamiento. Salimos de él con embeddings ligeramente mejores para las palabras involucradas en este paso ("not", "thou", "aaron" y "taco"). Ahora procedemos a nuestro próximo paso (el siguiente ejemplo positivo y sus ejemplos negativos asociados) y realizamos el mismo proceso nuevamente.

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



Proceso de entrenamiento: Paso a paso (VII)

- Los embeddings continúan mejorando mientras recorremos todo nuestro conjunto de datos varias veces. Luego, podemos detener el proceso de entrenamiento, descartar la matriz de contexto y usar la matriz de embeddings como nuestros embeddings pre-entrenados para la siguiente tarea.

Outline

Otras unidades de lenguaje

- Frase: Washington_Post es un periódico.
  - Las frases se pueden generar automáticamente en base a recuentos, por ejemplo:

$$\frac{\text{count}(w_i, w_j) - 6}{\text{count}(w_i) \rightarrow \text{count}(w_j)}$$

- Carácter: W a s h i n g t o n _ P o s t _ e s _ u n _ p e r i ó d i c o
  - Crear una representación de palabras a partir de sus caracteres
  - Modelos completamente a nivel de caracteres
- Sub-palabra: Wash #ing #ton Post es un #periódico
  - N-gramas, Codificación de Pares de Bytes (BPE), Wordpiece, Sentencepiece

Modelo de sub-palabras: fastText

fastText (Facebook, 2016)

- Arquitectura de skip-gram basada en sub-palabras: la representación vectorial de una palabra es la suma de los embeddings de los n-gramas de caracteres de la palabra actual ($3 \leq n \leq 6$ por defecto).

Ej: la representación fastText de la palabra 'where' es la suma de los embeddings de 15 sub-palabras (n-gramas):

- 3-gramas: <wh, whe, her, ere, re>
- 4-gramas: <whe, wher, here, ere>
- 5-gramas: <wher, where, her>
- 6-gramas: <where, where>
- + la propia palabra: <where>

Mejoras de fastText sobre word2vec

- **Modelado de sub-palabras**: fastText utiliza una arquitectura de skip-gram basada en sub-palabras. La representación vectorial de una palabra es la suma de los embeddings de n-gramas de caracteres. Esto permite a fastText capturar información morfológica y soportar palabras fuera del vocabulario (OOV)
- **Selección flexible de n-gramas**: fastText permite personalizar el rango de n-gramas de caracteres considerados durante el entrenamiento para ajustarlo según las características del idioma o la tarea.

Mejoras de fastText sobre word2vec (II)

- **Función de hash**: fastText utiliza una función de hash para reducir el uso de memoria. En lugar de almacenar explícitamente todos los n-gramas posibles, fastText aplica un truco de hash para mapear los n-gramas en un espacio de hash de tamaño fijo. La función de hash se define de la siguiente manera:

$$\text{hash\_function}(\text{n-gram}) = \text{hash}(\text{n-gram}) \mod B$$

Donde B es tamaño del bucket

Modelo híbrido: GloVe

GloVe: Vectores Globales para la Representación de Palabras

- Híbrido: recuentos de co-ocurrencia + predicción
- Las proporciones de las probabilidades de co-ocurrencia palabra-palabra tienen el potencial de codificar alguna forma de significado.
- El modelo GloVe se entrena en las entradas no nulas de una matriz de co-ocurrencia palabra-palabra global, que tabula con qué frecuencia las palabras co-ocurren entre sí en un corpus dado.
- El objetivo del entrenamiento es aprender vectores de palabras de manera que el producto escalar sea igual al logaritmo de la probabilidad de co-ocurrencia de las palabras (la relación es igual a la diferencia de los logaritmos).

Modelo híbrido: GloVe (II)

- Probabilidad de co-ocurrencia:

$$P_{ij} = \frac{X_{ij}}{X_i}$$

- Producto escalar de vectores de palabras:

$$\mathbf{v}_i \cdot \mathbf{v}_j = \log(P_{ij})$$

- Proporción de probabilidades de co-ocurrencia:

$$\frac{P_{ij}}{P_{ik}} = \frac{\exp(\mathbf{v}_i \cdot \mathbf{v}_j)}{\exp(\mathbf{v}_i \cdot \mathbf{v}_k)}$$

- Diferencia de logaritmos:

$$\mathbf{v}_i \cdot \mathbf{v}_j - \mathbf{v}_i \cdot \mathbf{v}_k = \log(P_{ij}) - \log(P_{ik})$$

GloVe: Algoritmo de entrenamiento

1. Inicializar los vectores de palabras $\mathbf{v}_i$ y los sesgos b_i.
2. Calcular la proporción de probabilidades de co-ocurrencia para cada par de palabras: $\frac{P_{ij}}{P_{ik}}$.
3. Definir la función de loss: $J = \sum_{i,j} f\left(P_{ij}\right)\left(\mathbf{v}_i \cdot \mathbf{v}_j - \log(P_{ij})\right)^2$.
4. Actualizar los vectores de palabras y los sesgos utilizando descenso de gradiente para minimizar la función de loss.
5. Repetir los pasos 2-4 hasta converger.

Outline

Ejemplo

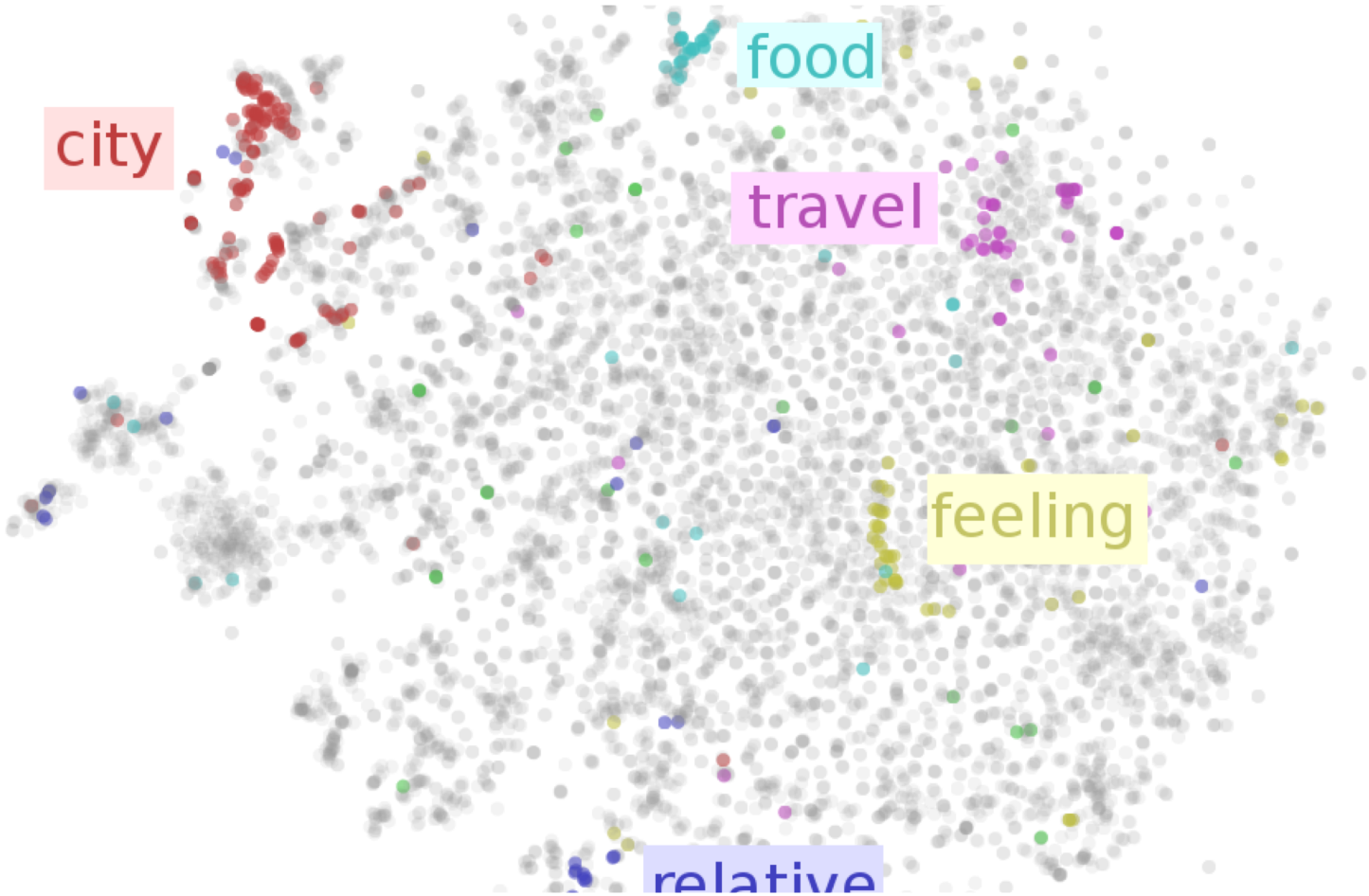
Palabras más cercanas a la palabra objetivo 'frog' (rana):

- frogs (ranas)
- toad (sapo)
- litoria
- leptodactylidae
- lizard (lagarto)
- eleutherodactylus

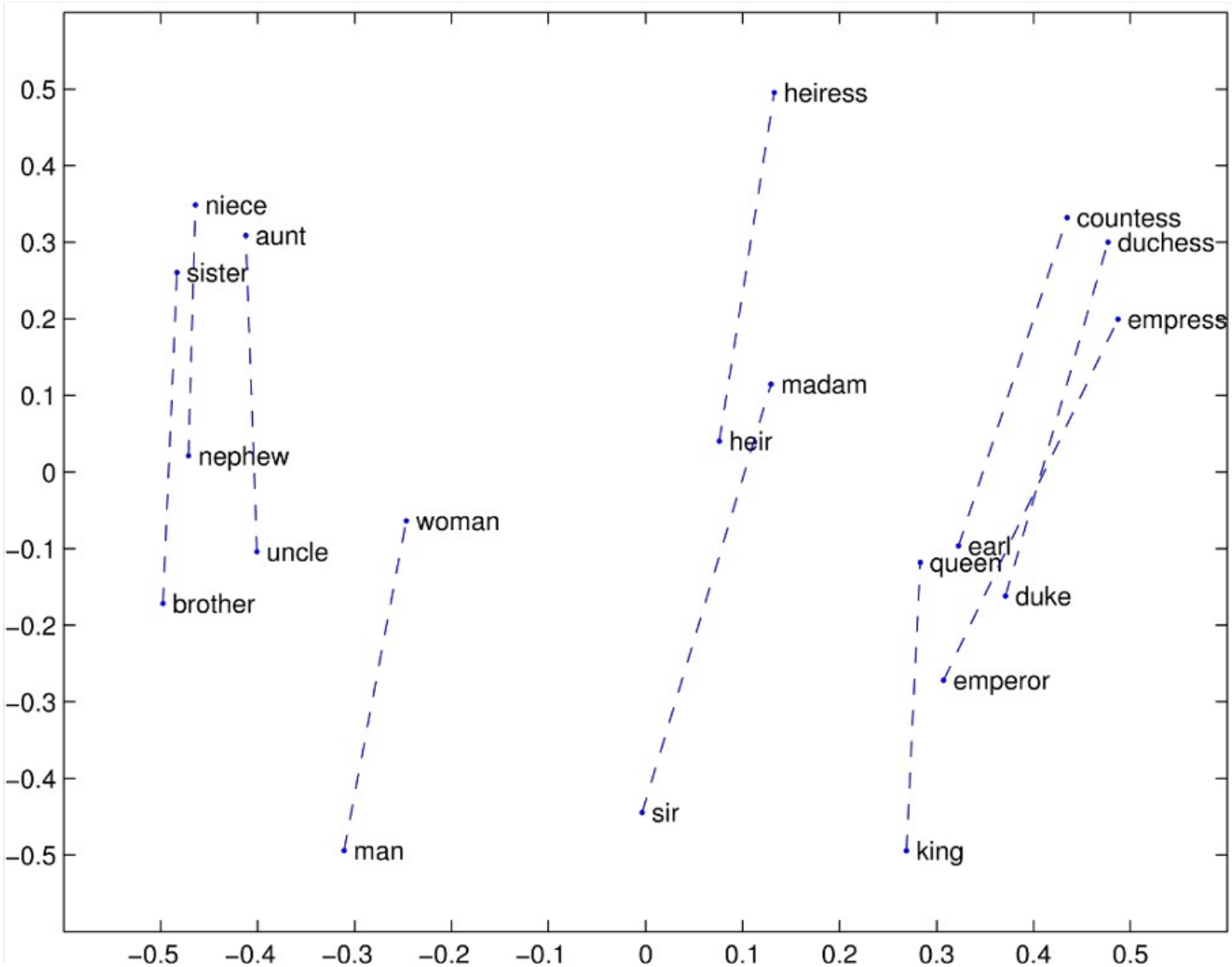
Traducciones de 'frog' (incrustaciones de palabras alineadas):

- 'rana', 'granota'
- 'ranas', 'granotes'
- 'sapo', 'gripau'
- 'litoria', 'litòria'

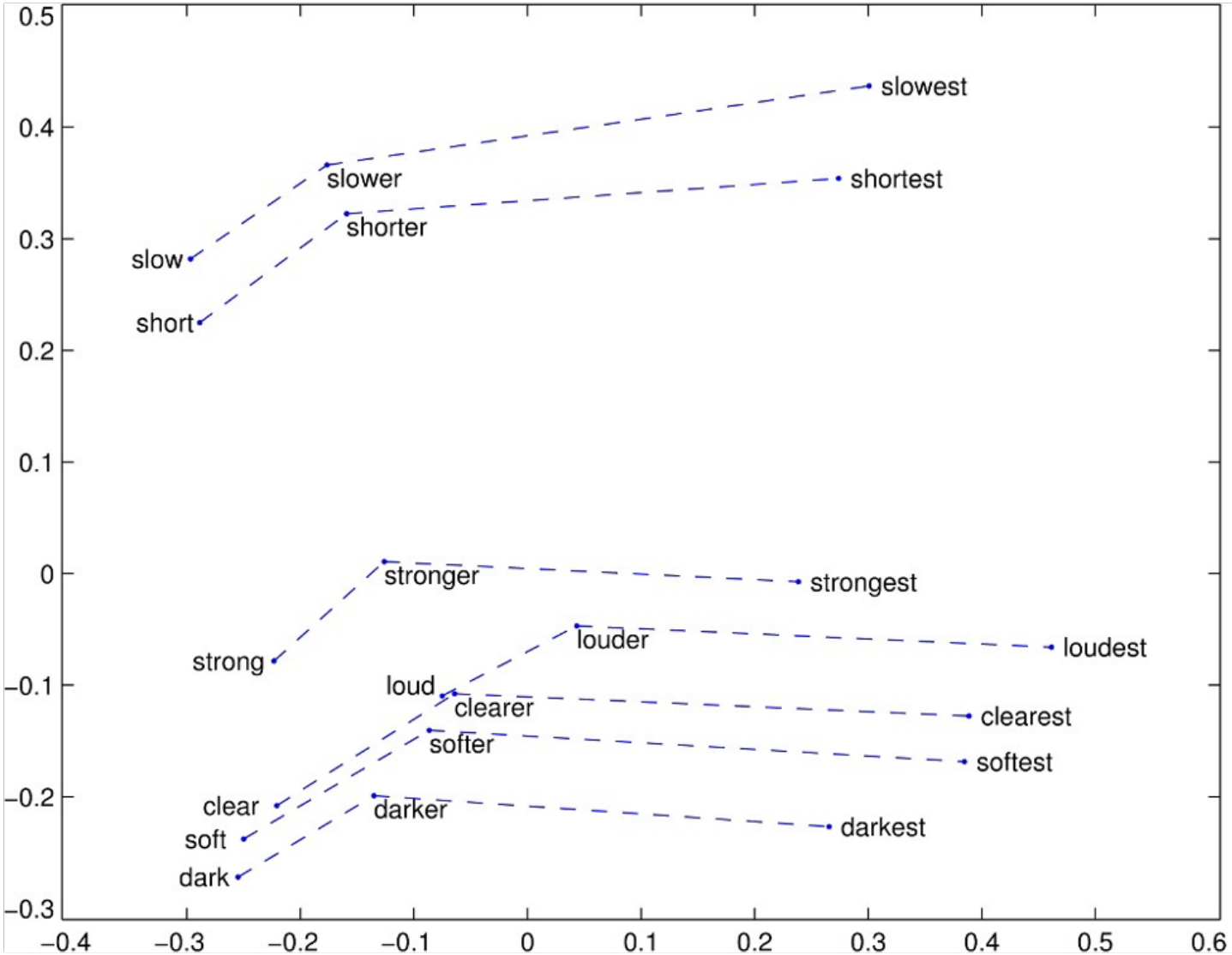
Visualización de Representaciones



Ejemplo: Estructuras lineales hombre-mujer



Ejemplo: Estructuras lineales comparativo - superlativo



Ejemplo: Vectores de palabras catalanes (CBOW)

- 'dimecres' + ('dimarts' - 'dilluns') = 'dijous'
- 'tres' + ('dos' - 'un') = 'quatre'
- 'tres' + ('2' - 'dos') = '3'
- 'viu' + ('coneixia' - 'coneix') = 'vivia'
- 'la' + ('els' - 'el') = 'les'
- 'Polònia' + ('francès' - 'França') = 'polonès'

Pregunta

¿Cómo podemos evaluar los vectores de palabras?

Evaluación

Evaluación intrínseca vs. extrínseca:

- Evaluación intrínseca: evaluar los vectores de palabras en función de su similitud, analogía y distancia.
- Evaluación extrínseca: evaluar los vectores de palabras en el contexto de tareas secund
arias como la traducción y el análisis de sentimientos.

Los métodos de evaluación intrínseca incluyen:

- Similitud de palabras: encontrar la palabra más cercana a una palabra objetivo.
- Analogía de palabras: encontrar una palabra que complete una analogía (por ejemplo, "a es a b como c es a...").
- Distancia: medir la similitud utilizando la similitud del coseno, la distancia euclidiana o el producto escalar.

Desafíos de los Vectores de Palabras

Los desafíos de los vectores de palabras en redes neuronales para el procesamiento del lenguaje incluyen:

- Evaluar adecuadamente los vectores de palabras en cuanto a similitud, analogía y distancia.
- Manejar conjuntos de datos grandes con millones o miles de millones de palabras.
- Asegurar que las operaciones matemáticas codifiquen el significado en los vectores de palabras.
- Capturar el significado de una palabra en función de su contexto y coocurrencia.

Resumen

Incrustación de Palabras con Significado

Cualquier técnica que mapea una palabra (o frase) desde su espacio de entrada original de alta dimensionalidad (el conjunto de todas las palabras) a un espacio vectorial numérico de dimensionalidad más baja, es decir, se incrusta la palabra en un espacio diferente.

Importancia de la Incrustación de Palabras

Las representaciones de palabras son un componente crítico de muchos sistemas de procesamiento del lenguaje natural.

Resumen: Mensaje clave

- La similitud en el significado se refleja en la similitud de los vectores. Las matemáticas deben ser capaces de codificar el significado.
- "Conocerás una palabra por las compañías que mantiene" - el entorno de una palabra le da significado.
- ¡Usar grandes conjuntos de datos, especialmente los modelos neuronales, requiere mucha información!