Procesamiento del lenguaje natural
Enfoques basados en word embeddings

WORD-EMBEDDINGS

Word embeddings

- Las word-embeddings son representaciones de palabras en espacios de varias dimensiones que permiten superar algunas de las limitaciones de las representaciones de palabras que hemos visto, por ejemplo:
  - Reflejando similitudes semánticas entre palabras
  - Plasmando relaciones complejas entre palabras

- En este caso cada palabra del diccionario es representada mediante un vector de varias dimensiones (decenas o centenas) que permiten capturar su relación con otras palabras

- Existen distintas técnicas para estimar word-embeddings a partir de un corpus de documentos y fijando el número de dimensiones que queremos usar
  - El resultado es una matriz de tantas columnas como palabras tenga el diccionario y tantas filas como dimensiones hayamos determinado

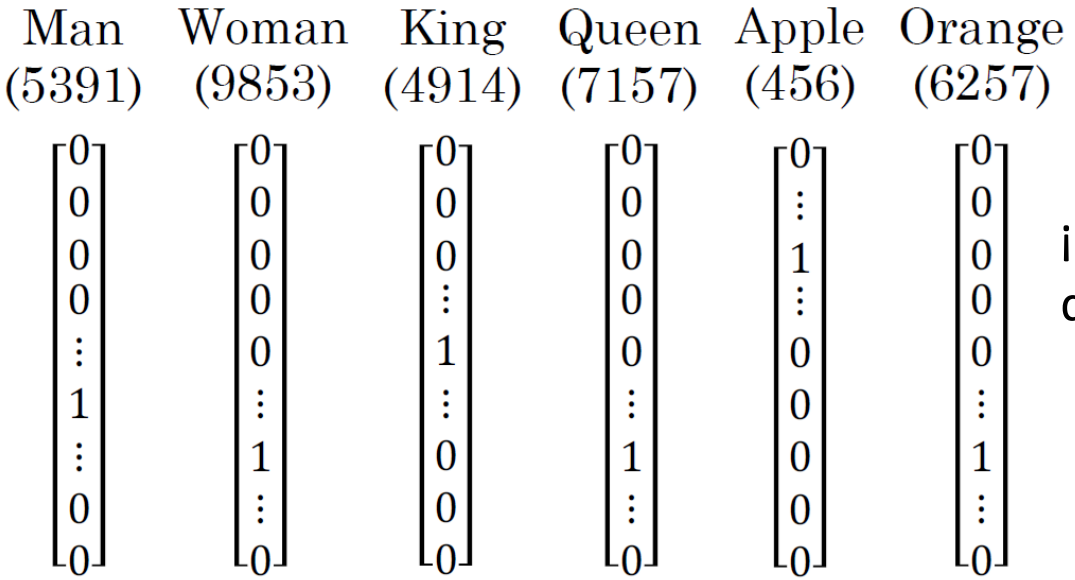
Representando un diccionario de palabras

La manera naïve de representar las palabras de un diccionario o corpus es un vector binario (one-hot encoding)

- Tantas dimensiones como palabras haya en el diccionario

$$V = [a, aaron, \ldots, zulu, <UNK>] \quad \sim 10.000 \text{ palabras}$$

- Todos valores a cero, menos el que representa la palabra que toma el valor 1
  - Es una representación dispersa



El número indica la posición de la palabra en el vector binario que representa todas las palabras del diccionario

¡No hay nada en esta representación que modele las relaciones existentes entre las palabras!

La similitud del coseno es cero entre todos ellos

Ejemplo tomado DeepLearning.ai

Representando palabras mediante un vector de características

- En lugar de la representación binaria, es mejor usar un conjunto de dimensiones que representen “conceptos” sobre los que situar las palabras del diccionario

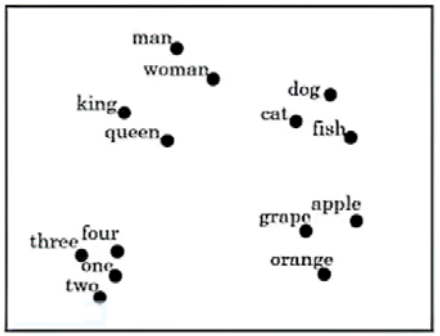
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
...	...	...	...	...	...	...

- El número de dimensiones es mucho menor que el número de palabras
- Los vectores de palabras resultantes son densos (o no tan dispersos) porque toman valores diferentes de cero para muchas de sus dimensiones.
  - Aunque para otras tomará valores muy cercanos a cero

Ejemplo tomado DeepLearning.ai

Palabras en un espacio n-dimensional

- Si mostramos las palabras en un espacio **denso** donde las dimensiones tienen un **significado**, la posición de las palabras en dicho espacio nos indicará mucho sobre ellas. Por ejemplo
  - Las palabras "*red*", "*blue*", "*yellow*", etc, deberían estar relativamente cercanas, ya que suelen jugar un papel parecido en las frases
  - La palabra "*tomatoe*" debería estar más cerca de la palabra "*red*" que, por ejemplo, de "*blue*" ya que están asociadas con más frecuencia
  - Sería posible inferir que "*man*" es a "*woman*", lo que "*king*" a "*queen*" si el espacio n-dimensional recoge las relaciones de género entre palabras
- Como ese espacio tiene muchas dimensiones, no se puede visualizar tal cual
  - Se podría mostrar usando técnicas de "reducción de la dimensionalidad" como análisis de componentes principales o, más sofisticadas, como t-SNE
  - Pero en la representación 2D resultante se habrán perdido matices importantes



Ejemplo tomado DeepLearning.ai

Analogías sobre word-embeddings

"Man" es a "Woman" como "King" es a…

e_i es el vector columna i-esimo de la matriz word-embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
…	…	…	…	…	…	…

"Man" es a "Woman" ~ $e_{5391} - e_{9853} \approx [-2\ 0\ 0\ 0\ \ldots]$

"King" es a "Queen" ~ $e_{4914} - e_{7157} \approx [-2\ 0\ 0\ 0\ \ldots]$

Ejemplo tomado DeepLearning.ai

Analogías sobre word-embeddings

"Man" es a "Woman" como "King" es a…

e_i es el vector columna i-esimo de la matriz word-embedding

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royalness	0.01	0.02	0.93	0.95	-0.01	0.00
Edad	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.05	0.01	0.02	0.01	0.95	0.97
…	…	…	…	…	…	…

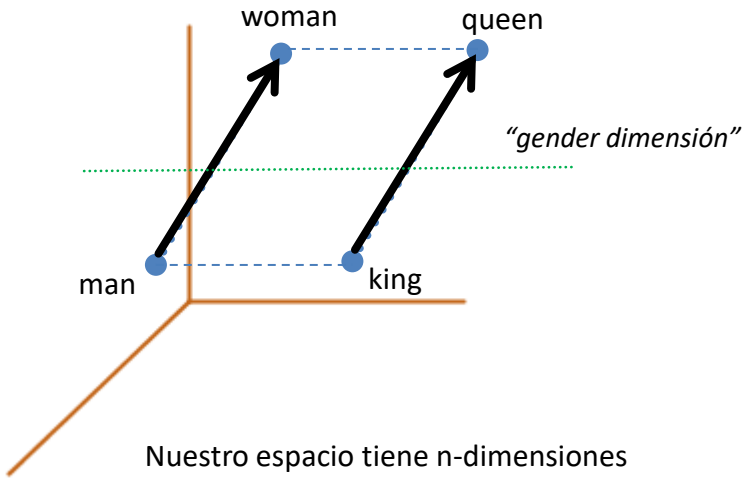
Formulación del problema

$$e_{man} - e_{woman} \approx e_{king} - e_w$$

Encontrar la palabra w tal que

$$\arg\max_w (sim(e_w, e_{king} - e_{man} + e_{woman})$$

Donde $sim(\cdot,\cdot)$ es una medida de semejanza adecuada como la similitud del coseno

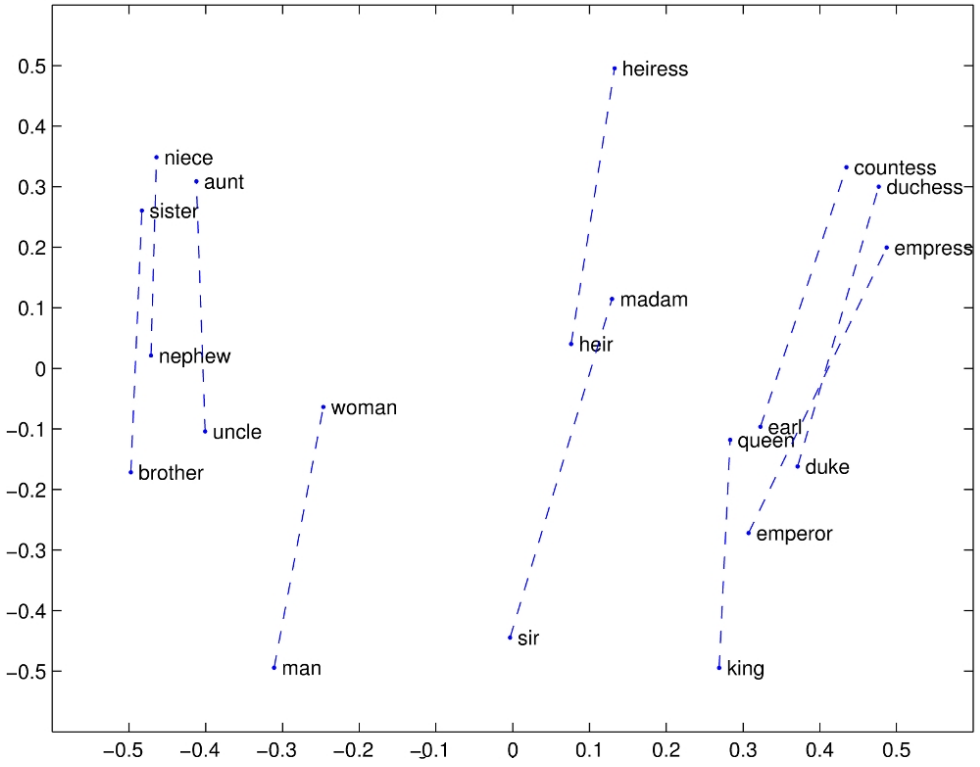


Nuestro espacio tiene n-dimensiones (aprox. decenas o cientos)

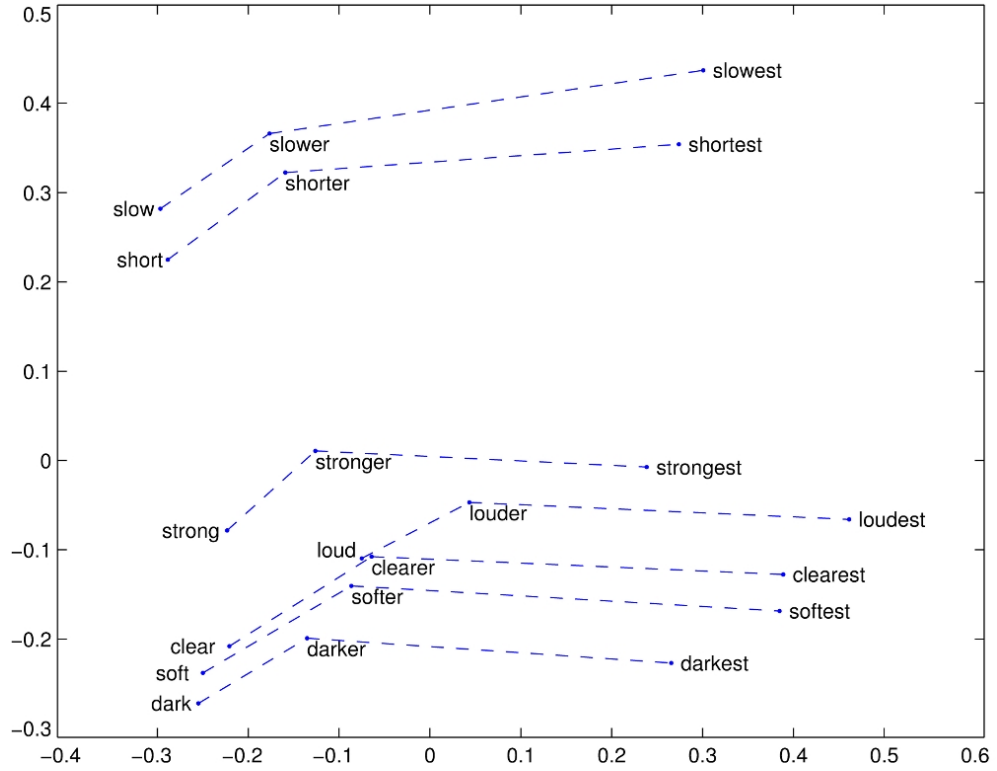
Ejemplo tomado DeepLearning.ai

Palabras en un espacio n-dimensional

La diferencia de vectores muestra lo que las diferencia y esta diferencia puede contener conceptos interesantes



Relación de género



Relación de comparativo y superlativo

Ejemplo tomado de la web de GloVe de U. de Stanford

Cómo se calculan los Word-embeddings

- Los vectores se calculan a partir de un corpus de documentos usando distintas aproximaciones (típicamente redes neuronales profundas)
- Hay que fijar de antemano el número de dimensiones que tendrá el espacio en el que vamos a representar las palabras
  - Sin embargo, el “significado” de esas dimensiones no se puede elegir
- La técnica que calcula los word-embeddings usa el conjunto de datos para fijar el “significado” de las dimensiones y qué valor toman las palabras en cada uno de ellas
  - Aprende cómo distribuir las palabras y cómo configurar las dimensiones para representar bien los ejemplos del conjunto de datos
  - El espacio resultante reflejará **de forma conjunta** relaciones como el color o el género, etc
    - Eso no quiere decir que una dimensión sea explícitamente el género, otra el color, etc
    - De hecho, las dimensiones resultantes no son interpretables, aunque sería posible encontrar “combinaciones” de las dimensiones que representen conceptos inteligibles

Aproximaciones para calcular word-embeddings

- Word2Vec (Mikolov et al.2013)
  - https://code.google.com/archive/p/word2vec/

  - Se basa en la tarea de predecir cual es la palabra que falta en una frase.
    - Basado en el concepto de modelos de lenguaje.

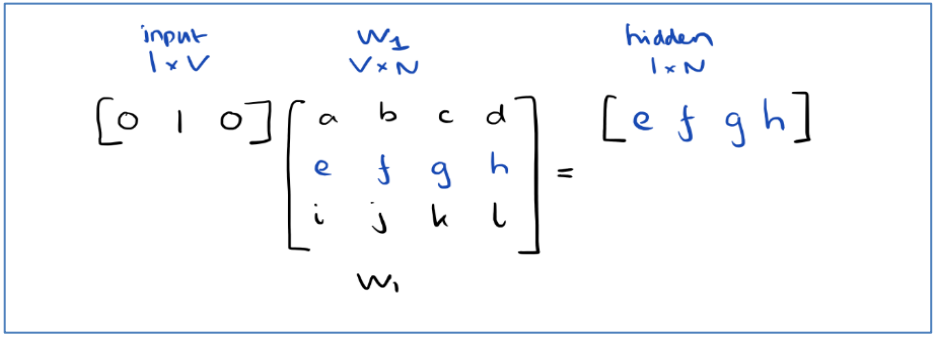
  - Hay 2 enfoques
    - CBOW: dado un contexto de palabras, predecir cual es la palabra que iría en medio.
      - … people who keep pet dogs or **cats** exhibit better mental and physical health …

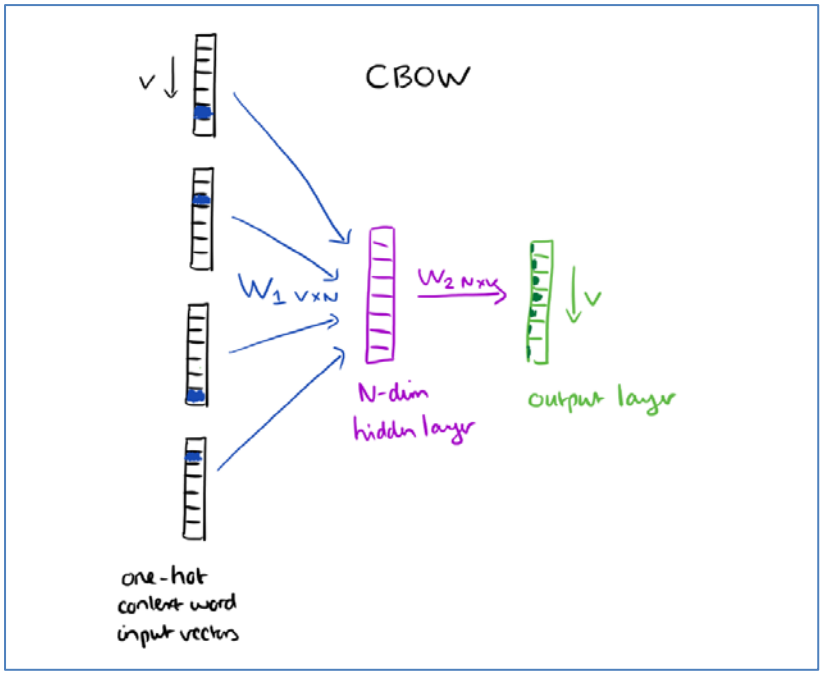
    - SKIP-GRAM: dada la palabra que va en el medio predecir qué palabras estarán en su contexto
      - **… people who keep pet dogs or** cats **exhibit better mental and physical health …**

Aproximaciones para calcular word-embeddings

- Word2Vec (Mikolov et al.2013)
  - En CBOW las palabras se representan como one-hot vectors del tamaño del vocabulario (200000) y se utiliza una red neuronal con una capa intermedia de una dimensión “pequeña” (300) y una capa final que realiza una clasificación multiclase (cada palabra una clase).
    - Podemos utilizar como datos de entrenamiento todos los textos que queramos

- Los pesos de la capa intermedia constituyen la matriz de word embeddigns.
- Se puede obtener la representación de una palabra en ese espacio multiplicando el vector one-hot de la palabra por esta matriz.





Aproximaciones para calcular word-embeddings

- GloVe: Global vectors for word representation (Pennington et al.2013)
  - Se basa en la factorización de una matriz que almacena estadísticas de co-ocurrencias de palabras en contextos
  - https://nlp.stanford.edu/projects/glove/

Uso de los word-embeddings

- Los word-embeddings han son un área que tiene un notable desarrollo “en abierto” y que permite la reutilización al menos con fines no-comerciales
- Es posible entrenar tus propios word-embeddings a partir de un corpus de documentos usando software libre, por ejemplo,
  - Gensim es una librería de PLN que permite entrenar tus propios word-embeddings con word2vec
    - también está disponible la implementación de Google
  - GloVe también tiene el código disponible en la web de la Universidad de Stanford y también en Gensim

Uso de los word-embeddings

- Sin embargo, esto requiere conocimiento y un corpus de documentos muy grandes.
- Para muchas tareas es mejor usar word-embeddings “pre-entrenados” de gran calidad (transfer learning):
  - Los word-embeddings de word2vec generados por Google
  - los de GloVe de Stanford para distintas configuraciones
  - word2vec sobre Wikipedias en distintas lenguas
- Dado un word-embedding pre-entrenado se puede incorporar en una red neuronal como una capa intermedia
  - o bien de manera estática, no permitiendo que se modifiquen los pesos
  - o dinámica, permitiendo que los pesos se ajusten con los datos de la tarea concreta que se esté resolviendo

Uso de los word-embeddings

- En los frameworks actuales de deep learning existen capas especiales que sirven para manejar los word embeddings
  - En Keras, uno de los framework de deep learning más utilizados, hay una capa especial para manejar los word embedings, el Embedding layer
    - En realidad es como un diccionario que mapea valores enteros (que representan palabras) a vectores densos.
  - Red neuronal en Keras, “misma idea” que MLP (lo veremos en el lab)
    - model = Sequential()
      - Se añaden capas de manera secuencial, una detrás de otra
    - model.add(Embedding(tamaño_vocabulario, tamaño_vector_denso, tamaño_texto))
      - Diccionario de mapeo
    - model.add(Flatten())
      - Aplana las salidas
    - model.add(Dense(1, activation=‘sigmoid’))
      - Una única neurona de salida con activación sigmoid
    - model.compile(optimizar=‘rmsprop’, loss=‘binary_crossentropy)
      - Para compilar el modelo
    - model.fit(X_train, y_train, epochs, batch_size, validation_split)
      - Para entrenar el modelo

RECURSOS DE PLN

Enlaces

- Enlaces interesantes:
  - https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/
  - http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

- **Gensim**
  - **https://radimrehurek.com/gensim/index.html**