Procesamiento del lenguaje natural

Aprendizaje profundo

Departamento de Sistemas Informáticos

E.T.S.I. de Sistemas Informáticos - UPM

8 de marzo de 2024

El lenguaje natural es considerado un rasgo distintivo de la inteligencia humana

- Se define como el lenguaje hablado y escrito que utilizamos a diario
- Es el mayor depósito de conocimiento humano que existe
- Permite la interacción hombre-máquina mediante sistemas que entienden el lenguaje

Abordaremos el NLP desde el punto de vista del aprendizaje profundo

- Veremos diferentes formas de representar el lenguaje y la solución más adoptada: los embeddings
- También resolveremos problemas usando CNN y RNN

Introducción

¿Qué es el procesamiento de lenguaje natural?

Área de la IA que cubre la **comprensión** y **generación** de lenguaje natural

- No está limitado únicamente al texto (p.ej. también hablado, lengua de signos, ...)
- Está mucho más avanzado en texto escrito (es del que hay más datos disponibles)
- También está relacionado con otros campos como la lingüística, la psicolingüística, las ciencias cognitivas y la estadística
- Uno de sus objetivos es conseguir que los ordenadores trbajen con nuestro lenguaje en lugar de trabajar nosotros con el suyo

Por cierto, la voz artificial suena cada vez más natural

- Con inflexiones tonales y prosódicas que imitan la humana (p.ej. VALL-E)

No pertenece únicamente al campo de la IA

Áreas de trabajo en NLP

Podemos dividir el NLP en dos áreas de trabajo bien diferenciadas:

1. Natural language generation (NLG): Generación de lenguaje a partir de representaciones internas
   - Puede considerarse como un componente de traducción entre los datos y el lenguaje natural
   - Se subdivide en múltiples tareas: planificación del discurso, selección léxica, ...
2. Natural language understanding (NLU): Comprensión lectora de las máquinas
   - Transformar el lenguaje natural en una representación adecuada para la máquina
   - Normalmente, requiere diferentes niveles de análisis: morfológico, sintáctico, semántico, discurso, ...

El NLU es más complejo que NLG (aunque ***ambos problemas son muy difíciles***)

[1] Artículo: Logical syntax and semantics: Their linguistic relevance

Los dos enfoques principales en NLP

Basados en gramáticas o **modelos lógicos**: Enfoque top-down

- Diseñados para intentar reflejar la estructura lógica del lenguaje
- Surgen de las teorías lingüísticas de N. Chomsky a mediados de los 1950[1]
- Desarrollar reglas de reconocimiento de patrones estructurales, utilizando un formalismo gramatical específico
- Los patrones de reconocimiento se definen por las reglas e información adicional

Basados en datos: Enfoque bottom-up

- Abarcan tanto modelos probabilísticos como basados en machine learning
- Se parte de ejemplos para calcular su probabilidad de aparición en un contexto
- Permiten predecir la siguiente unidad en un contexto determinado **sin reglas gramaticales explícitas**

Ventajas y desventajas

Ventajas

- Un usuario podría **hacer preguntas en su idioma natal** en lugar de aprender una sintaxis de consulta específica
- La exactitud de las respuestas tiende a aumentar según aumenta la cantidad de información relevante en la pregunta
- Las respuestas se pueden ofrecer en el idioma del usuario

Desventajas

- El sistema puede no ser capaz de dar una respuesta correcta si la pregunta está mal formulada o es ambigua
- Los sistemas se construyen para una tarea única y específica
  - Son incapaces de adaptarse a nuevos problemas

Ejemplos de aplicaciones

Recordatorio de tipos de problema

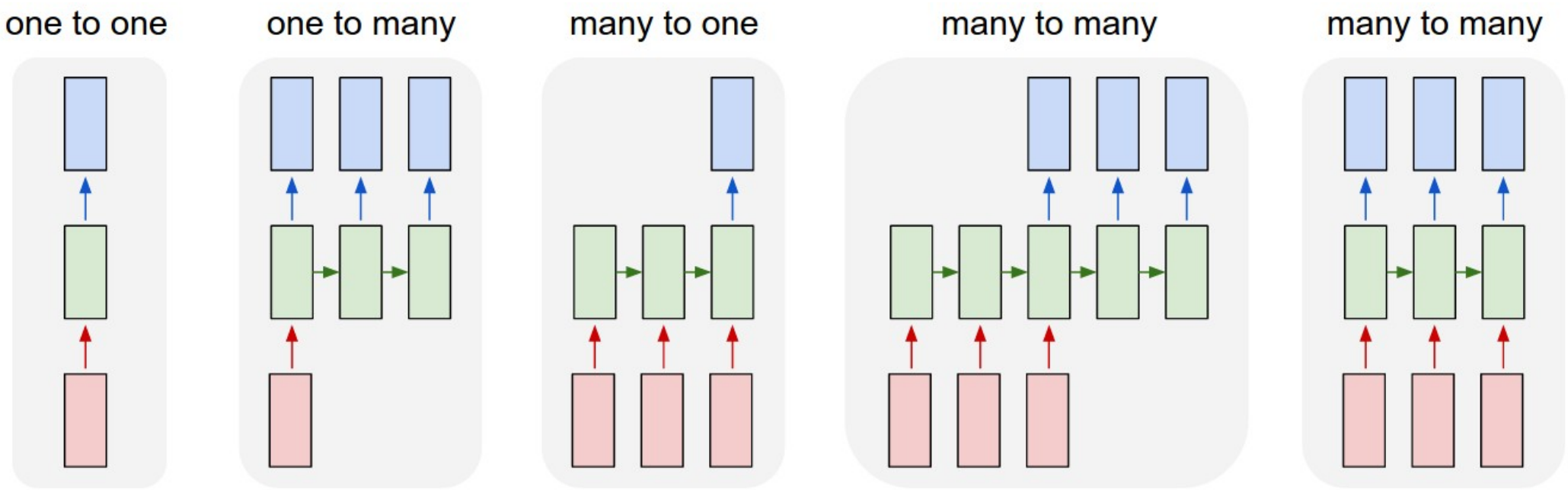

***Figura 1**. Diferentes problemas en redes neuronales recurrentes. Autor: Andrej Karpathy*[6].

[6] Extraído de la entrada del blog del autor The Unreasonable Effectiveness of Recurrent Neural Networks.

Aplicaciones one-to-many

En NLP no existen demasiadas aplicaciones one-to-many

- Después de todo, una parte del NLP es el NLU, que trabaja con secuencias de entrada

Las que existen suelen trabajar a partir de una semilla inicial para generar texto

- Etiquetado de vídeos
- Creación de pies de foto para imágenes

Aplicaciones many-to-one

Tenemos, por ejemplo, los motores de búsqueda como Google, Yahoo o Bing

- El motor de búsqueda de Google incluso entiende tanto el texto como el contexto
  - La misma query para dos personas o momentos diferentes arroja resultados diferentes

Las redes sociales (p.ej. twitter) infieren a partir de los intereses de los usuarios

- También de los contenidos generados por los usuarios en forma de publicaciones (a ver si no por qué iban a haber pagado tanto por WhatsApp)

En general, soluciones para clasificar texto

- Filtros de spam[2], procesamiento de historiales clínicos[3], detección de paráfrasis, ...

2 Artículo: Deep learning to filter SMS Spam
3 Artículo: Natural Language Processing of Clinical Notes on Chronic Diseases: Systematic Review

Aplicaciones many-to-many

Ejemplo típico $\rightarrow$ asistentes virtuales (p.ej. Siri de Apple o Alexa de Amazon)

- O implementaciones libres como Open Assistant[4], Mycroft AI[5] y Dragonfire[6]

Generación de información a partir de una secuencia de datos previa

- Desde sugerencias de continuación de una frase o incluso ¡juegos completos![7]

Recordemos que las many-to-many también funcionan como $T_x = T_y$

- Etiquetado gramatical, corrección ortográfica[8], ...

4 Proyecto: Open Assistant
5 Proyecto: Mycroft AI
6 Proyecto: Dragonfire
7 El juego AI Dungeon es una aventura conversacional totalmente generada por un modelo de lenguaje basado en un modelo propio de lenguaje (originalmente fue GPT-2)
8 Artículo: Personalized spell checking using neural networks.

Representación del lenguaje

n-gramas

Si tenemos la grase «el deep learning es la leche» y cada *token* es una palabra

- Modelo 1-gramas (unigrama): convierte la frase en una secuencia de palabras
  - `{el, deep, learning, es, la, leche}`
- Modelo 2-gramas (bigrama): convierte la frase en combinaciones de dos palabras
  - `{(el, deep), (deep, learning), (learning, es), (es, la), (la, leche)}`
- Modelo 3-gramas (trigrama), 4-gramas (tetragrama), 5-gramas (pentagrama), ...

Desglosar un texto en n-gramas es un proceso muy usado en el NLP tradicional

- Esencial para mantener el recuento de conceptos en frases
- Constituye la columna vertebral de los procesos matemáticos tradicionales

Bag-of-Words (BoW)

Representación de un texto como el multiconjunto[9] de sus palabras

La forma más común de representación es como formato tabular:

- **Fila**: Observación (p.ej. documento)
- **Columna**: Token (p.ej. n-grama, ...)
- **Celda**: El recuento del token en esa observación

ID	Agua	Archaeopteryx	Zumo
001	12	12	0
002	1	5	...
...	...	...	...
999	35	0	13

Solo se trabaja con frecuencias ya que el resto de información se ha perdido

9 En el contexto de la informática, un multiconjunto (bolsa o *bag*) es un conjunto que puede contener elementos repetidos (https://es.wikipedia.org/wiki/Multiconjunto)

PREPARÉMONOS PARA CONOCER LA VERDAD ABSOLUTA DEL NLP

LAS PALABRAS SON CADENAS DE TEXTO

(¿y el problema?)

¡EN APRENDIZAJE PROFUNDO SE TRABAJA CON NÚMEROS!

Codificación one-hot

Codificación *one-hot*

La codificación *one-hot* transforma una variable categórica en una numérica

- Por ejemplo si nuestro idioma se compone de las palabras `quiero`, `comer` y `pizza`, podemos codificarlas como:
  - `[1, 0, 0]` → `quiero`
  - `[0, 1, 0]` → `comer`
  - `[0, 0, 1]` → `pizza`

Rápido, sencillo, claro, ¿por qué no es una buena idea para codificar palabras?

- No es capaz de almacenar información de las relaciones con otras palabras
  - e.g. `gato` y `perro` están igual de relacionados que `gato` y `corneta`
- El espacio que ocupa en memoria es sencillamente prohibitivo

No lo imaginamos:

$\approx 10^5$ **palabras** $\implies$ $10^5 \times 10^5 = 10^{10}$ **elementos (bits)** $\approx 1,15$ **GB**

Vectores de palabras (*word vectors*)

	Género	Edad	Lugar
rey	+1	+1	0
reina	+1	+1	0
príncipe	+1	+1	0
princesa	+1	+1	0
manhattan	+1	+1	0
tomelloso	+1	+1	0

- **Vectores de características** en NLP
- Más compactos que one-hot

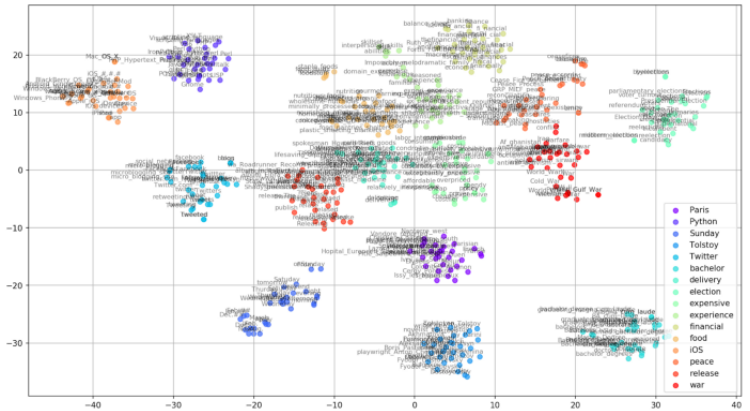

***Figura 2**. Visualización de clústers de palabras.*

Las palabras son vectores: $V(rey) - V(hombre) \approx V(reina) - V(mujer)$

¿Qué características escoger para las palabras?

Muy difícil determinar las características relevantes para clasificar las palabras

- ¿Qué características son las que vamos a utilizar?
- ¿Es relevante el rasgo triste? ¿Cómo medimos la tristeza de una patata?
- ¿Vamos a estudiar cada palabra individualmente?
- ¿Cuántas palabras existen en una lengua?
- Necesitamos formas automatizadas de crear esos vectores de palabras

Las inferidas en aprendizaje no supervisado no tienen por qué tener sentido

- Mientras tengan sentido geométrico, dejemos que el algoritmo infiera las mejores
- Análogo a vectores latentes, como las capas internas de una red neuronal

Embeddings

Un *embedding* es simplemente una **matriz de vectores de palabras apiladas**

- A partir de 2013[11] se sentaron las bases de las técnicas actuales
- Algunos autores los denominan *word embedding* o *word embedding matrix*

Convenio:

- $V \rightarrow$ Número de filas $=$ Tamaño del vocabulario $=$ Número de palabras distintas
- $D \rightarrow$ Dimensión del embedding $=$ Tamaño del vector de características

Transfer learning $+$ *embeddings* $=$ Pilar fundamental del NLP

- Veremos los dos métodos más conocidos: Word2Vec y GLoVe

[11] Existen desde el 2001 (A neural probabilistic language model), pero el verdadero boom fue a partir del 2013 (Distributed representations of words and phrases and their compositionality), momento en el que se mejoró su eficiencia de manera dramática.

¿Cómo se usan los embeddings?

Generalmente a través de transfer learning, lo cual es bueno porque:

- Hay gente/compañías con más recursos computacionales que nosotros
- Encontrar los mejores hiperparámetros es una tarea ardua (ensayo y error)
- Evitamos otros preprocesos sobre los datos (p.ej. mitigación del sesgo)

Truco computacional: Obtener el vector de palabra indexando, no multiplicando

- En Keras significa no usar una capa `Dense()`, sino las capas `Embedding()`

¿Y si el dataset tiene palabras que no existen en el embedding preentrenado?

- Abreviaturas, modismos, palabras mal escritas, palabras poco comunes, ...
- Los pesos pueden ser inicializados aleatoriamente o a 0

¿Deberíamos entrenar los embeddings?

`model.fit(X, Y)` **actualiza** automáticamente **todos los parámetros** del modelo

- ¿Entrenamos el embedding? La mayoría de las bibliotecas no son tan finas
  - Vamos, o lo entrenamos entero, o no lo entrenamos
- No olvidemos de que *las palabras poco comunes son*, por definición, *poco comunes*

Por lo general, no tenemos que molestarnos en hacer un fine-tunning del embedding

- Pero a veces es bueno experimentar con nuestros propios datos
- En Keras, si lo necesitamos, basta simplemente con cambiar el atributo `trainable`:
  - `embedding_layer.trainable = True` → fine-tunning (valor por defecto)
  - `embedding_layer.trainable = False` → Pesos constantes

Creación y visualización de *embeddings*.ipynb

Word2vec

Introducción

Word2vec es una de las técnicas existentes para representar texto como vectores

- Su eficacia reside en su capacidad para agrupar vectores de palabras similares
- Estima el significado de una palabra basándose en sus apariciones en el texto

Las operaciones algebraicas sobre los vectores permite jugar con sus significados

- $V(peral) - V(pera) + V(nuez) = V(nogal)$

Existen dos arquitecturas dependiendo de su cometido

- Continous bag-of-words (CBoW): Predice una palabra según las circundantes
- Skip-grams: Predice las palabras circundantes en función de la central

¿Por qué es interesante?

Permite a las palabras **relacionarse en dimensiones aprendidas**

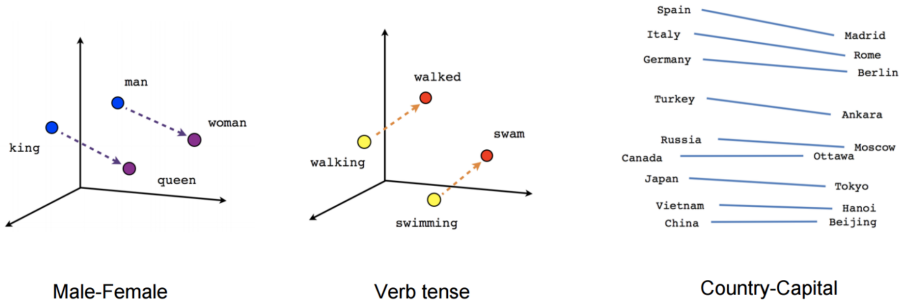

***Figura 3**. Relaciones entre palabras aprendidas por Word2Vec.*

- Surgieron muchos estudios para intentar aclarar estas relaciones lineales[12]
- Hubo estudios que demostraron que las relaciones no están exentas de sesgo[13]

12 Algunos de estos artículos: Evaluating the stability of embedding-based word similarities, A latent variable model approach to pmi-based word embeddings, The strange geometry of skip-gram with negative sampling y Factors influencing the surprising instability of word embeddings.

13 Artículo Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

Arquitectura CBoW (I)

Idea: Tomar los n vectores anteriores y posteriores del objetivo y sumarlos

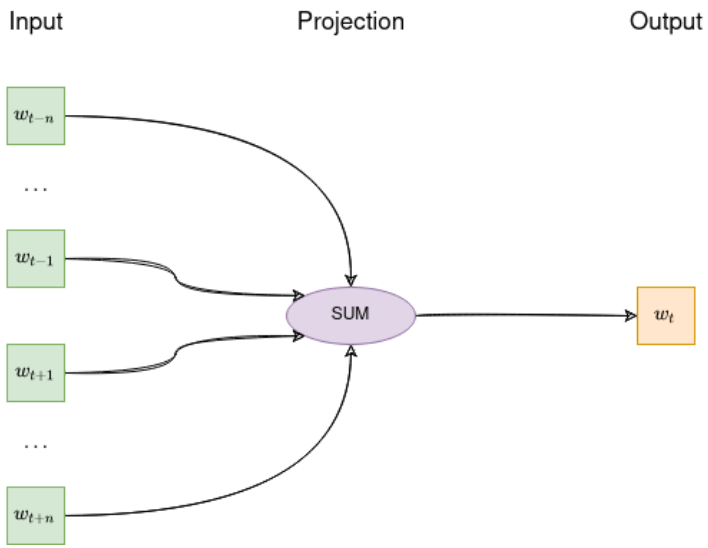

***Figura 4**. Esquema de arquitectura CBoW. La suma de los vectores es suficiente para deducir la palabra.*

Nota $\rightarrow$ Al igual las *bag-of-words*, *las CBoW también eliminan el orden*

Arquitectura CBoW (II)

Ejemplo de frase: «Hoy me *voy* de parranda»; ventana: 5 palabras (± 2 palabras)

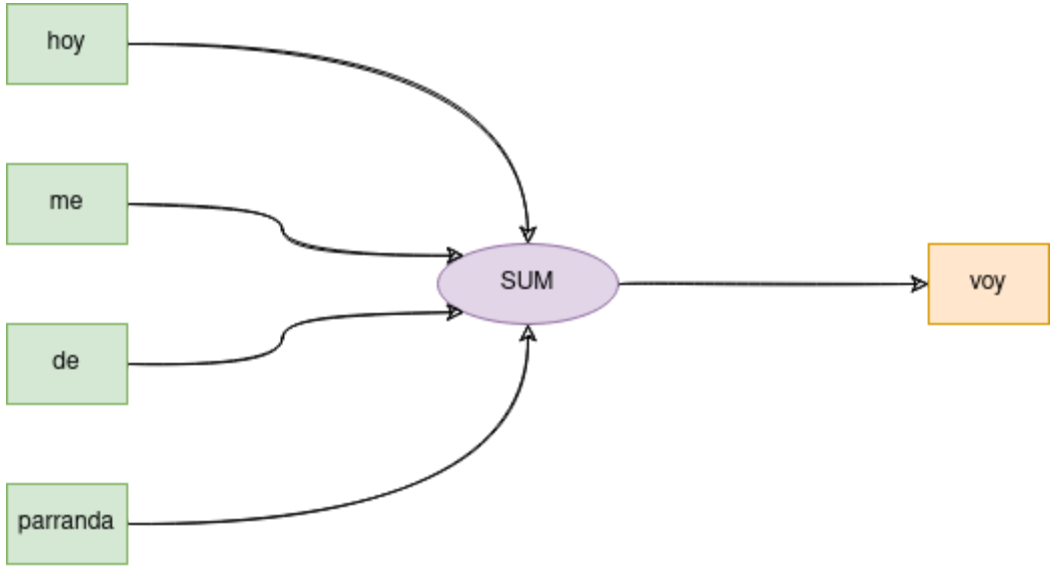

***Figura 5**. Entrenaremos el modelo indicando que debe de inferir `voy` a partir de `hoy`, `me`, `de` y `parranda`.*

Arquitectura CBoW (III)

Cada palabra (entrada y salida) se codifican mediante one-hot

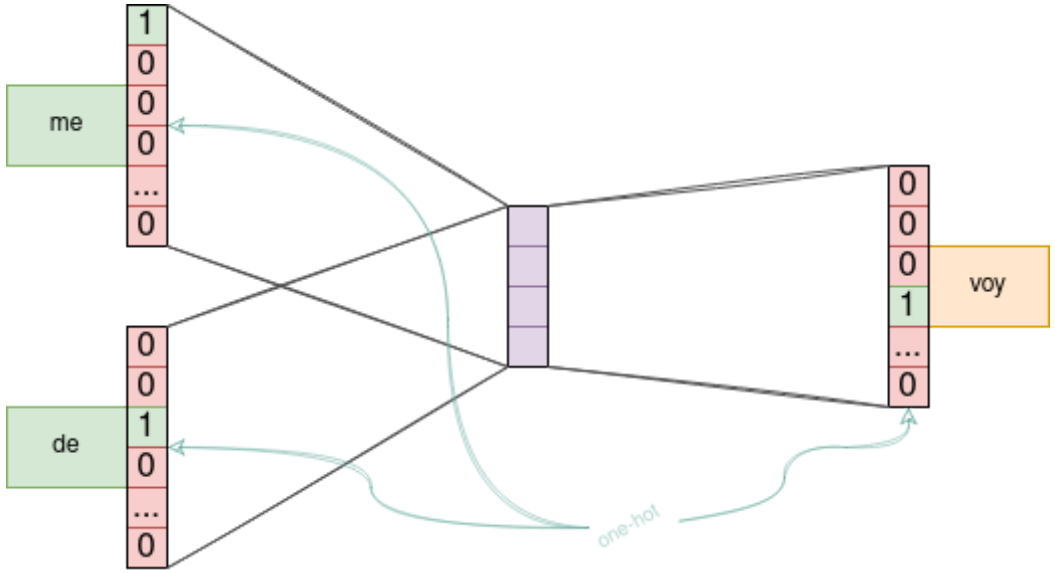

***Figura 6**. Codificación de cada palabra en one-hot. Se muestran únicamente dos palabras por motivos de espacio.*

Arquitectura CBoW (IV)

Nuestro objetivo es aprender dos matrices: $W_{V\times N}$ y $W'_{N\times V}$

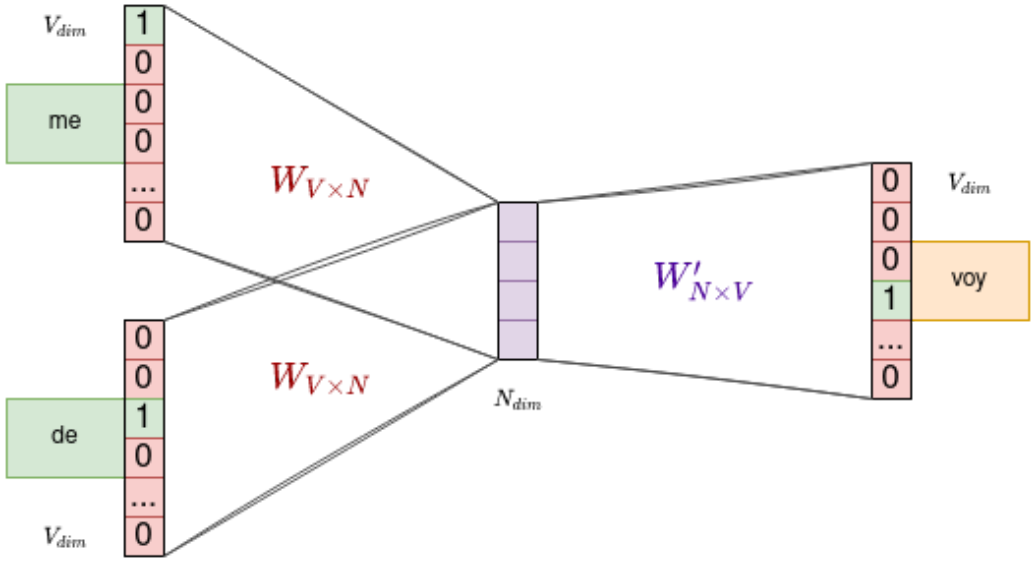

***Figura 7**. Matrices de transformacion de palabra a word vector ($W_{V\times N}$) y viceversa ($W'_{N\times V}$).*

V Será el tamaño del vocabulario y N la dimensión del vector de palabra

Arquitectura CBoW (V)

El producto entre $W^t_{V\times N}$ y el one hot de una palabra nos da su vector

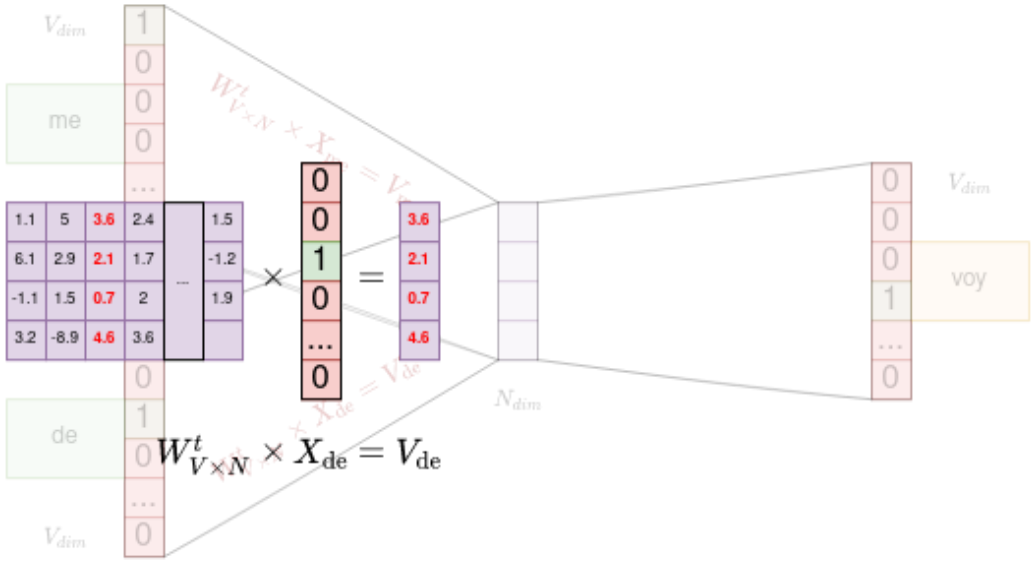

***Figura 8**. Obtención del vector de características de una palabra a partir de su codificación one hot.*

Arquitectura CBoW (VI)

El producto entre el vector de una palabra y $W'_{N\times V}$ nos da un vector de dimensión V

- Este lo aproximamos al one hot más cercano, que será el de la palabra buscada

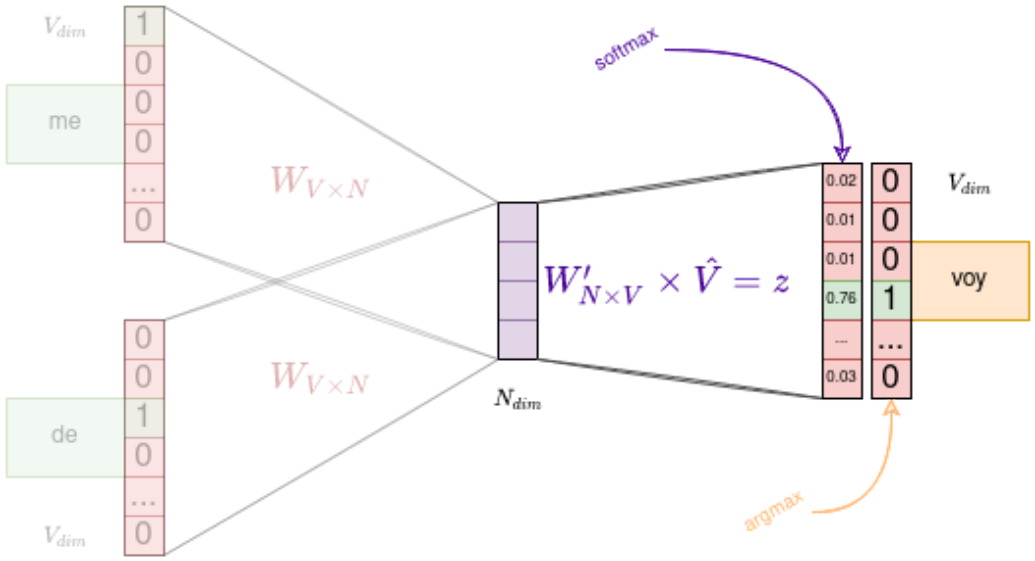

***Figura 9**. Obtención de la palabra a partir de un determinado vector de características.*

Arquitectura CBoW (i VII)

Tras el entrenamiento podemos considerar la matriz $W_{V \times N}$ como el embedding

$$W_{V \times N} = \begin{array}{|cccc|c|c|} \hline 1.1 & 5 & 3.6 & 2.4 & & 1.5 \\ 6.1 & 2.9 & 2.1 & 1.7 & \cdots & -1.2 \\ -1.1 & 1.5 & 0.7 & 2 & & 1.9 \\ 3.2 & -6.9 & 4.6 & 3.6 & & \\ \hline \end{array}$$

***Figura 10**. Nuestro embedding resultante.*

Hay que destacar que:

- La matriz $W'_{N \times V}$ puede servir también como capa de embedding
- De hecho, es bastante común usar ambas matrices como dicha capa

Arquitectura *skip-grams*

Es la arquitectura inversa a la CBoW

- Intenta predecir la ventana de palabras circundantes a partir de la palabra actual

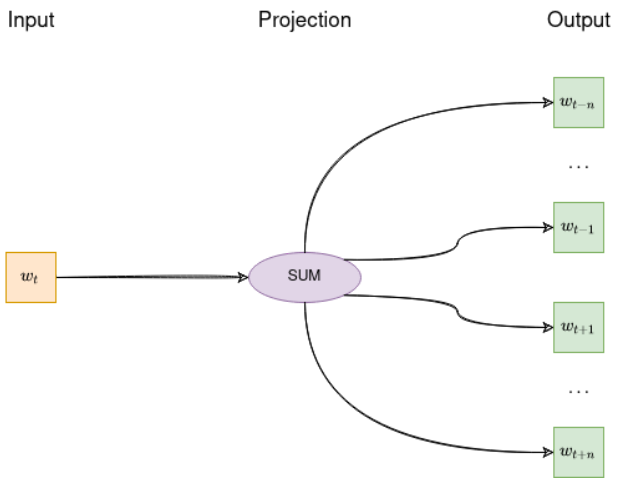

***Figura 11**. Esquema de arquitectura skip-gram. Se deducen los vectores a partir de una determinada palabra.*

Problema mucho menos concreto, pero empíricamente parece que escala mejor

Implementación de *word2vec* basado en *skip-grams*.ipynb

Más allá de Word2vec

Las relaciones obtenidas mediante *word2vec* parecen totalmente intuitivas

- Luego se demostró que se podían obtener mediante factorización matricial[14]

A pesar de los avances, *word2vec* sigue siendo muy utilizado en la actualidad

- Hay estudios que van desde el nivel "subpalabra"[15] al de oración[16]
- Trasciende el NLP, aplicandose al dominios como los grafos[17] o la biología[18]
- Proyectar embeddings de diferentes idiomas sobre el mismo espacio vectorial[19]

14 Artículo Glove: Global vectors for word representation.
15 Artículo: Bag of Tricks for Efficient Text Classification (FastText).
16 Artículos: Skip-thought vectors y Distributed representations of sentences and documents.
17 Artículo: node2vec: Scalable Feature Learning for Networks.
18 Concretamente al estudio de las secuencias biológicas: Continuous distributed representation of biological sequences for deep proteomics and genomics.
19 El objetivo es la transferencia multilingüe. Es un caso concreto Zero-shot Learning (ZSL), cuyo objetivo es conseguir un modelo útil a partir de pocos o ningún ejemplo etiquetado.

Global Vectors for Word Representation (GloVe)

¿Qué es GloVe?

Es una técnica para la obtención de vectores de palabras

- Al igual que word2vec, usa ventanas para obtener estadísticas locales
- Incorpora además estadísticas globales: coocurrencia entre palabras
  - Esto es, uso de dos o más palabras como unidad de significado (e.g. ¡no me digas!)

Idea: Se pueden deducir relaciones semánticas entre palabras mediante su matriz de coocurrencia

- Los cálculos para el cálculo de la matriz son demasiado extensos
- Un buen recurso es Intuitive Guide to Understanding GloVe Embeddings[20]

[20] Que, aunque se titule intuitive, cubre prácticamente todos los cálculos.

Después de GloVe

Cada vez es más fácil aprender una buena proyección de forma no supervisada

- Fantástico para las lenguas de bajos recursos y la traducción no supervisada[21]

Hay muchas más técnicas de embedding que se usan en la actualidad

- Embedding from Language Model (ELMo)[22]: Diseñado para oraciones y párrafos, aunque también se usa para palabras o caracteres
- Bidirectional Encoder Representations from Transformers (BERT): Arquitectura avanzada que hace uso de transformers para comprender mejor el lenguaje
- Sentence-BERT[23] es una modificación sobre BERT para mejorar el rendimiento al trabajar sobre frases

21 Artículo: A survey of cross-lingual word embedding models.
22 Artículo: Deep contextualized word representations.
23 Artículo: Sentence-bert: Sentence embeddings using siamese bert-networks.

Arquitecturas y problemas

Redes neuronales aplicadas al NLP

Desde el 2013 las redes neuronales se usan extensivamente en NLP:

- **CNN**: Ampliamente utilizadas en visión, se aplican también al lenguaje[25]
  - En texto los filtros solo se mueven a lo largo de la dimensión temporal
  - Más paralelizables: cada instante depende del contexto local, no de todos el histórico
- **RNN**: La elección obvia para secuencias dinámicas, omnipresentes en NLP
  - Hasta 2013[24] se pensaba que las RNNs eran difíciles de entrenar; En la actualidad no es así
  - CNNs y RNNs se suelen combinar[26] e incluso usar para acelerar LSTMs[27].
- **Recursive Neural Networks**: El lenguaje es intrínsecamente jerárquico
  - Secuencia ≡ árbol, donde cada nodo calcula una representación a partir de las de los hijos

24 Artículo: Training recurrent neural networks.
25 Artículos: A convolutional neural network for modelling sentences y Convolutional neural networks for sentence classification.
26 Artículo: Dimensional sentiment analysis using a regional cnn-lstm model.
27 Artículo: Logical syntax and semantics: Their linguistic relevance.
28 Artículo: Recursive deep models for semantic compositionality over a sentiment treebank.
29 Artículo: Improved semantic representations from tree-structured long short-term memory networks.

Redes neuronales convolucionales

¿Por qué son útiles? Dos razones fundamentales:

- Útiles para identificar donde pueden existir pistas locales en los datos de entrada
- Más *rápidas* y *sencillas* de codificar que las RNN

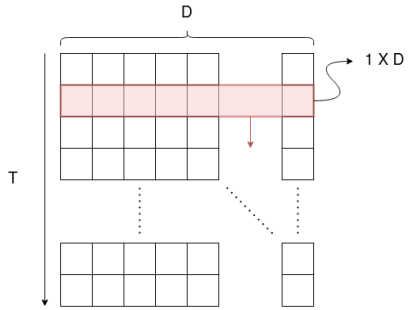

***Figura 12**. Convolución 1D aplicada al NLP.*

Además, estamos enfocando el NLP desde una perpectiva de DL, y las CNNs son DL

Y funcionan como sustitución directa de las RNN, todo ventajas

En definitiva, nos *ayudan a conocer lo más básico del problema* fácil y rápidamente

Clasificación de texto con CNN.ipynb

Redes neuronales recurrentes simples

Son la primera aproximación recurrente obvia para el NLP

- No son muy usadas, pero sirven como para comprender arquitecturas más complejas

Afortunadamente, todas las interfaces de aprendizaje automático son iguales

- Bueno, quizá las RNNs sean un poco más complejas, pero mínimamente
- Casi se pueden utilizar como sustitutos de cualquier otro modelo basado en ANNs

Clasificación de texto con RNN.ipynb

Redes neuronales recurrentes bidireccionales

Supongamos que queremos identificar personas y compañías (Fuente: Fallout 2)

- ***General*** *clothing cannot be repaired by merchants that offer repair services*
- ***General*** *Atomics International has the finest industrial robots in the world*
- ***General*** *Constantine Chase is a gruff and determined general officer*
- Ninguna RNN simple, por sí misma, tendrá contexto suficiente al analizar ***General***

Las RNN bidireccionales surgen como solución al problema de falta de contexto

- Después de todo, nosotros humanos al leer vemos toda la frase simultáneamente
- Son RNN pero en dos pasos: secuencia directa más secuencia en orden inverso

Apenas implica cambios en el código: de `LSTM(M)` a `Bidirectional(LSTM(M))`

Sequence to sequence (seq2seq) (I)

Esta arquitectura se compone normalmente de dos componentes clave:

- ***Encoder***: Procesa y codifica la secuencia de entrada en un único vector de contexto
- ***Decoder***: Genera toda una secuencia a partir del vector de contexto

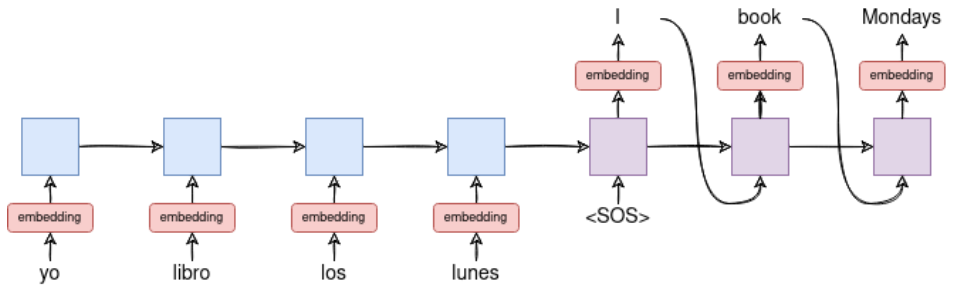

Figura 13. *Esquema de arquitectura seq2seq para un problema de pregunta-respuesta.*

Sí, lo de codificar y decodificar es una idea que se lleva mucho en DL

Sequence to sequence (seq2seq) (II)

Sobre el *encoder*:

- No se usa la salida de la red porque no estamos haciendo ninguna predicción
  - Sólo mantenemos el estado final h_t (y c_t si usamos una LSTM)
  - En keras: `return_sequences=False`
- Representa de alguna manera la secuencia de entrada en un vector
  - Se llama *encoder* porque **codifica/comprime la entrada original**

Sobre el *decoder*:

- Equivalente a un problema *one-to-many*
- Debe tener el mismo tamaño de vector para el estado
  - El estado final anterior debe pasarse como estado inicial de esta nueva red h'_0
- El vector de entrada X será un token `<SOS>` (del inglés *Start of Sequence*)

Sequence to sequence: Detalles de implementación

Usar <SOS> más $y(t)$ como $x(t+1)$ funciona bien para generar predicciones

- Para el entrenamiento, sin embargo, funciona mucho mejor el teacher forcing
  - Ofrecer la verdadera secuencia destino como Xt en lugar de la palabra generada en Yt−1
- ¿Por qué? Porque es difícil aprender a generar toda la secuencia de una vez
- Los humanos hacemos algo similar cuando estamos aprendiendo un idioma
  - Cuando vamos traduciendo en un idioma extranjero, cometemos errores
  - Si nadie nos corrige en un principio, lo más normal es que vayamos cayendo en barrena
  - Si un profesor nos corrige, podemos seguir trabajando a partir de la frase corregida

OJO: Si la secuencia no empieza por <SOS>, la red aprenderá a copiar la entrada

¡Atención!

¿Qué es esto del mecanismo de atención?

Una de esas ideas felices que de repente se convierten en la más influyente

- Seq2seq sigue sin solucionar bien el problema de la memoria a largo plazo

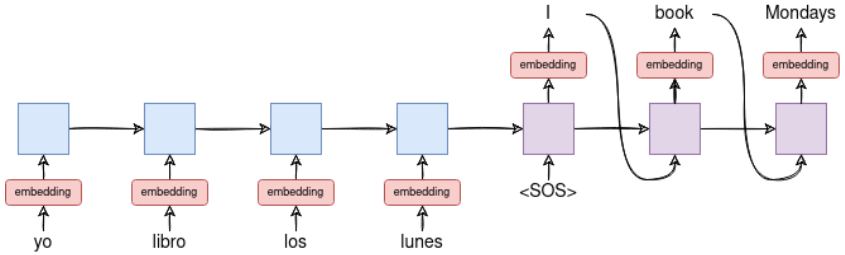

***Figura 14**. Esquema del mecanismo de atención en seq2seq.*

Se inspira en el mecanismo de atención del cerebro

- En el *encoder* ahora interesan todos los estados ocultos
- El *decoder* usará como vector de contexto su *suma ponderada*

Transformers (I)

Son la base del estado de la cuestión actual en NLP[30]

- No deja de ser un modelo secuencia a secuencia
- A diferencia de los anteriores no utiliza RNNs

En los últimos años, las arquitecturas de transformers han avanzado mucho:

- BERT[31]: Uno de los principales avances sobre el que h se apoyan muchos modelos
- GPT-4[32]: Uno de los modelos más avanzados actualmente, el cual requiere licencia
  - Choca bastante con los principios de OpenAI[33], aunque existen alternativas abiertas[34]

30 Artículos: *Attention is all you need*, *Transformers: State-of-the-art natural language processing*.
31 Presentado por Google. Artículo: *Bert: Pre-training of deep bidirectional transformers for language understanding*.
32 Presentado por OpenAI. Artículo: *GPT-4 Technical Report*.
33 La misión de OpenAI (openai.com) es asegurar que la AI beneficia a la humanidad en su conjunto.
34 Más información en https://eleuther.ai/projects/gpt-neo y https://eleuther.ai/projects/gpt-neox

Transformers (y II)

Se trata de un modelo complejo del que sólo explicaremos por encima algunos detalles

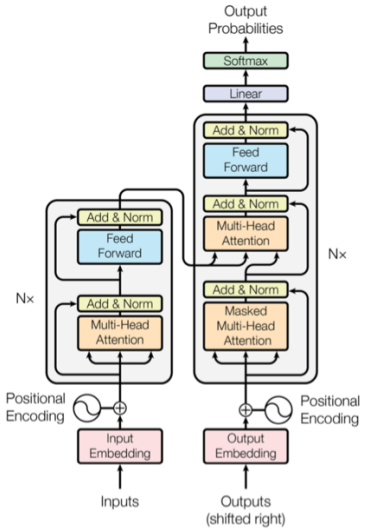

***Figura 15**. Esquema de la arquitectura transformer. Fuente: Attention is all you need..*

- Izquierda: *encoder*, derecha: *decoder*
- Cada bloque de autoatención determina con qué palabras del contexto está relacionada la que se está procesando
- Es un problema de generación de secuencia $\rightarrow$ En la decodificación se trabaja iterativamente ($X_t = Y_{t-1}$)
- La salida *softmax* indica qué palabra es la más probable a continuación

Principales desafíos

Los principales desafíos en el NLP

El lenguaje natural es **orgánico**, es decir, **poco consistente** (cambia, evoluciona, ...)

- Todo lo contrario a los lenguajes informáticos, cuyo objetivo es ser consistentes
- No involucran solo al lenguaje hablado: feedback y lenguaje no verbal, empatía, ...

Lo normal es que no estén bien construidos y que la gramática sea inconsistente

- Y aún expresándonos mal, curiosamente, la mayoría del tiempo nos entendemos

Vamos a ver cuáles son los principales desafíos que plantea el NLP

- Tratar cada reto por separado es complejo, pero todo a la vez una locura
- Por eso los problemas se suelen acotar **mucho** para simplificar su resolución
- Algunos están algo superados; otros siguen siendo un problema abierto

Ambigüedad en los lenguajes naturales

En lenguaje natural una palabra puede tener diferentes significados

- No sólo usamos el conocimiento de la lengua para decidir el significado de algo
- También tenemos en cuenta factores como contexto, deseos, objetivos y creencias

No estamos limitados a palabras, también tenemos:

- Modismos (frases hechas): *Echar una mano*, *Meter la pata*, *Comerse la cabeza*, *Romper a llorar*, *Matar el gusanillo*, *Ir de punta en blanco*, ...
- Hipérboles: *Más inútil que un karaoke de música clásica*, *repetirse más que un yogur de ajo*, ...
- Expresiones generacionales: *LOL*, *shippeo*, *renta mazo*, *frontear*, *flexear* ...
- Semántica ambigüa: *Raúl llamó a su mujer, y también lo hizo Verónica*

Sesgos y prejuicios inherentes al lenguaje (I)

Algunos ejemplos de sesgo para ponernos en contexto

- **Hombre** es a **mujer** como **desarrollador** es a **ama casa**[2]
- **Negro** es a **delincuente** como blanco a **policía**[3]
- El sentimiento de **ira** es mayor en frases con más **sustantivos femeninos**[2]
- El de **ofensivo** es mayor en aquellos tweets escritos por **afroamericanos**[4]

En estos ejemplos, *el algoritmo está expresando esencialmente estereotipos*

- Algunos casos (p.ej. rey-reina) llevan implícitas propiedades en si (p.ej. el género)
- Otros no, pero el lenguaje se las asigna, generalmente, en forma de prejuicios

2 Artículo: *Mitigating gender bias in natural language processing: Literature review*
3 Artículo: *Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings*
4 Artículo: *Racial bias in hate speech and abusive language detection datasets*

Sesgos y prejuicios inherentes al lenguaje (y II)

Es **esencial** tratar los problemas de sesgo **antes de que lleguen al mundo real**

- Ejemplo: Sistema COMPAS, probabilidad de reincidencia en crímenes[5]
  - Este sistema se ha utilizado en algunas de las cortes de justicia de los EEUU
  - Resultó tener un evidente sesgo implícito contra los ciudadanos afroamericanos
    - Hasta el doble de falsos positivos frente a ciudadanos caucásicos
  - Este sesgo implícito no se detectó antes de que se implantara el sistema
    - Se predijo injusta e incorrectamente que muchos afroamericanos volverían a delinquir

Pero, ¿cómo pueden ser parciales los algoritmos si no tienen emociones?

- El problema son los datos, que poseen implícitamente sesgos
  - Además la moral de una sociedad evoluciona con el tiempo

[5] El artículo Are Algorithms Building the New Infrastructure of Racism? aborda el problema de los sesgos y habla no solo del sistema COMPAS, sino de otros sistemas y casos reales donde el sesgo de los datos es relevante

Alternativas para mitigar el sesgo

Abordar el sesgo **en conjuntos de datos**

- Una opción, eliminar datos (o *datasets* enteros) que contengan sesgos
- La forma más aceptada de eliminar el sesgo es diversificarlo
- Muy costoso: requiere analizar *datasets* que no son precisamente pequeños

Abordar el sesgo **en el propio modelo**

- Modificar las representaciones vectoriales de las palabras en los propios modelos[7]
- Vía mucho más eficiente con resultados prometedores (aunque no infalibles)

6 El MIT retiró su conjunto de datos de visión computacional *Tiny Images* al descubir que estaba lleno de sesgos sociales, incluyendo racismo y misoginia.

7 Por ejemplo, los algoritmos *Hard Debias* y *Double-Hard Debias* (artículo: *Double-Hard Debias: Tailoring Word Embeddings for Gender Bias Mitigation*) eliminan información estereotipada (p.ej. «secretaria» y «mujer») manteniendo al mismo tiempo la información de género útil (p.ej. «reina» y «mujer»).

Ironías del NLP

Ironía #1: En la era del Big Data sufrimos de escasez de datos para NLP

- Cada modelo requiere **terabytes de datos para** entender **un idioma** específico
- Los idiomas de los que existen tantos datos se denominan *high resource languages*
  - Sólo 20 de los cerca de 7000 idiomas existentes se pueden considerar de este tipo

Ironía #2: Nos centramos en idiomas que se enseñan bien en todo el mundo

- Pero la mayoría de la gente habla un idioma considerado de escasos recursos
- Desde un punto de vista utilitarista, no merece la pena esforzarse, ¿no?
  - Total, los beneficios obtenidos serán limitados en comparación con el esfuerzo
- **Un objetivo clave del NLP es construir sistemas para romper barreras**
  - Modelos que permitan a la gente leer noticias escritas en otros idiomas
  - Sistemas para hacer preguntas sobre su salud cuando no tienen acceso a un médico

La evaluación de los resultados

La técnica evoluciona, y con ella se allana el camino a tareas más complejas

- Pero aún no disponemos de herramientas para entender por qué ciertas técnicas y arquitecturas funcionan mejor que otras en según qué escenarios
- Es un problema bastante descuidado pero que es extremadamente importante
- Es necesario entender por qué hay enfoques que funcionan y otros no
- Sabiendo esto, debemos desarrollar medidas de evaluación basadas en ello

Necesitamos una nueva generación de *datasets* y técnicas de evaluación

- Así podremos validar si nuestras técnicas generalizan la variedad de un idioma
- También podríamos conseguir conjuntos de datos de inferencia en lenguaje natural mucho más eficientes

Licencia

Puedes encontrar su código en el siguiente enlace:
https://github.com/etsisi/Aprendizaje-profundo