Taller básico de PLN:
Representaciones Vectoriales y Transfer learning

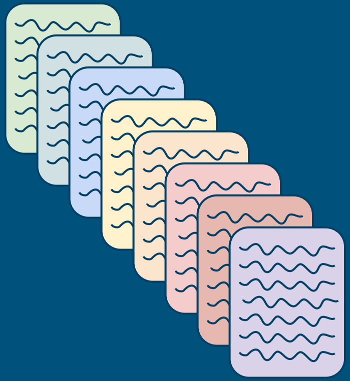
Ana Valeria González
Ximena Gutiérrez-Vasques

Datos de entrada → Modelo Aprendizaje Automático → Output

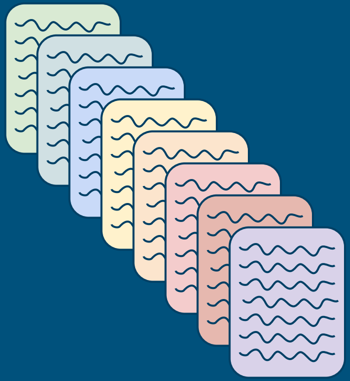
Datos de entrada

Modelo Aprendizaje Automático

Output



Datos de entrada



- modelos de aprendizaje automático toman vectores como entrada, no textos
- Obtener buenas representaciones del lenguaje natural en espacios vectoriales es difícil
- Hoy en día, la forma más común y eficiente de representar texto es utilizando modelos pre-entrenados → una forma de ***transfer learning***

En esta plática...

- Representaciones vectoriales:
    - desde lo básico hasta word2vec
- State-of-the-art
- Un poco más sobre transfer learning
    - algunos conceptos importantes

Trabajemos un ejemplo....

Consideramos un corpus ***C***

d_1	*A Estefanía le gusta comer pizza y comer sushi*
d_2	*A Estefania le gusta jugar béisbol*

Vocab = *{ "a", "béisbol", "comer", "estefanía", "gusta", "jugar", "le", "pizza", "sushi", "y"}*

N = *10*

Vectores de frecuencias

Vectores de frecuencia (Term Frecuency)

d_1	*A Estefanía le gusta comer pizza y comer sushi*
d_2	*A Estefania le gusta jugar béisbol*

Vocab = *{ "a", "béisbol", "comer", "estefanía", "gusta", "jugar", "le", "pizza", "sushi", "y"}*

	a	béisbol	comer	estefania	gusta	jugar	le	pizza	sushi	y
d_1	1	0	2	1	1	0	1	1	1	1
d_2	1	1	0	1	1	1	1	0	0	0

Vectores de frecuencia (Term Frecuency)

- Palabras frecuentes no son necesariamente importantes
    - y, a , los, las, etc (stop words)
    - necesitamos darle menos importancia a estas

Vectores de TF-IDF

Introduce la noción de **<u>peso</u>**

- palabras que ocurren frecuentemente en todo el corpus tendrán menos peso
- palabras menos comunes en todo el corpus tendrán peso más alto

Vectores de TF-IDF

Introduce la noción de **peso**

- palabras que ocurren f … án menos peso
- pa … pus tendrán peso más alto … os

Material adicional sobre TF-IDF en Colab

C … por documento, ***no podemos comparar representaciones a nivel de … abras***

Codificaciones “one hot”

Codificaciones “one-hot”

Asignamos índices a cada palabra en nuestro vocabulario

{ “a” : 0, “béisbol” : 1, “comer” : 2 , “estefanía” : 3, “gusta” : 4, “jugar” : 5, “le” : 6, “pizza” : 7, “sushi” : 8, “y” : 9}

	0	1	2	3	4	5	6	7	8	9
a										
béisbol										
comer										
estefania										
gusta										
...										

Codificaciones “one-hot”

Asignamos índices a cada palabra en nuestro vocabulario

*{ **“a” : 0,** “béisbol” : 1, “comer” : 2 , “estefanía” : 3, “gusta” : 4, “jugar” : 5, “le” : 6, “pizza” : 7, “sushi” : 8, “y” : 9}*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
a	1	0	0	0	0	0	0	0	0	0
béisbol										
comer										
estefania										
gusta										
...										

Codificaciones "one-hot"

Asignamos índices a cada palabra en nuestro vocabulario

{ "a" : 0, ***"béisbol" : 1,*** *"comer" : 2 , "estefanía" : 3, "gusta" : 4, "jugar" : 5, "le" : 6, "pizza" : 7, "sushi" : 8, "y" : 9}*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
a	1	0	0	0	0	0	0	0	0	0
béisbol	0	1	0	0	0	0	0	0	0	0
comer										
estefania										
gusta										
...										

Codificaciones “one-hot”

Asignamos índices a cada palabra en nuestro vocabulario

{ “a” : 0, “béisbol” : 1, “comer” : 2 , “estefanía” : 3, “gusta” : 4, “jugar” : 5, “le” : 6, “pizza” : 7, “sushi” : 8, “y” : 9}

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
a	1	0	0	0	0	0	0	0	0	0
béisbol	0	1	0	0	0	0	0	0	0	0
comer	0	0	1	0	0	0	0	0	0	0
estefania	0	0	0	1	0	0	0	0	0	0
gusta	0	0	0	0	1	0	0	0	0	0
...										

Codificaciones “one-hot”

	0	1	2	3	4	5	6	7	8	9
a	1	0	0	0	0	0	0	0	0	0
béisbol	0	1	0	0	0	0	0	0	0	0
comer	0	0	1	0	0	0	0	0	0	0
estefania	0	0	0	1	0	0	0	0	0	0
gusta	0	0	0	0	1	0	0	0	0	0
jugar	0	0	0	0	0	1	0	0	0	0
le	0	0	0	0	0	0	1	0	0	0
pizza	0	0	0	0	0	0	0	1	0	0
sushi	0	0	0	0	0	0	0	0	1	0
y	0	0	0	0	0	0	0	0	0	1

```
[[1,0,0,0,0,0,0,0,0,0]  #a
 [0,0,0,1,0,0,0,0,0,0]  #estefanía
 [0,0,0,0,0,0,1,0,0,0]  #le
 [0,0,0,0,1,0,0,0,0,0]   #gusta
 [0,0,0,0,0,1,0,0,0,0]   #jugar
 [0,1,0,0,0,0,0,0,0,0]]  #béisbol
```

Codificaciones “one-hot”

- vectores son escasos
- Si el vocabulario es muy amplio, el tamaño de los vectores será demasiado grande.
- no hay noción de similitud
    - $Cos(v1, v2) = 0$

"You shall know a word by the company it keeps."

-John R. Firth (1957)

Vectores basados en predicción de contextos

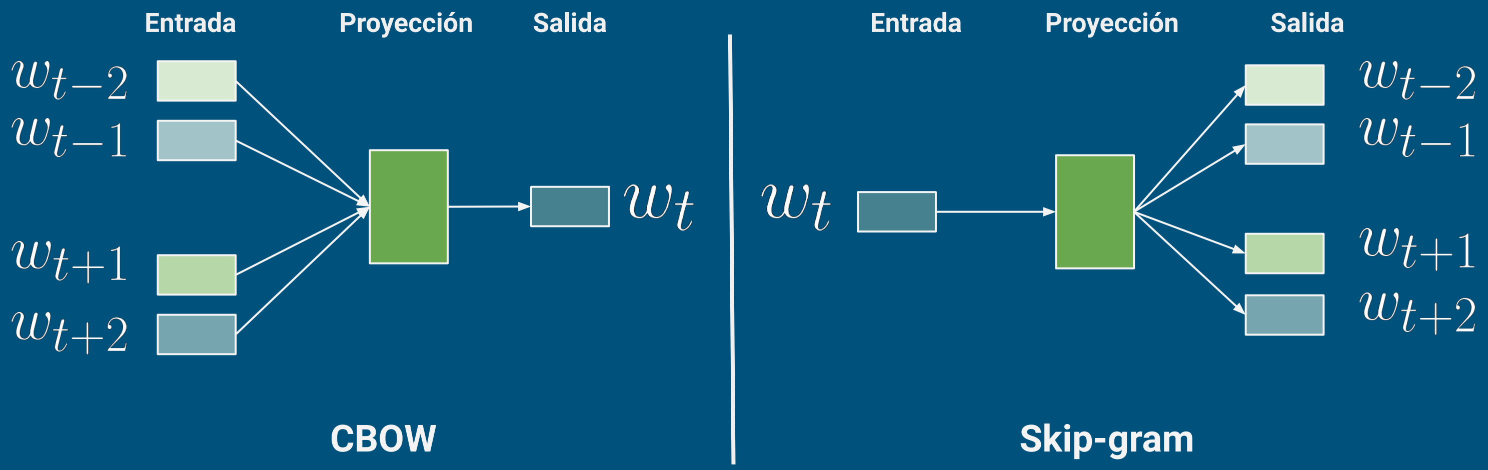
Language modeling : asignar probabilidades a secuencias de palabras

- asignar probabilidad de una palabra dado un contexto i.e $p(w|w_{t-k},\dots,w_{t-1},w_{t+1},\dots,w_{t+k})$

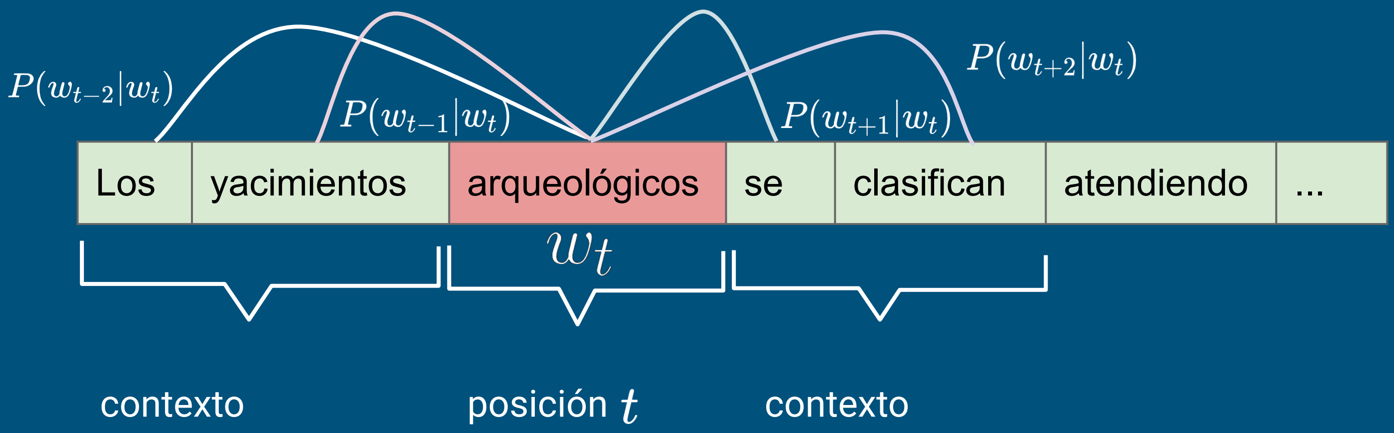
Vectores basados en predicción de contextos (word2vec)

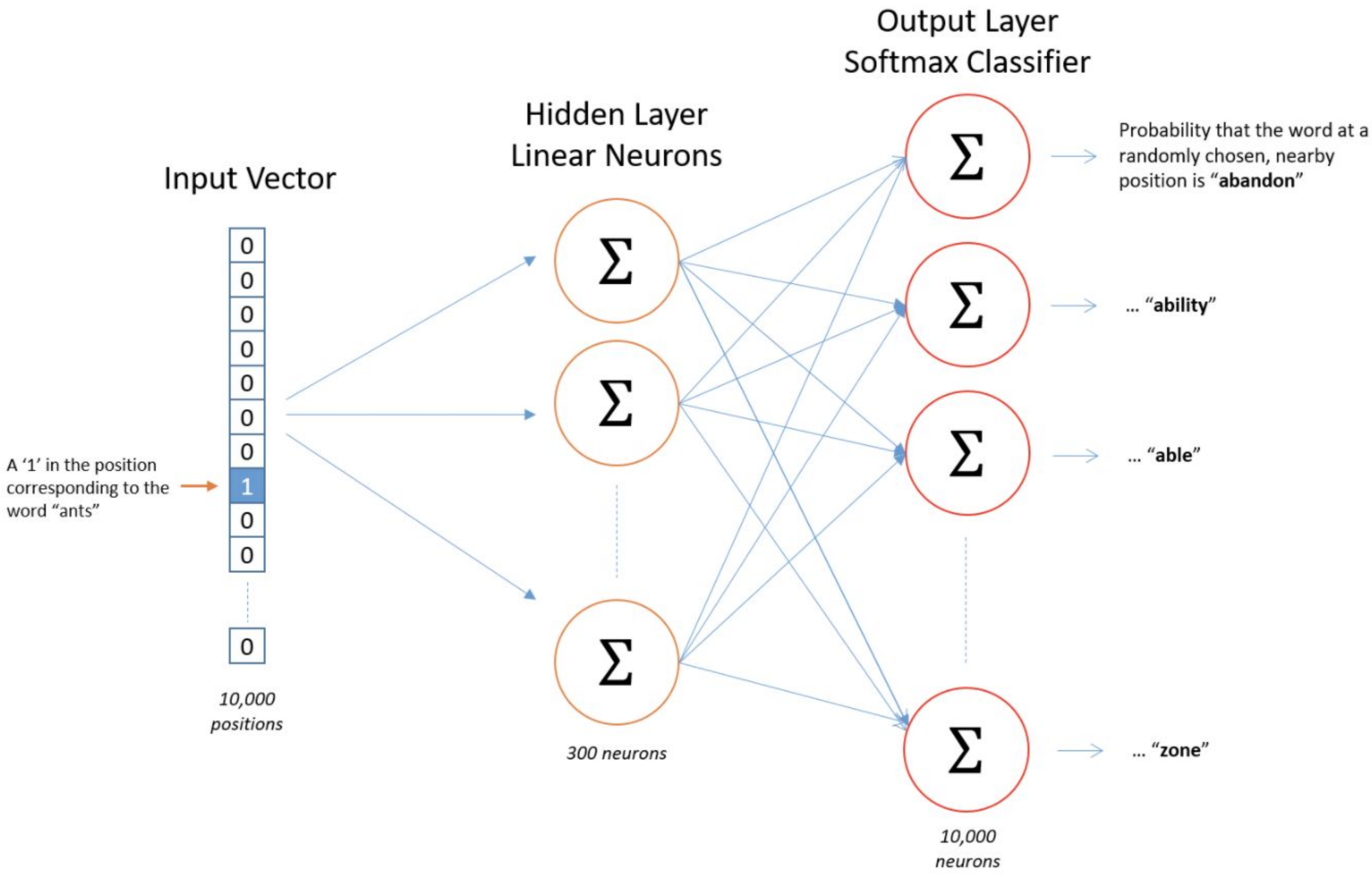
(Mikolov et al. 2013)

- **Continuous bag-of-words:** Dado un contexto, predecir la palabra
- **Skip-gram model** : Dada una palabra, predecir un contexto

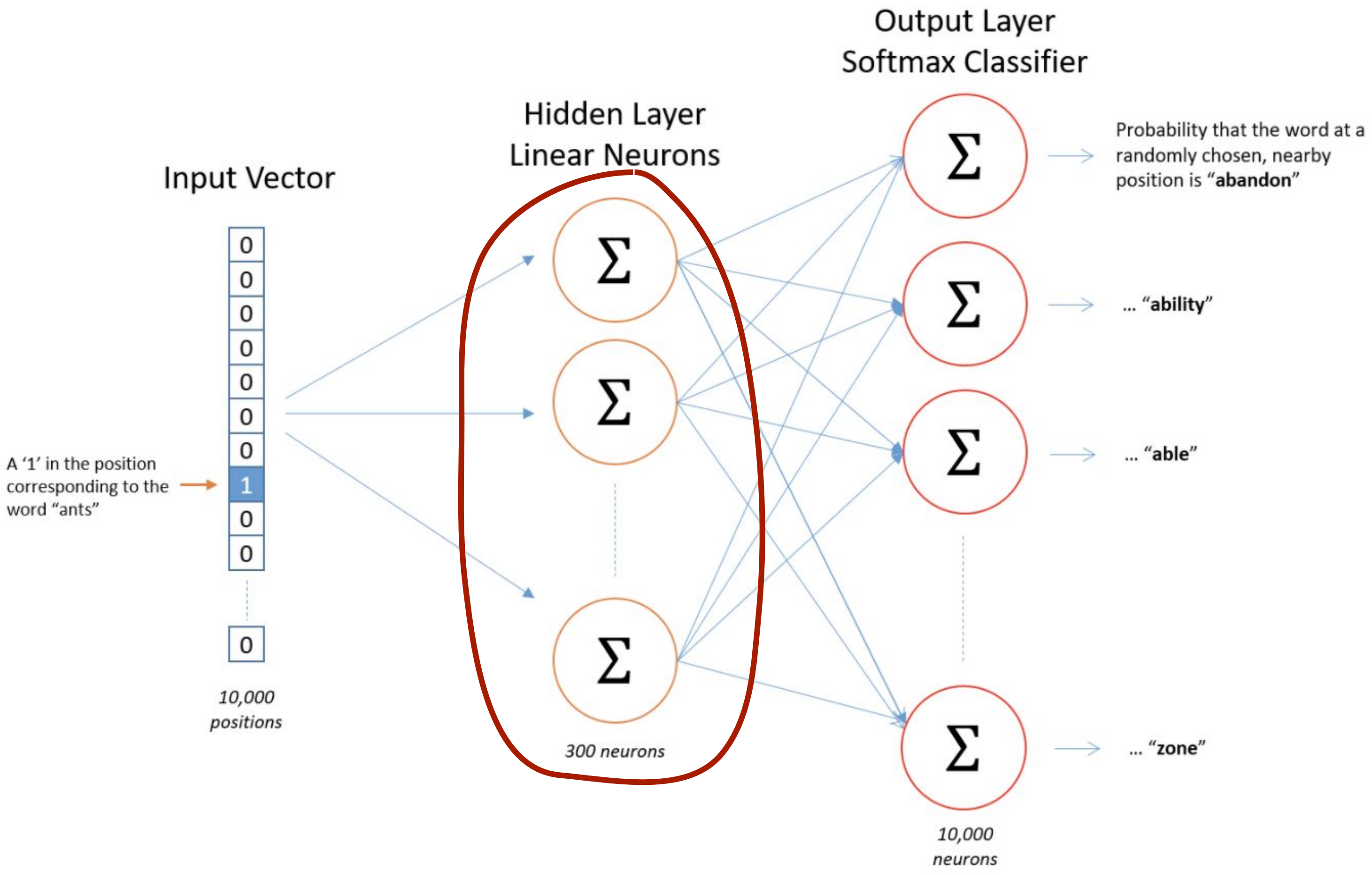


Vectores basados en predicción de contextos (word2vec)





Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
Probability that the word at a randomly chosen, nearby position is "**abandon**"
... "**ability**"
... "**able**"
... "**zone**"
10,000 neurons

Capa Oculta

300 neurons = 300 features

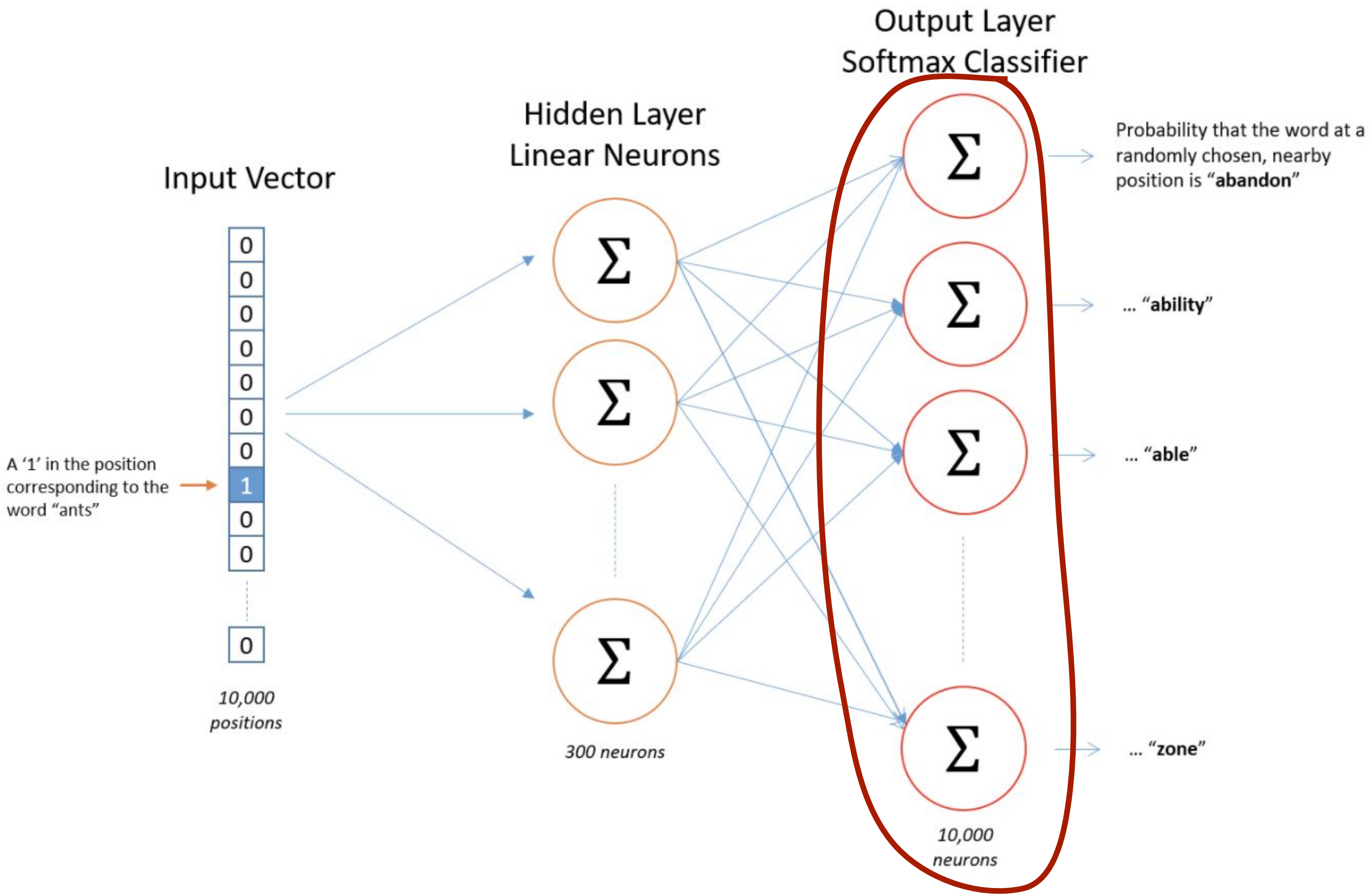
10,000 palabras en
nuestro vocabulario

capa oculta
Matriz de pesos para
cada palabra

input = one hot de la palabra “arqueológicos”

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{pmatrix} 21 & 17 & 3 & 4 \\ 19 & 10 & 1 & 5 \\ 5 & 4 & 18 & 1 \\ 11 & 18 & 25 & 12 \\ 2 & 4 & 6 & 10 \\ 23 & 5 & 8 & 4 \end{pmatrix} = \begin{bmatrix} 11 & 18 & 25 & 12 \end{bmatrix}$$

output de la capa oculta = vector de palabra “arqueológicos”


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"

Output layer

Output de la capa oculta
vector de la palabra
“arqueológicos”

$$\begin{bmatrix} 11 & 18 & 25 & 12 \end{bmatrix} \times$$

weights de la palabra
“yacimientos”

$$\begin{bmatrix} 1 & 2 & 18 & 24 \end{bmatrix} \rightarrow$$

softmax

$$\frac{e^x}{\sum e^x} \rightarrow$$

Probabilidad de que “yacimientos” suceda, dada la palabra “arqueológicos”

esto se hace para todos los pares de palabras

Output layer

weights de la palabra
“yacimientos”

Output de la capa oculta
vector de la palabra
“arqueológicos”

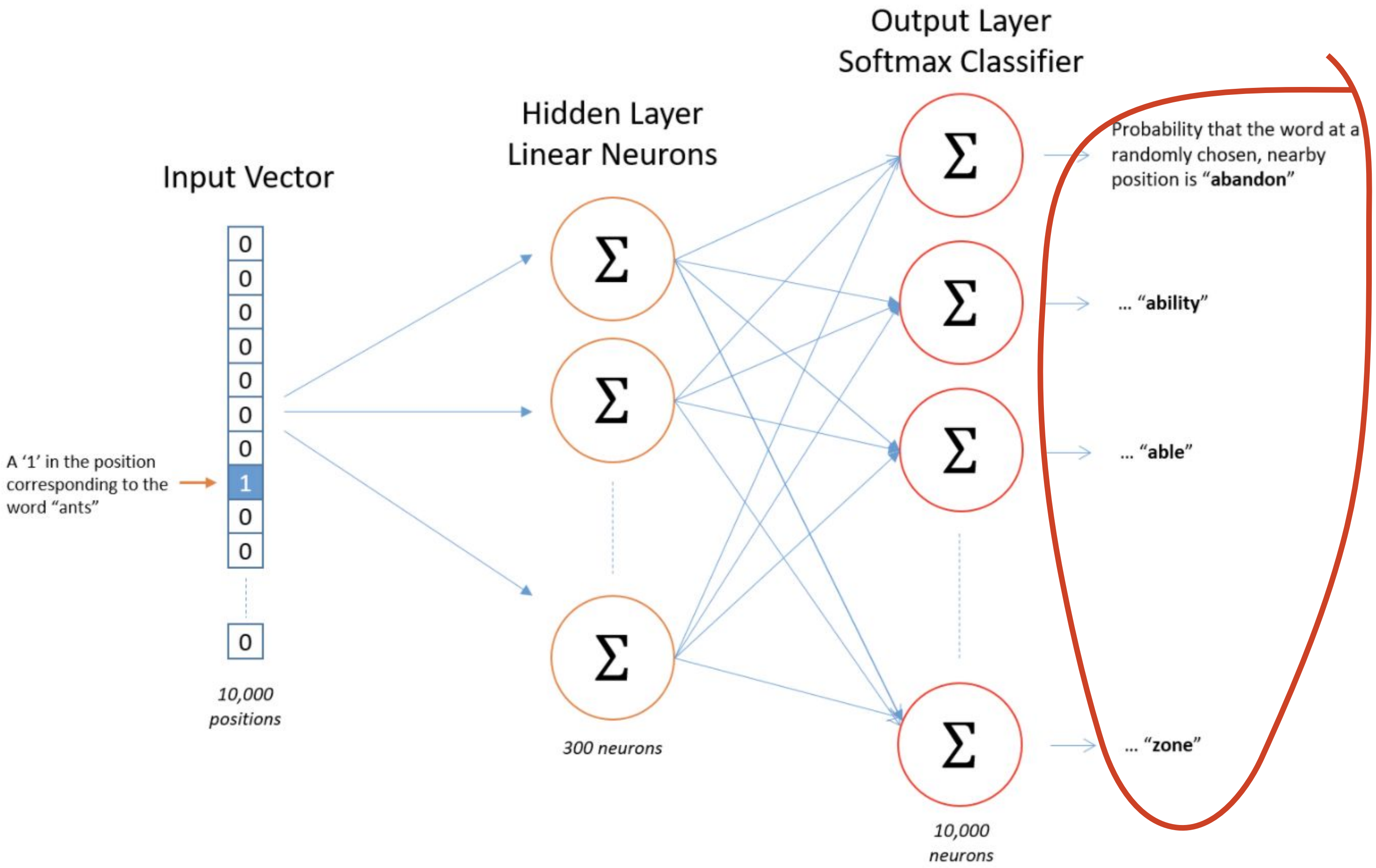
$$\begin{bmatrix} 11 & 18 & 25 & 12 \end{bmatrix} \times \begin{bmatrix} 24 \\ 18 \\ 2 \\ 1 \end{bmatrix} \rightarrow \underset{\text{softmax}}{\frac{e^x}{\sum e^x}} \rightarrow$$

Probabilidad de que “yacimientos” suceda, dada la palabra “arqueológicos”

$$\frac{\exp(\boxed{u_o^T v_c})}{\boxed{\sum_{W \in V} \exp(u_W^T v_c)}}$$

dot product para calcular la similitud de los vectores

Normalizamos utilizando TODO el vocabulario para asignar una distribución de probabilidades



Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"

Forward pass

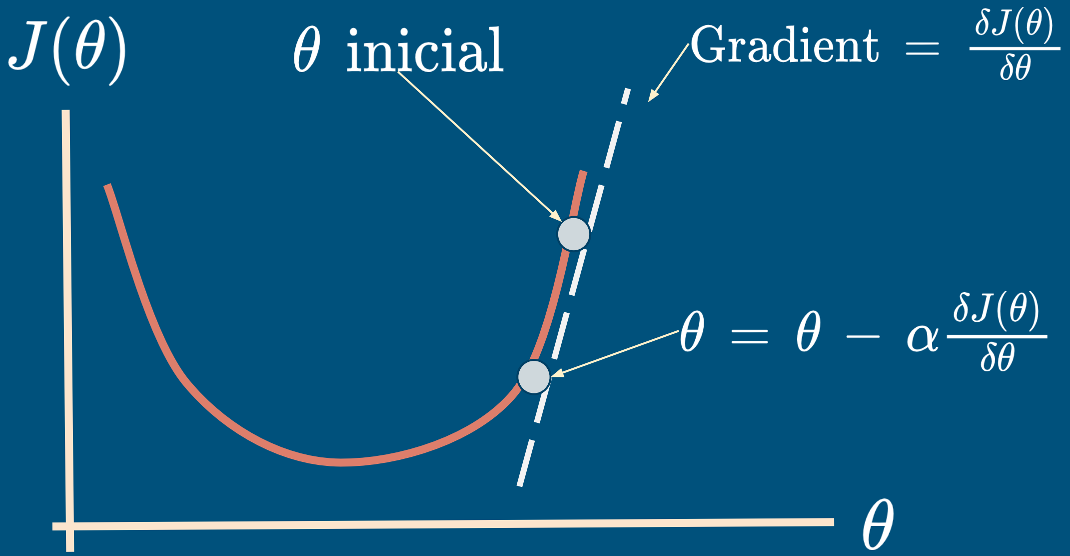
Objective Function (cost, loss, función de pérdida) es el mean **negative log likelihood**

$$J(\theta) = -\frac{1}{T}\log L(\theta)$$

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m; j\neq 0}\log P(w_{t+j}|w_t;\theta)$$

Al minimizar el objective function maximizamos la precisión predictiva

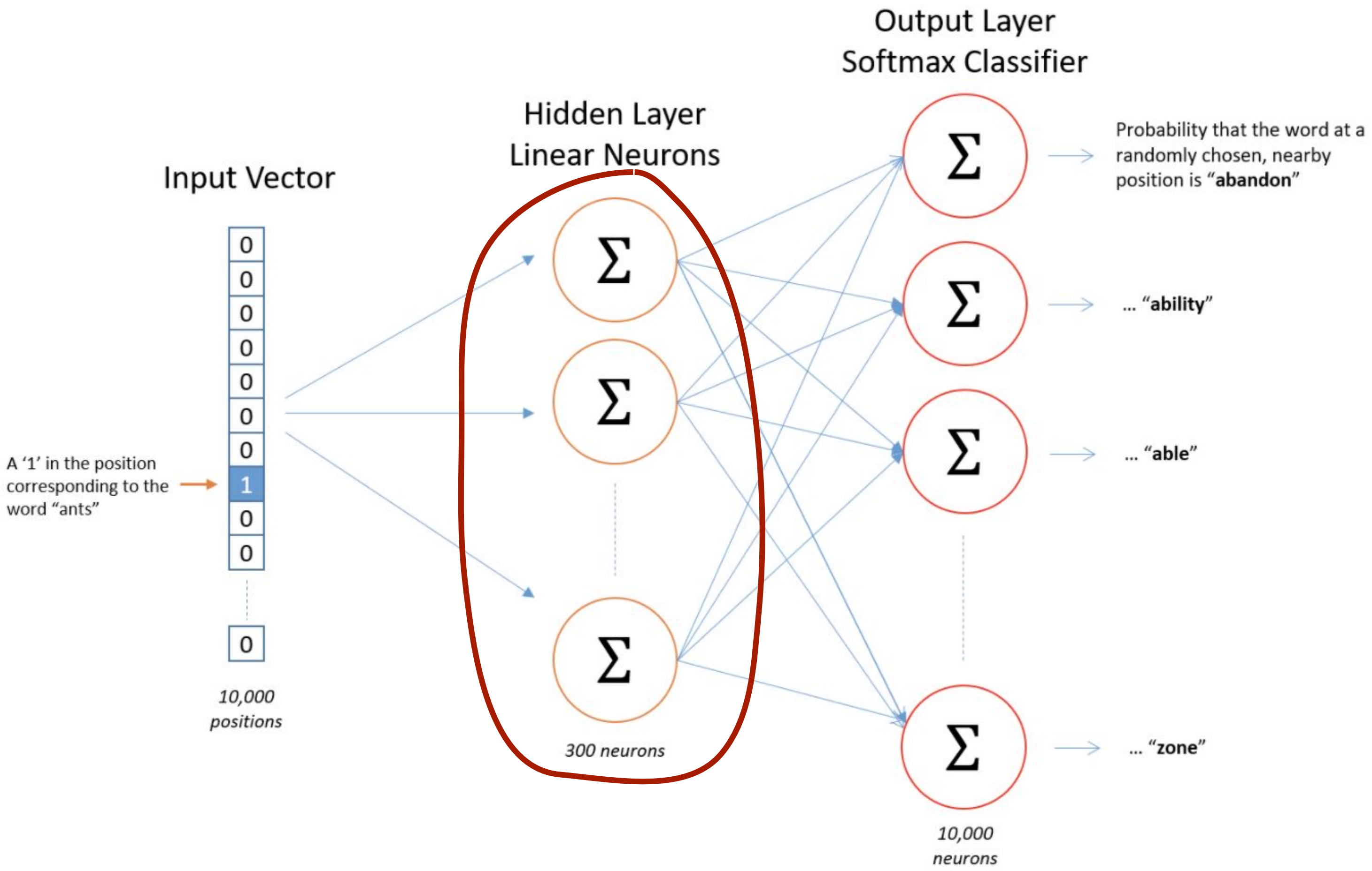
Backward Pass



Métodos de optimización:

- stochastic gradient descent
- negative sampling

link: https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"
10,000 neurons

posición de palabra x

vector look up table

vector de palabra x

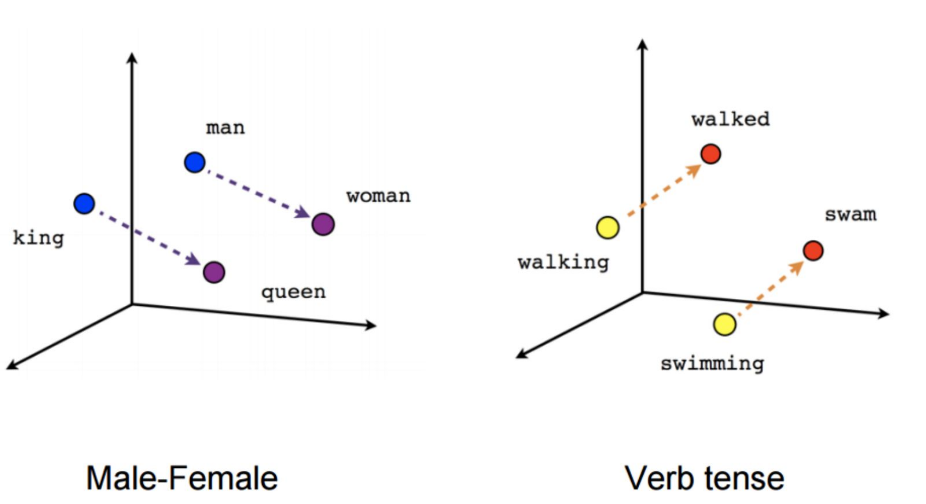
$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{pmatrix} 21 & 17 & 3 & 4 \\ 19 & 10 & 1 & 5 \\ 5 & 4 & 18 & 1 \\ 11 & 18 & 25 & 12 \\ 2 & 4 & 6 & 10 \\ 23 & 5 & 8 & 4 \end{pmatrix} = \begin{bmatrix} 11 & 18 & 25 & 12 \end{bmatrix}$$

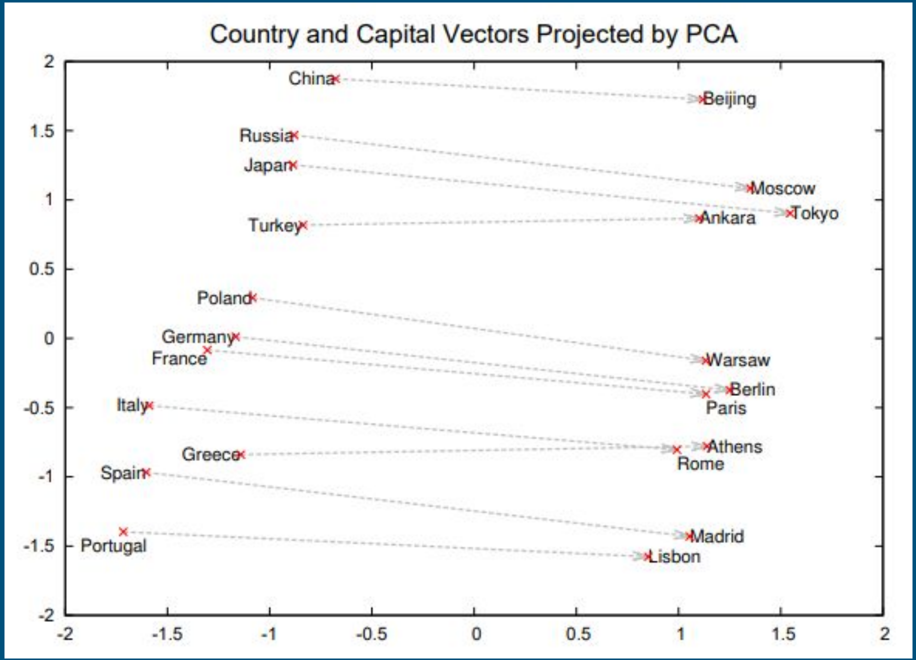
Propiedades aritméticas

`vector(Paris) – vector(France) + vector(Morocco) ~ vector(Rabat)`

$$\mathbf{u}_{\text{queen}} = \mathbf{u}_{\text{king}} - \mathbf{u}_{\text{man}} + \mathbf{u}_{\text{woman}}$$

Propiedades





Mikolov et al, 2013

De word2vec a hoy...

State-of the art: Modelos basados en transformers

Transformer: arquitectura conceptualmente sencilla que utiliza ***self-attention*** para aprender las dependencias entre input y output

- GPT, GPT-2 : Standard language modeling objective
  - Los estudiantes están muy _______ (atentos, interesados, aburridos, etc)
- BERT : Masked Language Modeling y Next sentence prediction
  - Los _______ están muy atentos.
  - Los estudiantes están muy atentos.
    - A el maestro le da mucho gusto **TRUE**
    - Mi perro tiene hambre **FALSE**

1. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
2. GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever.
3. GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever.
4. Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov.
5. XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le.
6. XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.
7. RoBERTa (from Facebook), released together with the paper a Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.
8. DistilBERT (from HuggingFace) released together with the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter by Victor Sanh, Lysandre Debut, and Thomas Wolf. The same method has been applied to compress GPT2 into DistilGPT2.
9. CTRL (from Salesforce), released together with the paper CTRL: A Conditional Transformer Language Model for Controllable Generation by Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher.
10. CamemBERT (from FAIR, Inria, Sorbonne Université) released together with the paper CamemBERT: a Tasty French Language Model by Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suarez, Yoann Dupont, Laurent Romary, Eric Villemonte de la Clergerie, Djame Seddah, and Benoît Sagot.
11. ALBERT (from Google Research), released together with the paper ALBERT: A Lite BERT for Self-supervised Learning of Language Representations by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.
12. T5 (from Google) released with the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer by Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu.
13. XLM-RoBERTa (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov.
14. MMBT (from Facebook), released together with the paper a Supervised Multimodal Bitransformers for Classifying Images and Text by Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine.
15. FlauBERT (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab.
16. BART (from Facebook) released with the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer.
17. ELECTRA (from Google Research/Stanford University) released with the paper ELECTRA: Pre-training text encoders as discriminators rather than generators by Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning.

1. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
2. GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever.
3. GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever.
4. Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov.
5. XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le.
6. XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.

7

8

9

https://huggingface.co/transformers/

10

11

Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.

12. T5 (from Google) released with the paper Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer by Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu.
13. XLM-RoBERTa (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov.
14. MMBT (from Facebook), released together with the paper a Supervised Multimodal Bitransformers for Classifying Images and Text by Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine.
15. FlauBERT (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab.
16. BART (from Facebook) released with the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer.
17. ELECTRA (from Google Research/Stanford University) released with the paper ELECTRA: Pre-training text encoders as discriminators rather than generators by Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning.

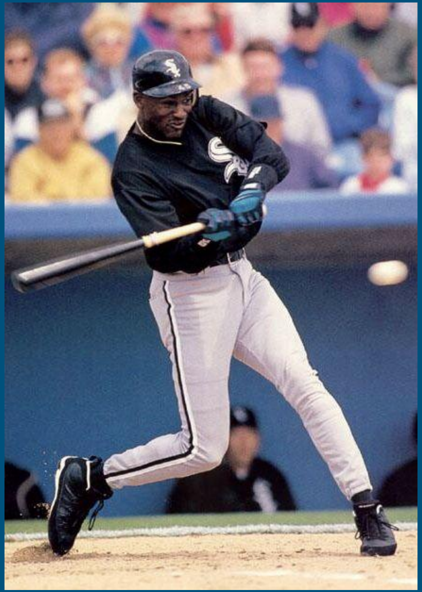
Utilizar modelos pre-entrenados es estándar en NLP

Transfer Learning

Modelo Pre-entrenado

finetuning

Performance en nuevo objetivo

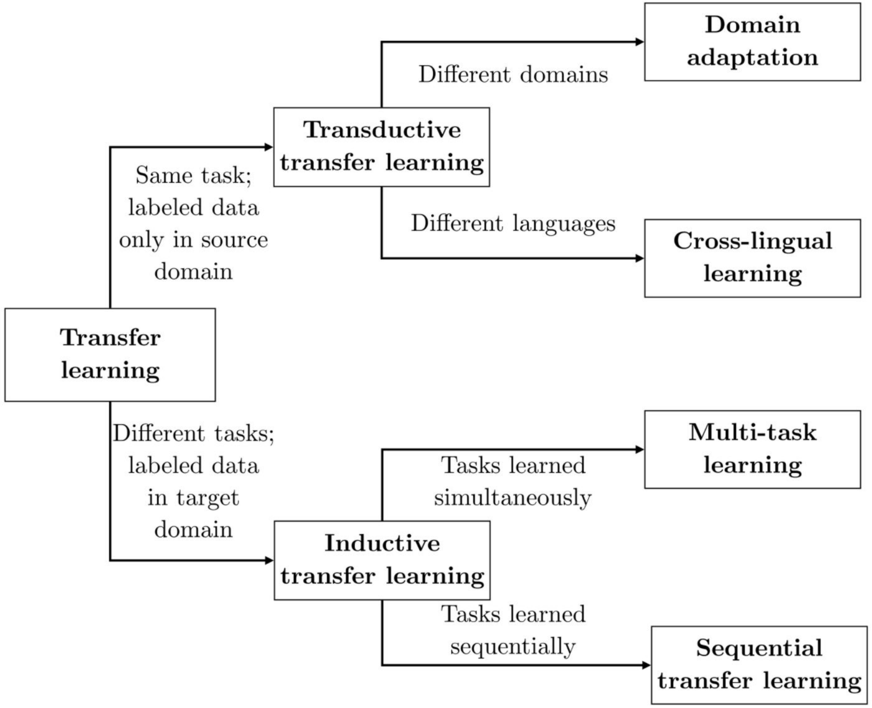


Transfer Learning

Utilizamos un modelo pre-entrenado para cierto objetivo como punto de partida hacia otro objetivo.

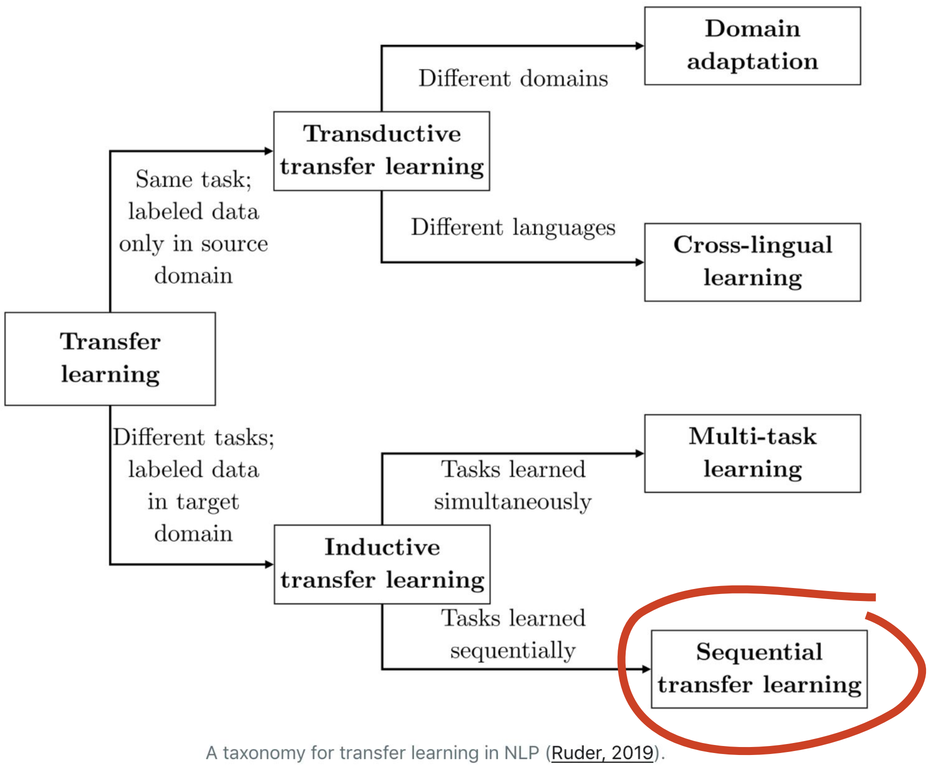
- reduce la necesidad de datos anotados
- transfer learning para aprendizaje multilingüe
  - alinear representaciones contextuales
  - compartir vocabulario y pre-entrenar un modelo en muchos idiomas

Transfer Learning



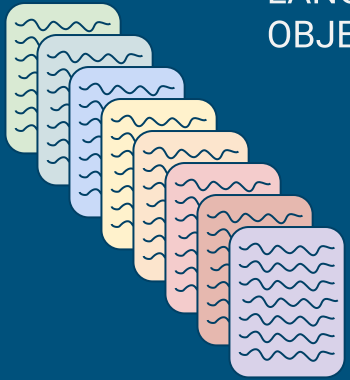
A taxonomy for transfer learning in NLP (Ruder, 2019).

Transfer Learning



A taxonomy for transfer learning in NLP (Ruder, 2019).

Transfer Learning



PRE-TRAINING CON VARIANTES DEL LANGUAGE MODELING OBJECTIVE

word2vec
GloVe
ELMo
ULMFiT
GPT
BERT

ADAPTACIÓN

Clasificación	Seq2seq
Clustering	etc

Resumen

- Representar texto de manera vectorial es difícil
  - Predicción de contextos ha tenido éxito en NLP
  - Métodos básicos como baselines
- Modelo de lenguaje es una tarea difícil que requiere información semántica, sintáctica y conocimiento del mundo
  - Podemos lograr buenos resultados si entrenamos modelos gigantes con corpus gigantes
- Pre-entrenar modelos utilizando este objetivo nos da una buena base para adaptar a otros objetivos.

Ejercicios

Vectores basados en predicción de contextos

Calcular $P(w_{t+j}|w_t;\theta)$

1. Por cada palabra ,utilizamos dos vectores
   a. vector *v* para la palabra central *c*
   b. vector *u* para la palabra de contexto *o*

Por una palabra central *c* y su palabra de contexto *o*

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{W \in V} \exp(u_W^T v_c)}$$

Vectores basados en predicción de contextos

se convierte en positivo

$$P(o|c) = \frac{\boxed{\exp}(\boxed{u_o^T v_c})}{\boxed{\sum_{W \in V} \exp(u_W^T v_c)}}$$

dot product para calcular la similitud de los vectores

Normalizamos utilizando TODO el vocabulario para asignar una distribución de probabilidades

Esta es la función softmax!!

Vectores basados en predicción de contextos

Entrenar nuestro modelo implica ajustar TODOS los parámetros θ para minimizar pérdidas ($J(\theta)$)

Representamos nuestros parámetros usando vectores y optimizamos usando el gradiente

$$\theta = \begin{bmatrix} v_{\mathrm{arqueologicos}} \\ v_{\mathrm{atendiendo}} \\ \cdots \\ \cdots \\ u_{\mathrm{arqueologicos}} \\ u_{\mathrm{atendiendo}} \end{bmatrix}$$

Vectores de TF-IDF

Frecuencia de Término (TF)

$TF(t,d)$ = número de veces que el término t ocurre en el documento d

Frecuencia Inversa de Documento (IDF)

$IDF(t,D)$ = $\log\left(\frac{D}{\{d \in D : t \in d\}}\right)$

D = *el número total de documentos*

$\{d \in D : t \in d\}$ = número de documentos que contienen t

Vectores de TF-IDF

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

	béisbol	comer	estefania	gusta	jugar	pizza	sushi	Y
d_1	0	2	1	1	0	1	1	comida
d_2	1	0	1	1	1	0	0	deporte

$$TFIDF(comer,\ d_1, D) = TF(comer, d_1) \times IDF(comer,\ D)$$

$$TFIDF(comer,\ d_1, D) = 2 * \log\left(\frac{2}{1}\right)$$

$$TFIDF(comer,\ d_1, D) = 0.6$$

Vectores de TF-IDF

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

	béisbol	comer	estefania	gusta	jugar	pizza	sushi	Y
d_1	0	2	1	1	0	1	1	comida
d_2	1	0	1	1	1	0	0	deporte

$$TFIDF(gusta,\, d_1, D) \; = \; TF(gusta, d_1) \times IDF(gusta,\, D)$$

$$TFIDF(gusta,\, d_1, D) = 1 * \log(1)$$

$$TFIDF(gusta,\, d_1, D) = 0$$

Vectores de TF-IDF

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

	béisbol	comer	estefania	gusta	jugar	pizza	sushi	Y
d_1	0	0.6	0	0	0	0.3	0.3	comida
d_2	0.3	0	0	0	0.3	0	0	deporte

$$TFIDF(gusta,\ d_1, D) = TF(gusta, d_1) \times IDF(gusta,\ D)$$

$$TFIDF(gusta,\ d_1, D) = 1 * \log(1)$$

$$TFIDF(gusta,\ d_1, D) = 0$$