Text Mining


Universidad
de Valparaíso
CHILE


ESCUELA DE INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE VALPARAÍSO

Unidad 2 - Representación de Texto

➔ Contenidos

- Modelo espacio vectorial
- **Representación distribuida**
- word embeddings

Unidad 2 - Representación de Texto

Representación distribuida

- Hemos visto:
  - palabras son la unidad lingüística menor, que tiene significado independiente.
  - Vectores one-hot, no permiten capturar estructura gramatical ni semántica
- Propuesta:
  - Harris y Firth, en 1954 y 1957: “la semántica de una palabra viene determinada por su contexto”.
  - “palabras con contextos similares tienen significados similares”.
  - Cuando se tenga mayor información del contexto ➜ lleva a tener una representación distribuida, que describe la semántica de las palabras.
  - Palabras con semántica similar, están cercanas en el espacio vectorial.

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida

- En Modelo Espacio Vectorial
  - Se utiliza ventana fija
  - n-gramas son difíciles de utilizar, número posible de n-gramas aumenta de forma exponencial, a medida que n crezca (maldición de la dimensionalidad, escasez de datos).
- Modelos de Redes Neuronales
  - permiten modelar contextos más complejos, por lo que la representación distribuida de palabras consideran información sintáctica y semántica
  - Datos de entrenamiento:

$$D = \{w_{i_1}^{m_i}\}_{i=1}^{M}$$

m_i :representa el número de palabras que contiene la i-ésima sentencia

$w_{i_1}^{m_i}$:representa la secuencia de palabras en la sentencia $w_{i_1}, w_{i_2}, \ldots, w_{m_i}$

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3l

Unidad 2 - Representación de Texto

Representación distribuida

- Cada palabra es mapeado a un vector distribuido d-dimensional - word embedding
- El vocabulario V, corresponde a word embeddings

$$\boldsymbol{L} \in \mathbb{R}^{|V| \times d}$$

  - Red neuronal es optimizar la matriz de word embedding ***L***
  - Idea: aprender representación precisa para cada palabra

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida - Neural Network Language Model

- Se tiene una sentencia, conformada por *m* palabras $w_1, w_2, ..., w_m$, la ocurrencia está dado por

$$p(w_1 w_2 \cdots w_m) = p(w_1)p(w_2|w_1)\cdots p(w_i|w_1, \cdots, w_{i-1}) \cdots p(w_m|w_1, \cdots, w_{m-1})$$

- También, se puede utilizar **método de máxima verosimilitud** para calcular la probabilidad condicional de que una palabra ocurra dado el contexto previo.

$$p(w_i|w_1, \cdots, w_{i-1}) = \frac{\text{count}(w_1, \cdots, w_i)}{\text{count}(w_1, \cdots, w_{i-1})}$$

  - A medida aumenta el número de palabras (i mayor) que se consideran en la secuencia, la estimación de probabilidad se vuelve menos precisa (menos probable es que aparezca la frase $w_1, w_2, ..., w_i$)
  - Solución: utilizar lenguaje de n-gramas, limitando a las n-1 palabras.

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida - Neural Network Language Model

- Suponer que la probabilidad de la palabra actual está dada solo por las (n-1) palabras $$p(w_i|w_1,\cdots,w_{i-1}) \approx p(w_i|w_{i-n+1},\cdots,w_{i-1})$$
  - n = 1 ➜ corresponde a unigrama, y la palabra es independiente
  - n = 2 ➜ bigrama, probabilidad considera solo la palabra anterior
  - n = 3, 4, 5 ➜ valores más utilizados
- Métodos de estimación de probabilidad, donde se busca la coincidencia de frases, siguen teniendo el problema de escasez de datos y no captan la similitud semántica,
  - Por ejemplo: p(boring | very) y p(uninteresting | very), pueden tener diferentes apariciones en el corpus, lo que da lugar a diferencia en las probabilidades.

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

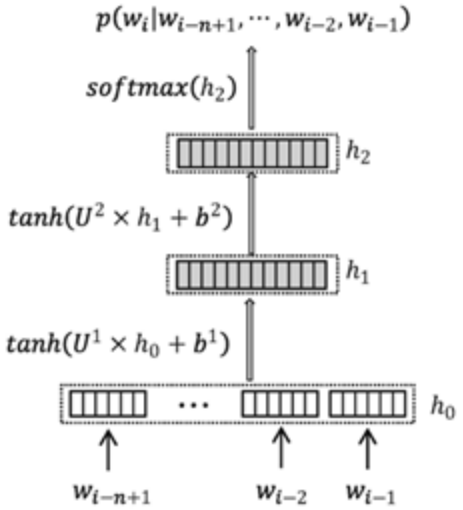
Representación distribuida

Neural Network Language Model

- Feed-forward neural network (FNN)
  - Lectura recomendada: A neural probabilistic language model.
  - Idea: convertir cada palabra en un vector de valores reales de baja dimensionalidad (word embedding)
  - Calcular la probabilidad de que una palabra ocurra dado su contexto previo

$$p(w_i|w_{i-n+1}, \cdots, w_{i-1})$$

  - Esta técnica se refiere a los modelos n-grama en el espacio vectorial continuo



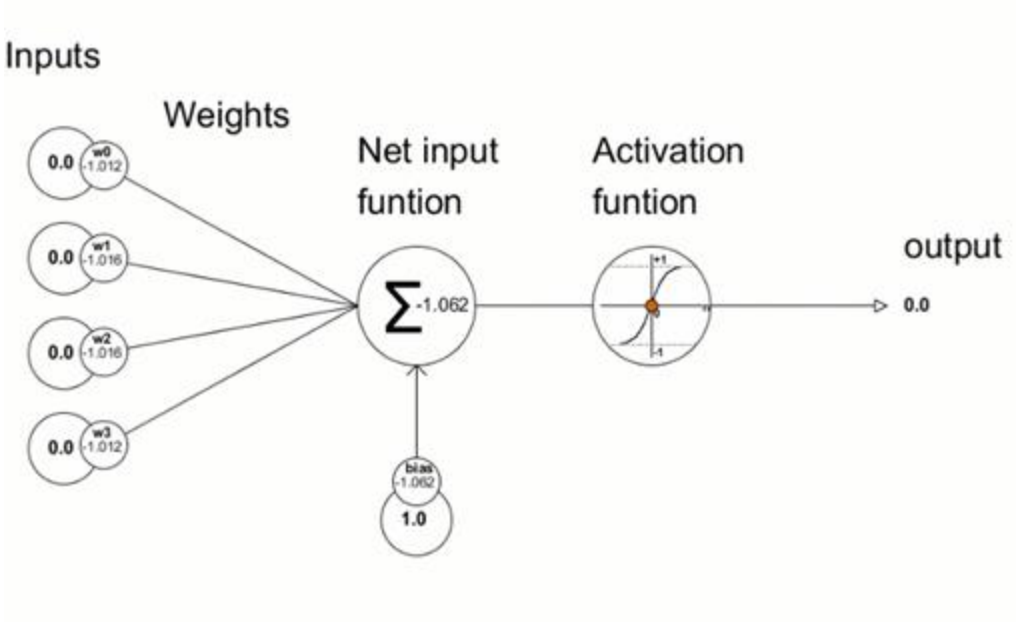
Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). *A neural probabilistic language model*. Journal of Machine Learning Research, 3, 1137–1155

Unidad 2 - Representación de Texto

Representación distribuida

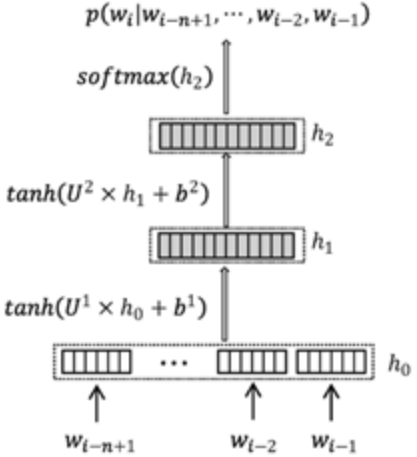
Neural Network Language Model



Unidad 2 - Representación de Texto

Representación distribuida

Neural Network Language Model



(n-1) palabras son mapeadas a un embedding, y luego se concatenan para obtener h_0

$$\boldsymbol{h}_0 = [\boldsymbol{e}(w_{i-n+1}); \cdots ; \boldsymbol{e}(w_{i-1})]$$

$$\boldsymbol{e}(w_{i-1}) \in \mathbb{R}^d$$

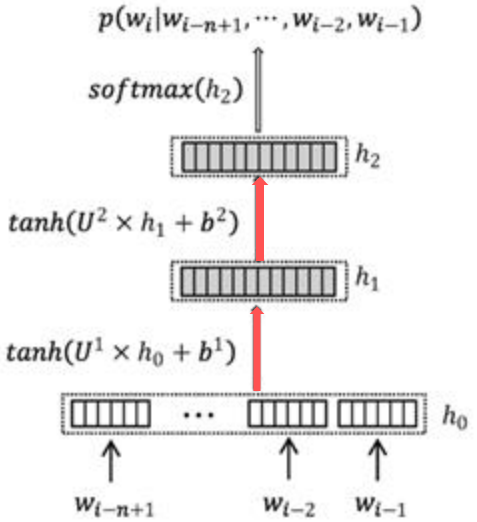
representa el embedding para la palabra $w_{(i-1)}$, que se obtiene de la matriz ***L***

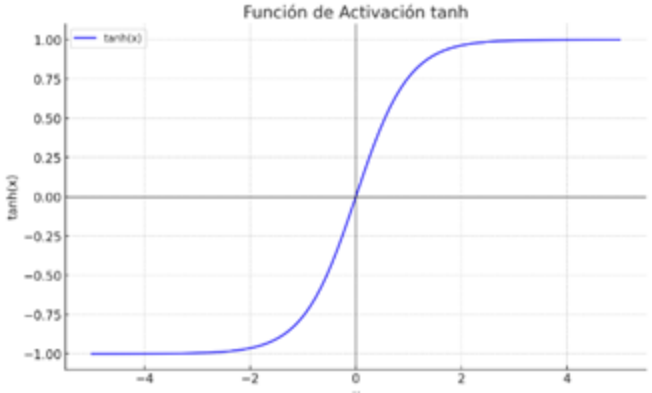
Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3
Imagen creada por chatGPT

Unidad 2 - Representación de Texto

Representación distribuida

Neural Network Language Model





$$h_0 = [e(w_{i-n+1}); \cdots ; e(w_{i-1})]$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

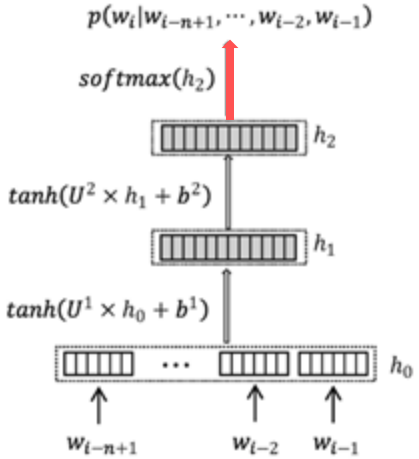
- Rango de valores [-1,1]
- Centrada en 0
- Valores grandes de x, la función aproxima a 1.
- Valores negativos grandes de x, se aproxima a -1.
- Para valores de x cercanos a 0, la función tiene una pendiente más pronunciada y se acerca a 0.

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3
Imagen creada por chatGPT

Unidad 2 - Representación de Texto

Representación distribuida

Neural Network Language Model



softmax: función de activación

$$\text{softmax}(h_2^{(i)}) = \frac{e^{h_2^{(i)}}}{\sum_{j=1}^{n} e^{h_2^{(j)}}}$$

Asegura que todos los valores de salida sean positivos.

Suma de todas las probabilidad será igual a 1 (salidas normalizadas).

Permite obtener una probabilidad sobre diferentes clases, y seleccionar aquella que tenga mayor probabilidad.

$$h_2 = [h_2^{(1)}, h_2^{(2)}, \ldots, h_2^{(n)}]$$

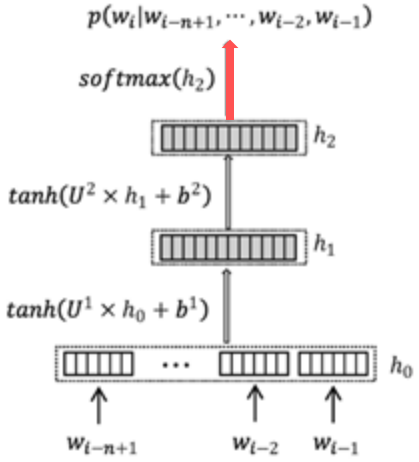
Vector de activación logits (valores de salida) de la última capa, antes de la salida. La activación convierte los valores en un formato adecuado para la tarea que se está resolviendo.

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida

Neural Network Language Model



softmax: función de activación

$$\text{softmax}(h_2^{(i)}) = \frac{e^{h_2^{(i)}}}{\sum_{j=1}^{n} e^{h_2^{(j)}}} \Rightarrow p(w_i|w_{i-n+1},\cdots,w_{i-1}) = \frac{\exp\{\boldsymbol{h}_2 \cdot \boldsymbol{e}(w_i)\}}{\sum_{k=1}^{|V|} \exp\{\boldsymbol{h}_2 \cdot \boldsymbol{e}(w_k)\}}$$

Distribución de probabilidad de cada palabra en V, calculada por la función softmax

Matrices de peso: U1, U2, b1, b2 y L, parámetros (θ) de la red neuronal que ajustan su valor en el entrenamiento. Objetivo optimizar θ

$$\Rightarrow \theta^* = \underset{\theta}{\text{argmax}} \sum_{m=1}^{M} \log p(w_{i_1}^{m_i})$$

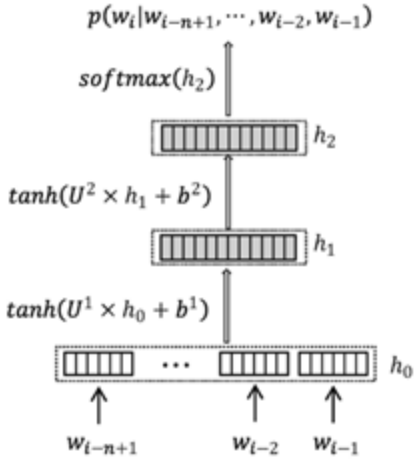
M: tamaño de los datos de entrenamiento

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida

Neural Network Language Model



La función de activación en la capa de salida, puede cambiar según el problema.
Algunas recomendaciones:

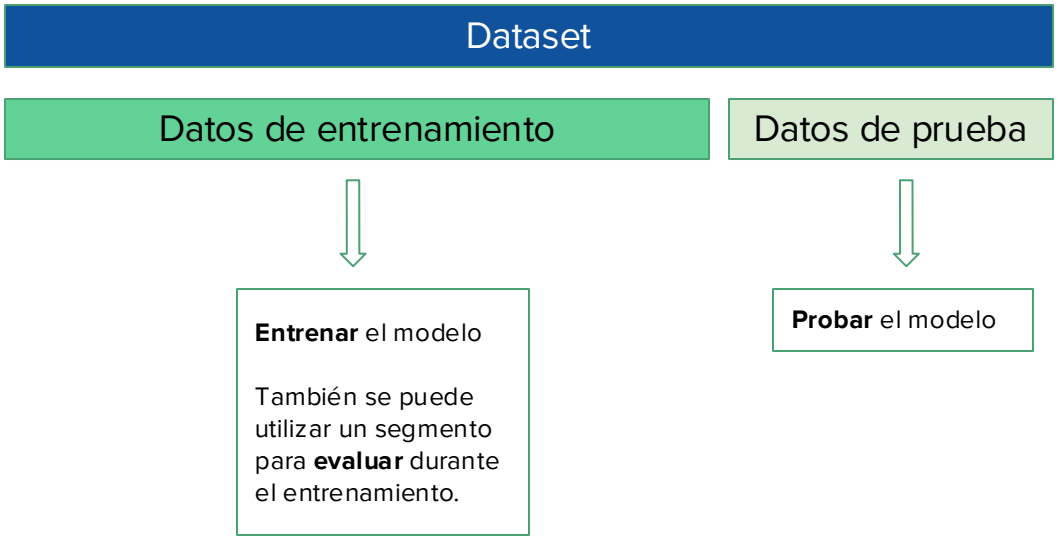
- clasificación multiclase: softmax
- clasificación binaria: sigmoide
- regresión: activación lineal

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Antes de continuar:

Para entrenamiento y pruebas



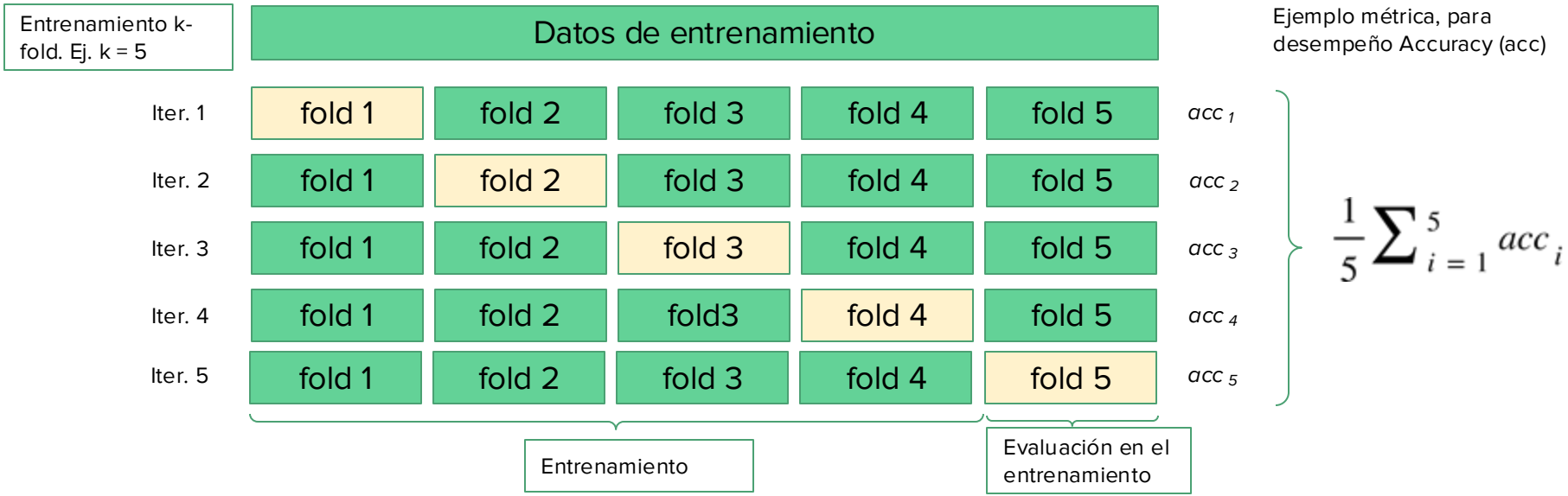
Ejemplos de distribución de los datos, entrenamiento/pruebas ejemplo:

- 70/30
- 80/20

Unidad 2 - Representación de Texto

Antes de continuar:

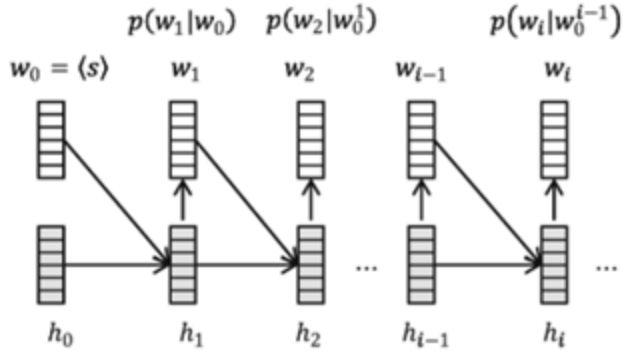
Para entrenamiento y pruebas



Unidad 2 - Representación de Texto

Representación distribuida

Recurrent neural network model (RNN)



- FNN captura el contexto en una ventana.
- RNN: utiliza la información histórica (w_1, W_2 , ... , w_{i-1}) para predecir la probabilidad de la actual palabra (w_i)

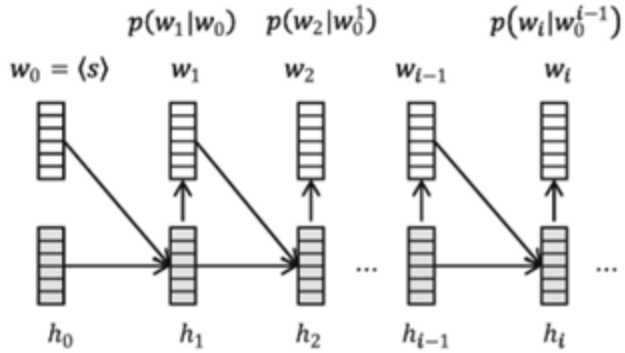
$$p(w_i|w_1, \cdots, w_{i-1})$$

Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida

Recurrent neural network model (RNN)



$$\boldsymbol{h}_i = f(\boldsymbol{W} \times \boldsymbol{e}(w_{i-1}) + \boldsymbol{U} \times \boldsymbol{h}_{i-1} + \boldsymbol{b})$$

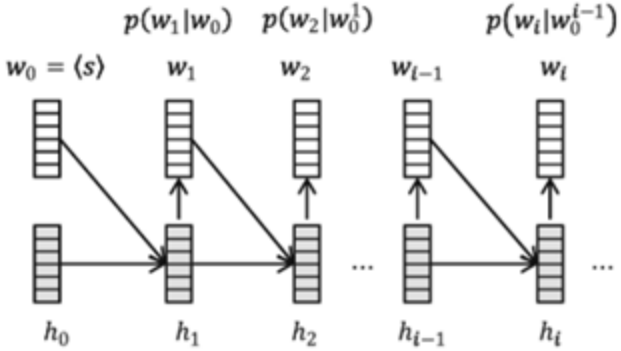
time step: 0 a (n-1)

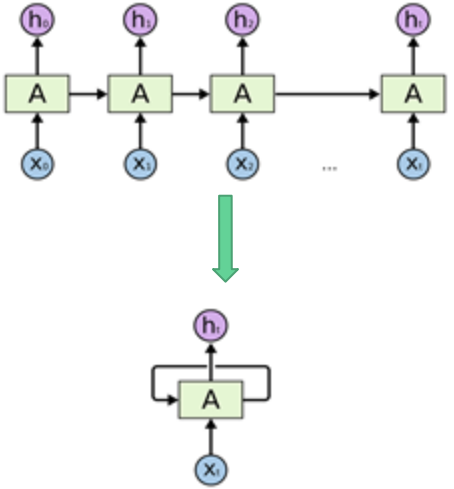
Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3

Unidad 2 - Representación de Texto

Representación distribuida

Recurrent neural network model (RNN)





LSTM
GRU

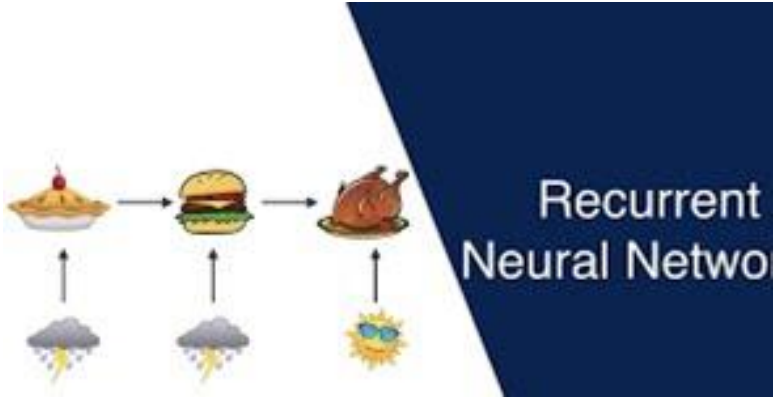
Material basado en: https://link.springer.com/chapter/10.1007/978-981-16-0100-2_3
https://www.kaggle.com/code/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru

Unidad 2 - Representación de Texto

Representación distribuida

Recurrent neural network model (RNN)

Click en la imágen para ver video.



Recomendado:
https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2
https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd
https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235