

# Natural Language Processing with Disaster Tweets

*David Muñoz*  
*Paco Sangorrín*


# El reto

El reto es crear un modelo de aprendizaje automático que prediga qué tuits se refieren a catástrofes reales y cuáles no.

# El reto

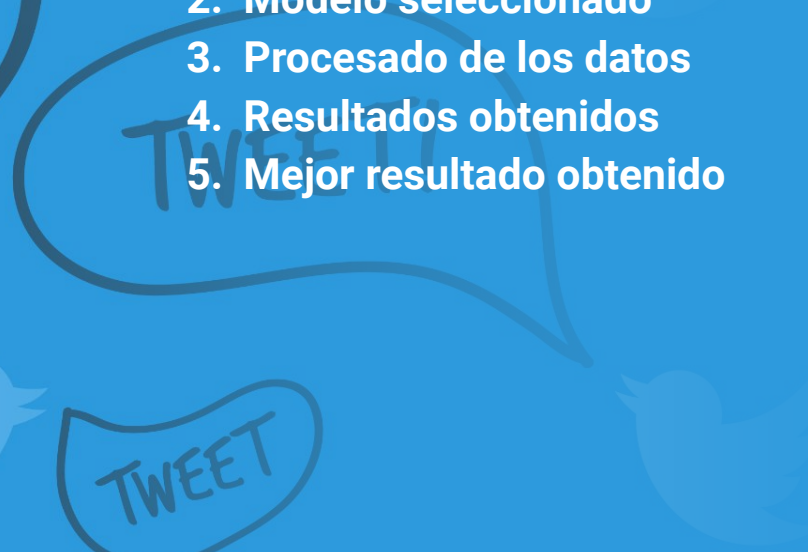
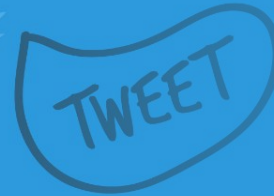
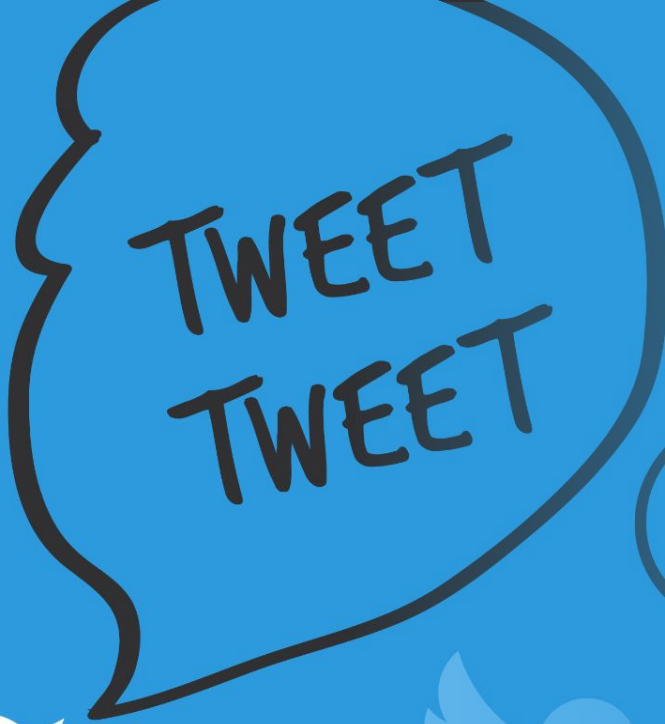
**Twitter** se ha convertido en un **importante canal de comunicación** en tiempos de emergencia. Por ello, cada vez más organismos están interesados en vigilar Twitter de forma programada. Sin embargo, **no siempre está claro si las palabras empleadas por una persona anuncian realmente una catástrofe** ya que en ocasiones se usan palabras de forma metafórica o sarcástica.

De este modo, nos enfrentamos a un **problema de procesamiento de lenguaje natural (NLP)** en el que debemos crear un **modelo de aprendizaje automático que prediga qué tuits son sobre catástrofes reales y cuáles no.**



# Índice

1. Exploración de variables
2. Modelo seleccionado
3. Procesado de los datos
4. Resultados obtenidos
5. Mejor resultado obtenido



# Exploración de variables

# Exploración de variables: Descripción del dataset

El dataset está compuesto por **4 variables explicativas**:

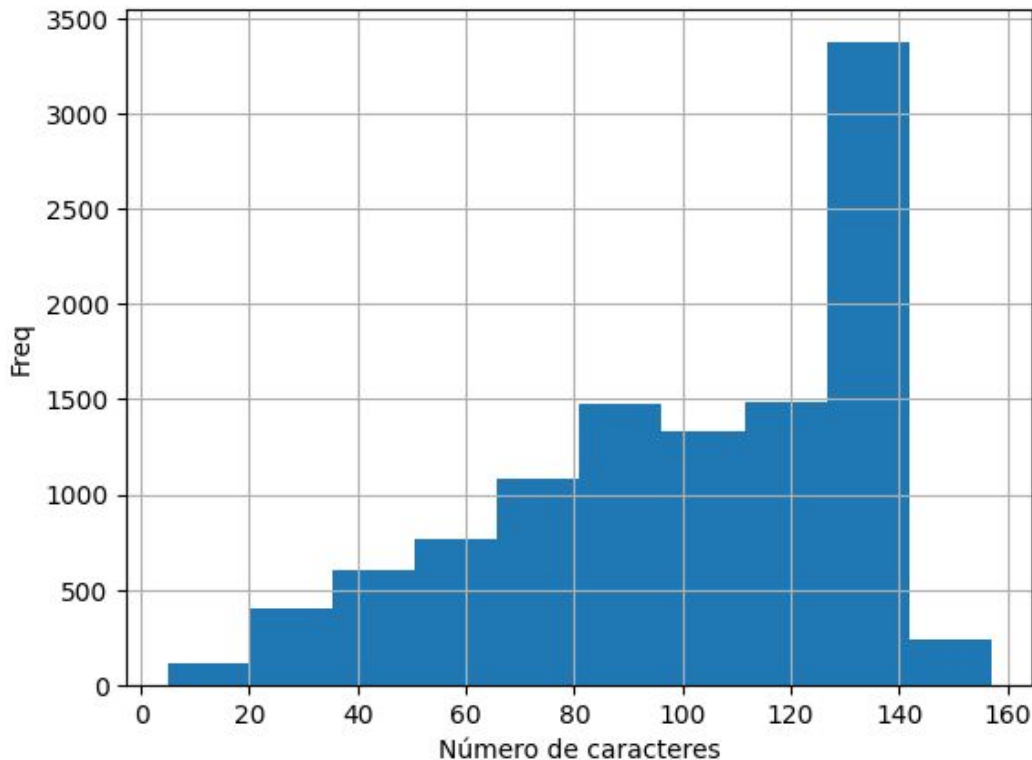
- id: Identificador del registro. Variable descartada.
- keyword: Palabra clave del tweet
- location: Localización de donde se mandó el tweet.
- text: Texto del tweet

La **variable objetivo** (target) es binaria.

- 0: El tweet no trata sobre un desastre real
- 1: El tweet sí trata sobre un desastre real



# Exploración de variables: Tweets



Como es texto no podemos hacer un análisis al uso, pero podemos ver como se distribuye aunque sea la longitud de los textos.

```
count    10876.000000
mean      101.358680
std       33.840687
min        5.000000
25%       78.000000
50%      108.000000
75%      134.000000
max      157.000000
Name: text, dtype: float64
```

# Exploración de variables: Keywords y Location

Uniendo train y test (10876 casos):

- 3638 no tienen localización. Rellenamos con “No location”
- 87 no tienen keyword. Rellenamos con “No keyword”





# Exploración de variables: Target

La variable respuesta es binaria.

- 0: 4342 tweets no trata sobre un desastre real
- 1: 3271 tweets sí tratan sobre desastres naturales





# Modelo seleccionado

# Modelo seleccionado: BERT

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin   Ming-Wei Chang   Kenton Lee   Kristina Toutanova  
Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

### Abstract

We introduce a new language representation model called **BERT**, which stands for **Bidirectional Encoder Representations from Transformers**. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute im-

provement). There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-

## BERT (Bidirectional Encoder Representations from Transformers):

- Modelo de procesamiento lenguaje natural desarrollado en 2018 por Google.
- Es el primo hermano de los GPT (Generative Pre-trained Transformer)
- Sintetizando mucho, tanto BERT como los GPT son dos modelos expertos en predecir la siguiente palabra más probable a un texto.

# Preprocesado de los datos

# Preprocesado de los datos: Pasos realizados

1. Limpiamos de caracteres extraños los tweets.
2. Tokenizamos las frases.
3. Creamos datasets de entrenamiento y validación con batch 32



# 1. Limpieza de texto

- Quitamos URLs
- Quitamos emoticonos
- Expandimos formas contraídas (I'm -> I am)
- Quitamos puntuaciones y caracteres no alfabéticos
- Quitamos stopwords que no aportan información (of, the, ...)



## 2. Tokenizamos las frases

Tokenizamos la frase 'This is an example for the MiDS class'

```
['this', 'is', 'an', 'example', 'for', 'the', 'mid', '##s', 'class']  
[2023, 2003, 2019, 2742, 2005, 1996, 3054, 2015, 2465]
```



## 2. Tokenizamos las frases

Tokenizamos la frase 'This is an example for the MiDS class'

['this', 'is', 'an', 'example', 'for', 'the', 'mid', '##s', 'class']

[2023, 2003, 2019, 2742, 2005, 1996, 3054, 2015, 2465]

Usamos un BERT

[101, 2023, 2003, 2019, 2742, 2005, 1996, 3054, 2015, 2465, 102]





## 2. Tokenizamos las frases

Tokenizamos la frase 'This is an example for the MiDS class'

```
['this', 'is', 'an', 'example', 'for', 'the', 'mid', '##s', 'class']  
[2023, 2003, 2019, 2742, 2005, 1996, 3054, 2015, 2465]
```

Usamos un BERT

```
[101, 2023, 2003, 2019, 2742, 2005, 1996, 3054, 2015, 2465, 102]
```

Con 50 tokens por frase es suficiente para no tener que hacer padding de más



# Resultados obtenidos

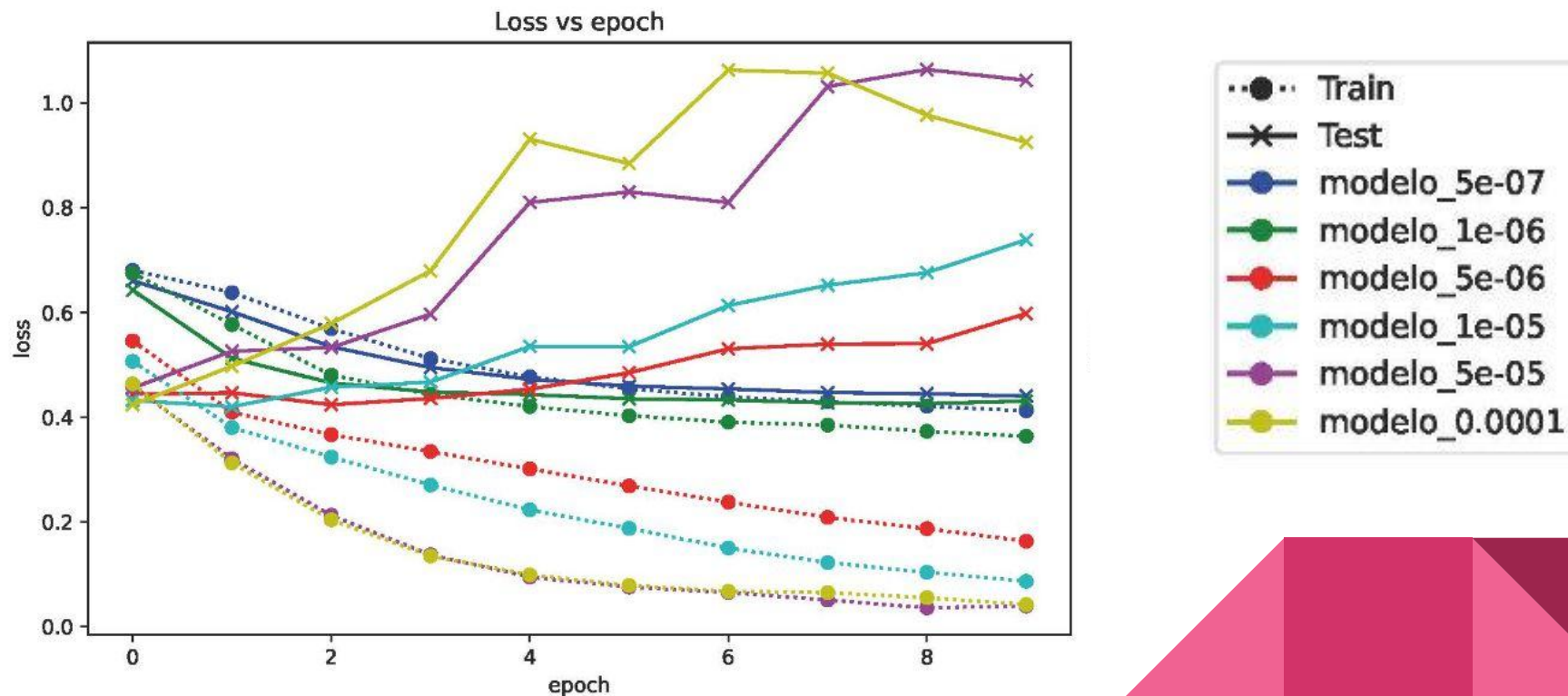
# Resultados obtenidos: Optimizador y loss

- Optimizador: Adam. Utilizaremos diferentes tasas de aprendizaje.
- Loss: SparseCategoricalCrossentropy.
- Metric: Accuracy

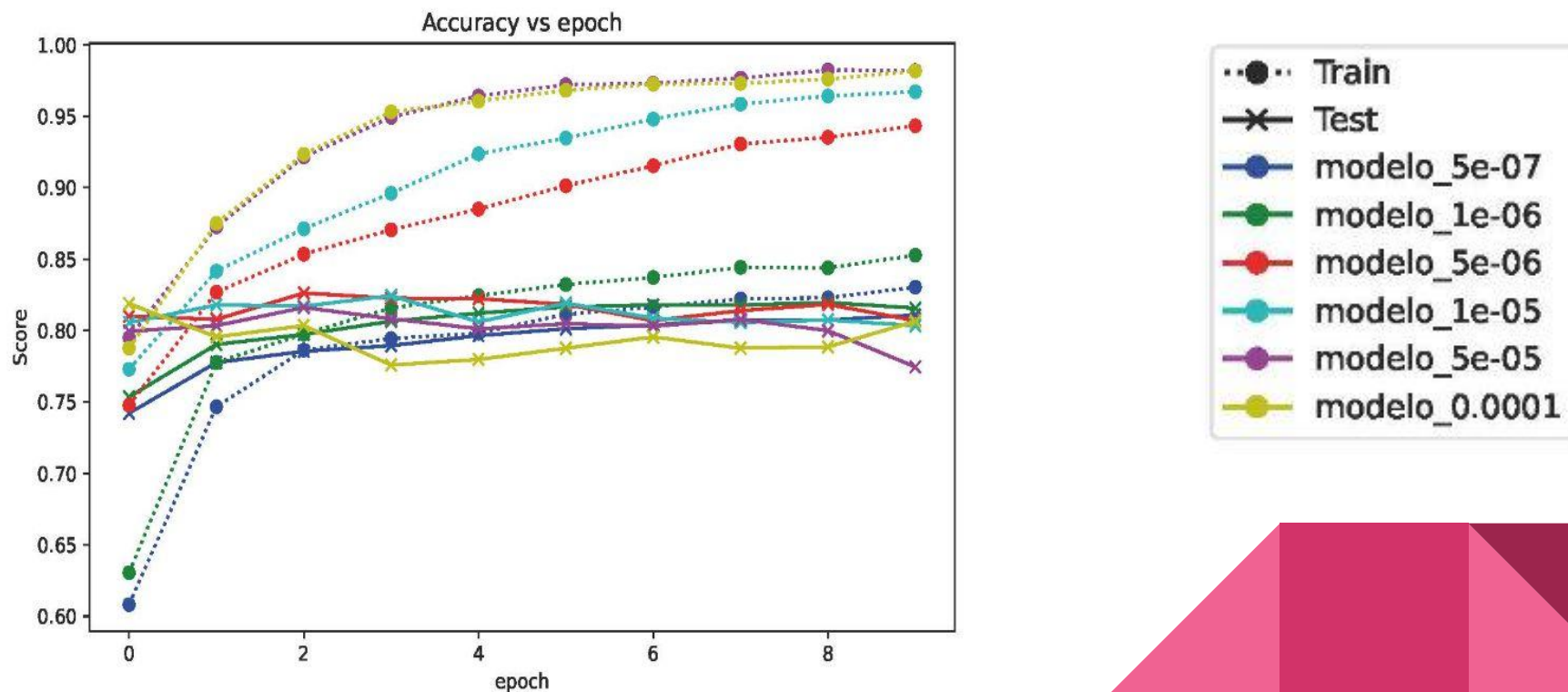
El uso de SparseCategoricalCrossentropy y Accuracy en lugar de BinaryCrossentropy y F1Score es por cómo está construido el modelo que da problemas



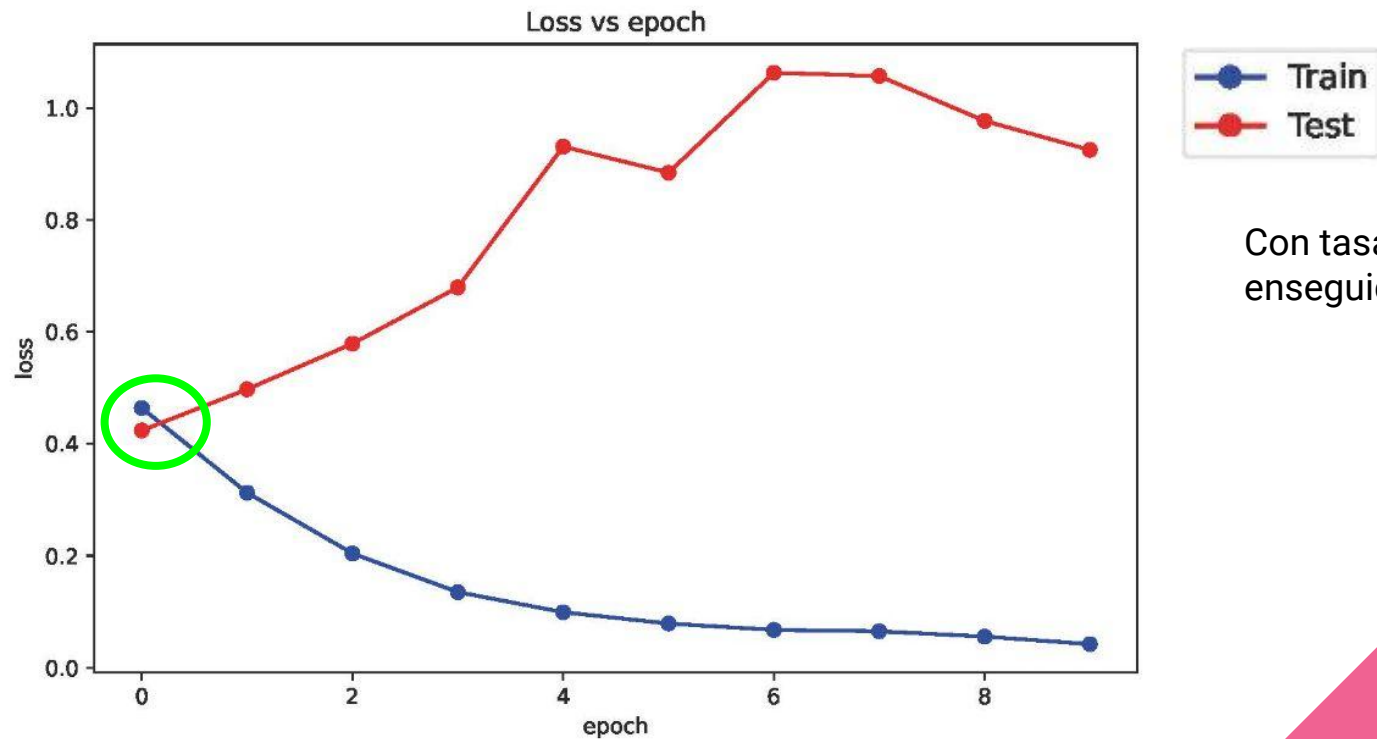
# Resultados obtenidos: Comparativa



# Resultados obtenidos: Comparativa

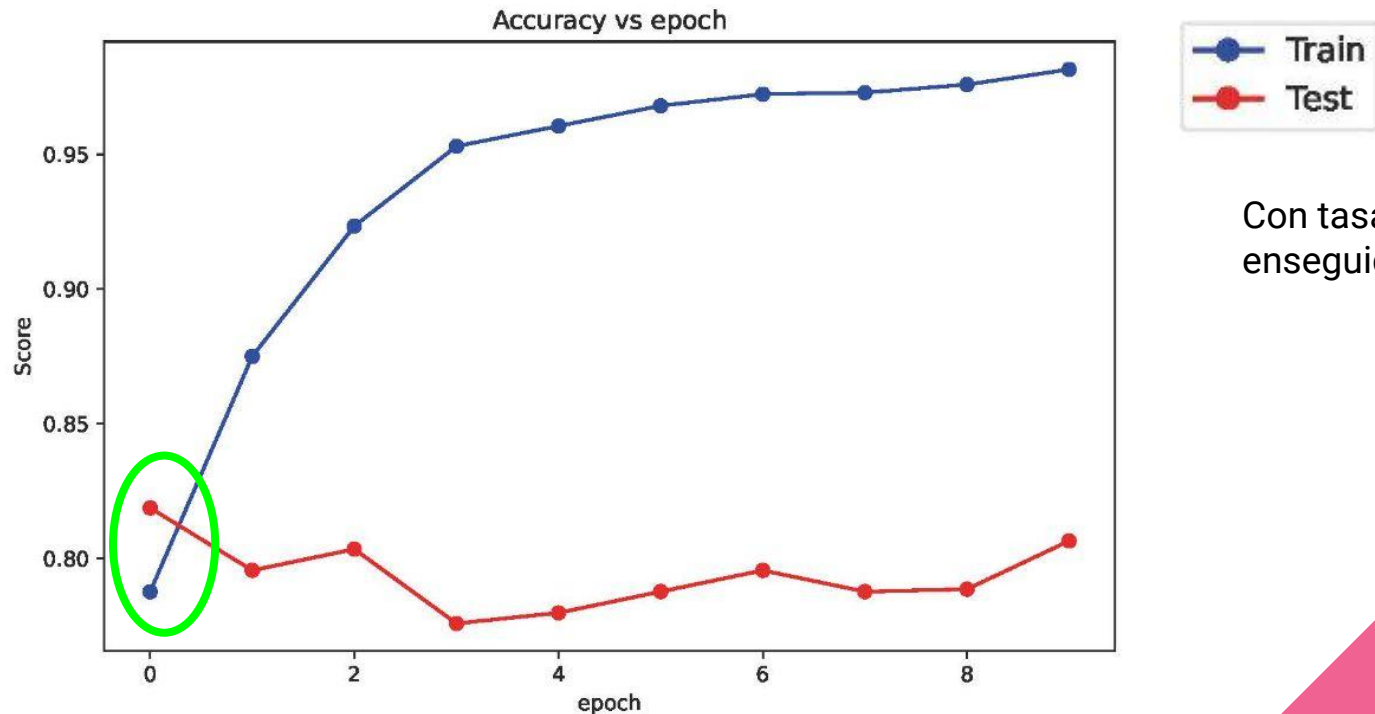


# Resultados obtenidos: Learning Rate $\rightarrow 0,0001$



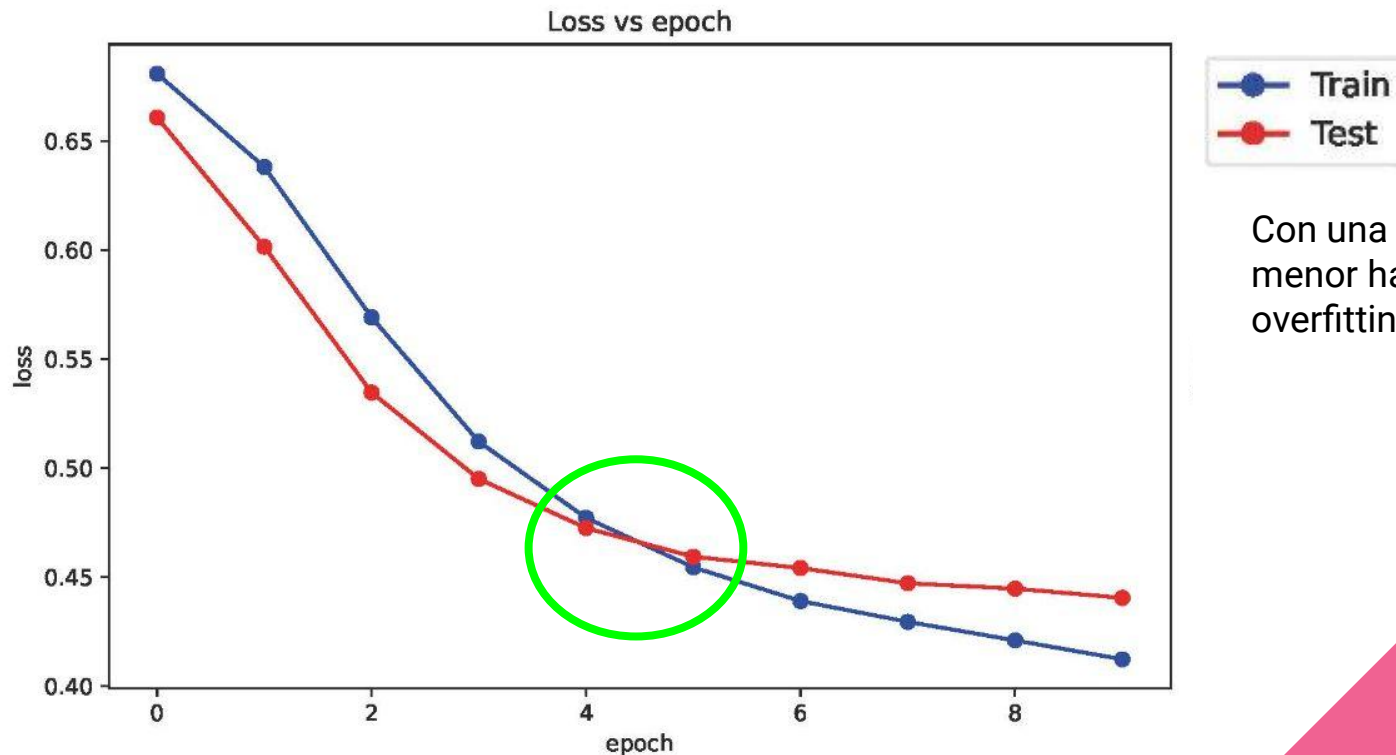
Con tasa de aprendizaje alta  
enseguida hay overfitting

# Resultados obtenidos: Learning Rate $\rightarrow 0,0001$



Con tasa de aprendizaje alta  
enseguida hay overfitting

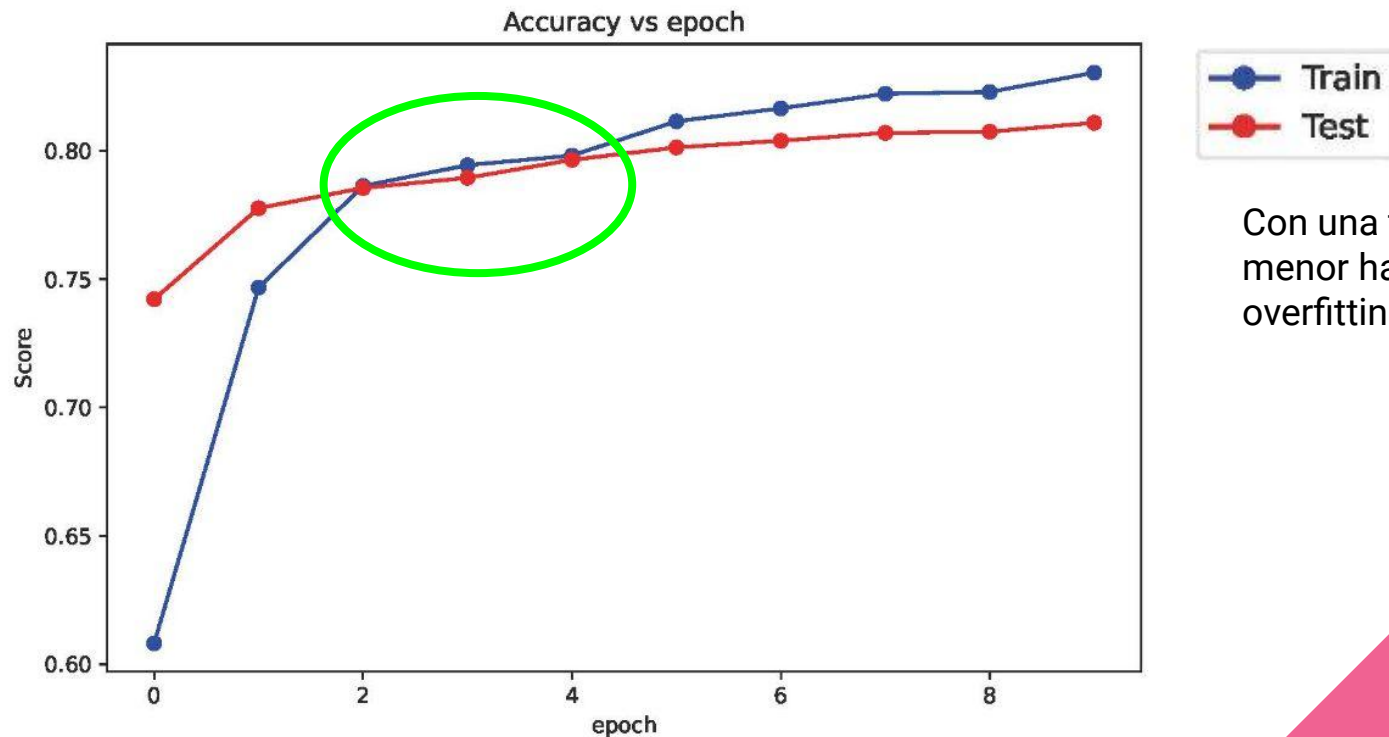
# Resultados obtenidos: Learning Rate $\rightarrow 5e^{-07}$



Con una tasa de aprendizaje menor hay más control sobre el overfitting y underfitting



# Resultados obtenidos: Learning Rate $\rightarrow 5e^{-07}$



Con una tasa de aprendizaje menor hay más control sobre el overfitting y underfitting



Mejor resultado obtenido

# Mejor resultado obtenido



**0.83052**

Entrenamos 2 épocas con tasa de aprendizaje de  $2 \cdot 10^{-5}$

Tabla

Otras pruebas que hicimos:

- Construir el modelo desde 0
- Añadir keyword y localización al inicio del tweet