

Natural Language Processing Workshop

Advanced Machine Learning AI

Prof. Juan Martin Maglione - Prof. Javier Blanco Cordero

Contact information - Prof. Juan Martin Maglione



E-mail: maglionejm@gmail.com
Mobile Argentina: +54 9 11 3691 7966
Mobile Spain: +34 691 82 69 82



Juan Martín Maglione holds a degree in Business Administration, with a Master's degree in Strategic and Technological Management from ITBA, a Master's Degree in Strategic and Technological Management from EOI (Spain). He has specialised in International Business Consultancy at Fontys University of Applied Sciences in Eindhoven, Holland, holds several specialisations in Big Data in ITBA, in Product Management, Agile Methodologies, Cloud Engineering and Deep Learning also in ITBA.

He is Global Product Transformation Manager for **Santander** (Madrid, Spain), Head of Artificial Intelligence and Technology for **SAI** (Amsterdam, Holland), co-Founder of **Fictionary**, a mobile game and of **Fon.ai**, an organisation that provides Artificial Intelligence and Analytics solutions for national and international public and private organisations.

He served as Regional Engagement Manager in **Cognitiva** for Argentina, Paraguay and Uruguay. He was the leader in the creation, construction and updating of the consultative methodologies used by Cognitiva, having coordinated the team of consultants throughout the region (Costa Rica, Colombia, Mexico, Chile and Argentina). He was part of the consulting firm **KPMG**, having participated in the creation of the practice of S & OP - Strategy and Operations.

Agenda

- NLP basics recap
- Guidelines presentation methodology:
 - Activity 1 - Tensorflow - Basics & Model Generation to recognize sentiment
 - Activity 2 - BERT - Natural Language Processing with Disaster Tweets

Tokenization

Is a representation of natural language that a computer can understand.

Tokenization is one of the first steps in NLP, and it's the task of **splitting a sequence of text into units with semantic meaning**.

These units are called tokens, and the difficulty in tokenization lies on how to get the ideal split so that all the tokens in the text have the correct meaning, and there are no left out tokens.

Tokenization in NLP



Tokenization - Word level

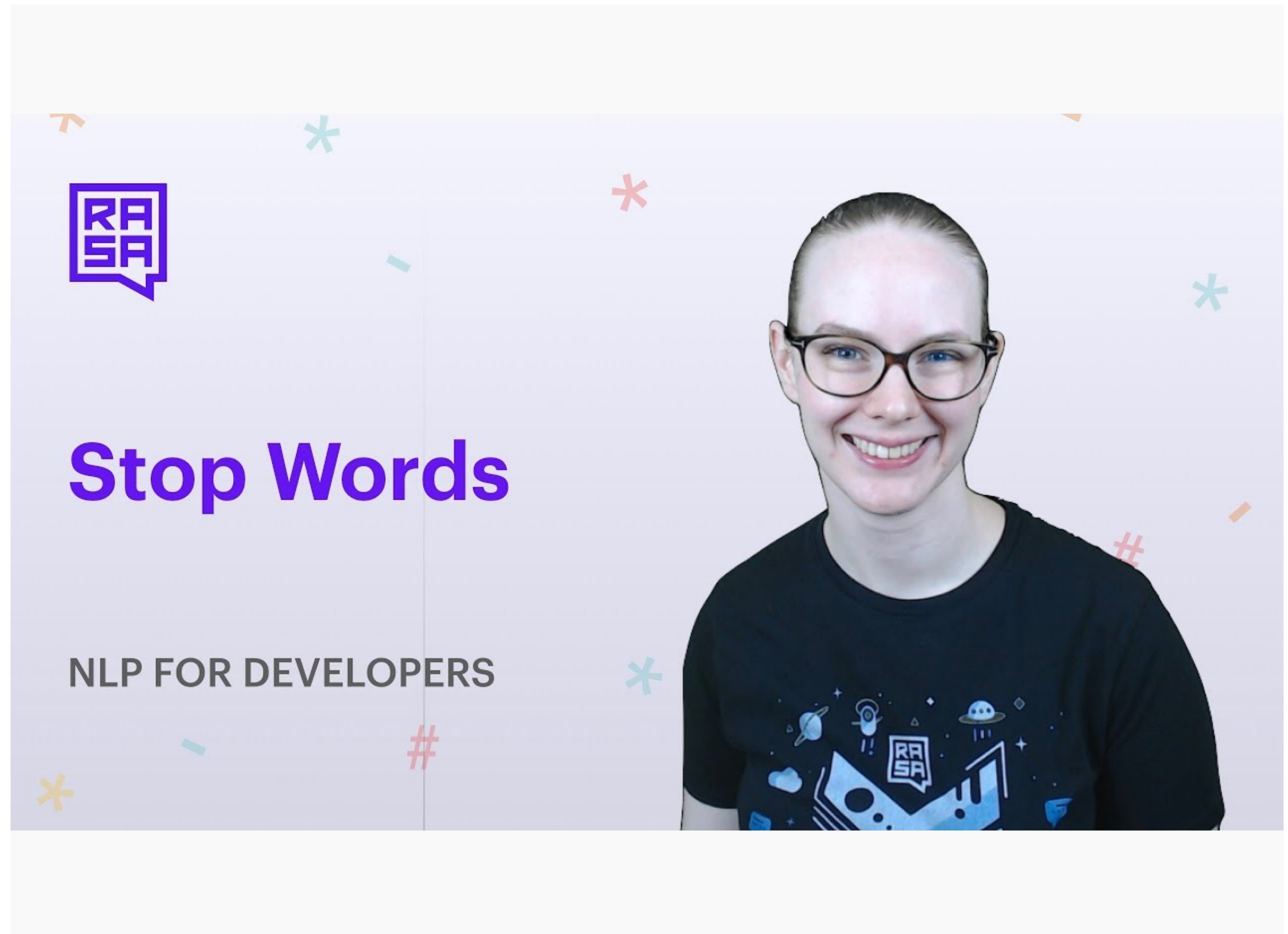
It consists only of splitting a sentence by the whitespace and punctuation marks.

- Raw text: I ate a burger, and it was good.
- Tokenized text: ['I', 'ate', 'a', 'burger', ',', 'and', 'it', 'was', 'good', '.']

The difficulty in tokenization lies on how to **get the ideal split** so that all the tokens in the text have the **correct meaning**, and there are no left out tokens.

Stop Words

A list of very common but uninformative words that you want to ignore



Stop Words

Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information

 **Do we always remove stop words? Are they always useless for us?**

- Movie review: “The movie was not good at all.”
- Text after removal of stop words: “movie good”


We **do not always remove the stop words**. The removal of stop words is highly dependent on the task we are performing and the goal we want to achieve.

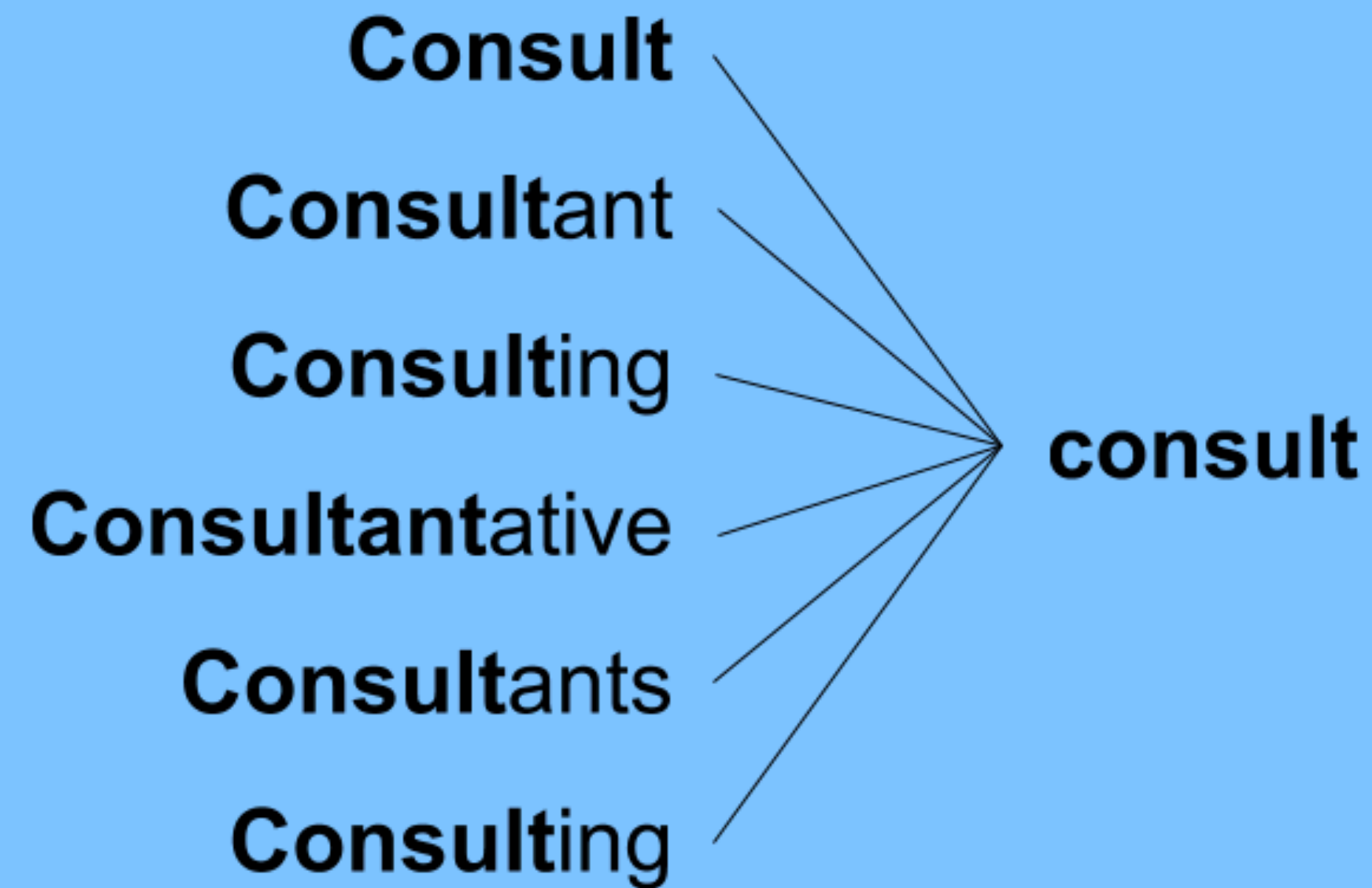
Stemming

Process of **reducing inflected (or derived) words to their root word** or word stem.

Applications of stemming:

- Stemming is used in information retrieval systems like search engines.
- It is used to determine domain vocabularies in domain analysis.

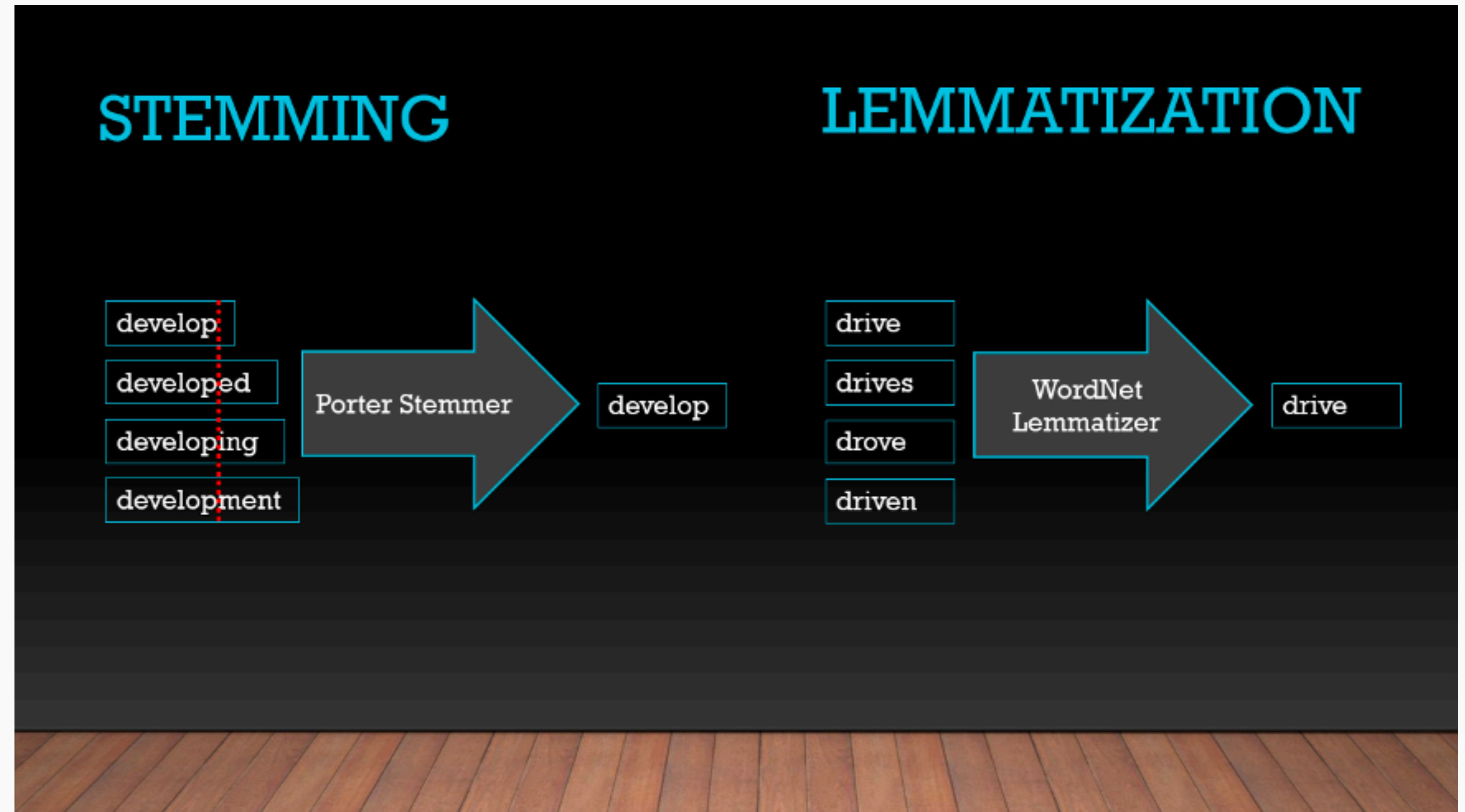
 **Fun Fact:** Google search adopted a word stemming in 2003. Previously a search for “fish” would not have returned “fishing” or “fishes”.



Lemmatization

Lemmatization is a method responsible for **grouping different inflected forms of words into the root form**, having the same meaning.

It is similar to stemming, in turn, it gives the stripped word that has some dictionary meaning. The Morphological analysis would require the extraction of the **correct lemma of each word**.



Activity 1

Activity 1 - Libraries import

Import tensorflow libraries on Google Colab environment:

W_001	Installing libraries	<pre>import tensorflow as tf from tensorflow import keras from tensorflow.keras.preprocessing.text import Tokenizer</pre>	2 min.
-------	----------------------	---	--------

Activity 1 - Tokenization basics

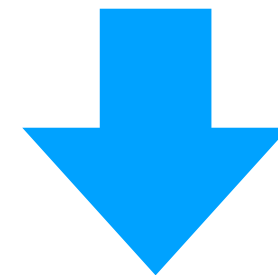
Represent these sentences as python array of strings:

W_002

Installing libraries

```
sentences = [  
    'i love my dog',  
    'I, love my cat'  
]
```

5 min.



```
{ 'i': 1, 'love': 2, 'my': 3, 'dog': 4, 'cat': 5 }
```

Activity 1 - Turning sentences into data

Create a sequences of tokens for these sentences:

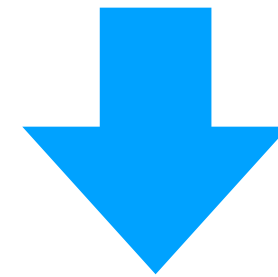
W_003

Tokenization

```
sentences = [  
    'I love my dog',  
    'I love my cat',  
    'You love my dog!',  
    'Do you think my dog is amazing?'  
]
```

5 min.

Print the word index and the sequence



```
Word Index = {'my': 1, 'love': 2, 'dog': 3, 'i': 4, 'you': 5, 'cat': 6, 'do': 7, 'think': 8, 'is': 9, 'amazing': 10}
```

```
Sequences = [[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5, 8, 1, 3, 9, 10]]
```

What happens if we need to train a Neural Network but the NN was not trained with those words?

Activity 1 - Challenge when tokenising

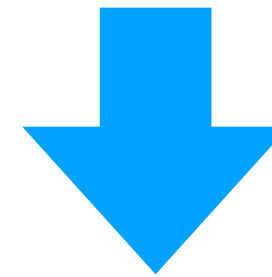
Create a sequences of tokens for these sentences that the tokenizer wasn't fit to

W_004

Tokenization
challenge

```
test_data = [  
    'i really love my dog',  
    'my dog loves my manatee'  
]
```

5 min.



```
Test Sequence = [[4, 2, 1, 3], [1, 3, 1]]
```

The word really does not appear in the first sequence: [4, 2, 1, 3] = I love my dog

On the 2nd sentence we do not have loves and manatee: [1, 3, 1] = My dog my

How do you think we can solve this?

Activity 1 - Out of vocabulary

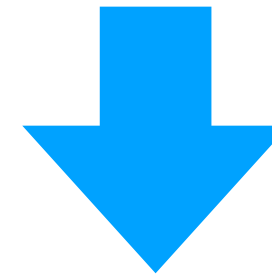
W_005

Out of vocabulary
management

By using a oov_token and setting is as something we wouldn't expect:

```
sentences = [  
    'I love my dog',  
    'I love my cat',  
    'You love my dog!',  
    'Do you think my dog is amazing?'  
]  
  
tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
```

5 min.



```
Word Index = {'<OOV>': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}
```

```
Sequences = [[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
```


Sentences have different lengths, so sequences will also have that problem. How do we manage that?

Activity 1 - Length

W_006

Import
pad_sequences

```
from tensorflow.keras.preprocessing.sequence import  
pad_sequences #import pad_sequences
```

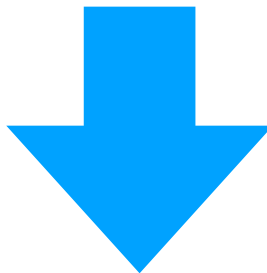
2 min.

W_007

Padded sequence

Generate a padded sequence with zeros at the beginning.

NOTE: #pre: zeros at the beggining \ post: zeros at the end



```
Word Index = {'<OOV>': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}  
  
Sequences = [[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]  
  
Padded Sequences:  
[[ 0  0  0  5  3  2  4]  
 [ 0  0  0  5  3  2  7]  
 [ 0  0  0  6  3  2  4]  
 [ 8  6  9  2  4 10 11]]
```

Activity 1 - Model generation (1/2)

W_008	Import json	Import json library get json file	2 min.
W_009	Get data	!wget --no-check-certificate \ https://storage.googleapis.com/laurencemoroney-blog.appspot.com/ sarcasm.json \ -O /tmp/sarcasm.json	2 min.
W_010	Open data	we open the data and iterate to fill in 3 created variables (sentences = [] labels = [], urls = [])	5 min.
W_011	Tokenization of titles	Lets tokenize the sentences (titles)	2 min.
W_012	Sequences validation	Lets check the sequences	2 min.
W_013	Paremeters	<pre>#parameters that we will use during our model training vocab_size = 10000 embedding_dim = 16 max_length = 100 trunc_type='post' padding_type='post' oov_tok = "<OOV>" training_size = 20000</pre>	2 min.

Activity 1 - Model generation (2/2)

W_014	Partition	What is the problem here? #If we would like to simply partition the dataset, there is a bit of a problem because we generated a tokenizer with the entire dataset	2 min.
W_015	Generate model	<pre>model = tf.keras.Sequential([tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length), tf.keras.layers.GlobalAveragePooling1D(), tf.keras.layers.Dense(24, activation='relu'), tf.keras.layers.Dense(1, activation='sigmoid')]) model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])</pre>	10 min.
W_016	Tensorflow 2.x	<pre># Need this block to get it to work with TensorFlow 2.x import numpy as np training_padded = np.array(training_padded) training_labels = np.array(training_labels) testing_padded = np.array(testing_padded) testing_labels = np.array(testing_labels)</pre>	5 min.
W_017	Train model	<pre>num_epochs = 50 history = model.fit(training_padded, training_labels, epochs=num_epochs, validation_data=(testing_padded, testing_labels), verbose=2)</pre>	5 min.
W_018	Use model	<pre>sentence = ["granny starting to fear spiders in the garden might be real", "game of thrones season finale showing this sunday night"] sequences = tokenizer.texts_to_sequences(sentence) padded = pad_sequences(sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type) print(model.predict(padded))</pre>	

Activity 2

Activity 2 - Libraries installation and import

W_001	Installing libraries	<p>Install using pip tensorflow libraries on Google Colab environment:</p> <ul style="list-style-type: none">- tensorflow- tensorflow-hub- sentencepiece	5 min.
W_002	Libraries import	<p>Import the following libraries:</p> <ul style="list-style-type: none">- tensorflow- tensorflow_hub- pandas- numpy- spacy- nltk- matplotlib.pyplot- seaborn- tensorflow.keras -> layers- tensorflow.keras -> Model- tensorflow.keras.optimizers -> SGD, Adam- tensorflow.keras.callbacks -> ModelCheckpoint- tensorflow.keras.layers -> Dense, Input, Dropout, GlobalAveragePooling1D	5 min.

Activity 2 - Data import and understanding

W_003	Data import	Clone data from my Github: https://github.com/maglionejm/kaggle_nlp	5 min.
W_004	Data understanding	Look at the different datasets. What is the data about? What is the composition of the different datasets?	10 min.
W_005	Data concatenation	Concatenate train and test datasets	5 min.
W_006	Cleaning nulls (?)	Detect and clean all the null values. NOTE: Be careful with the 30% of data with NULL values. Should you delete it?	5 min.

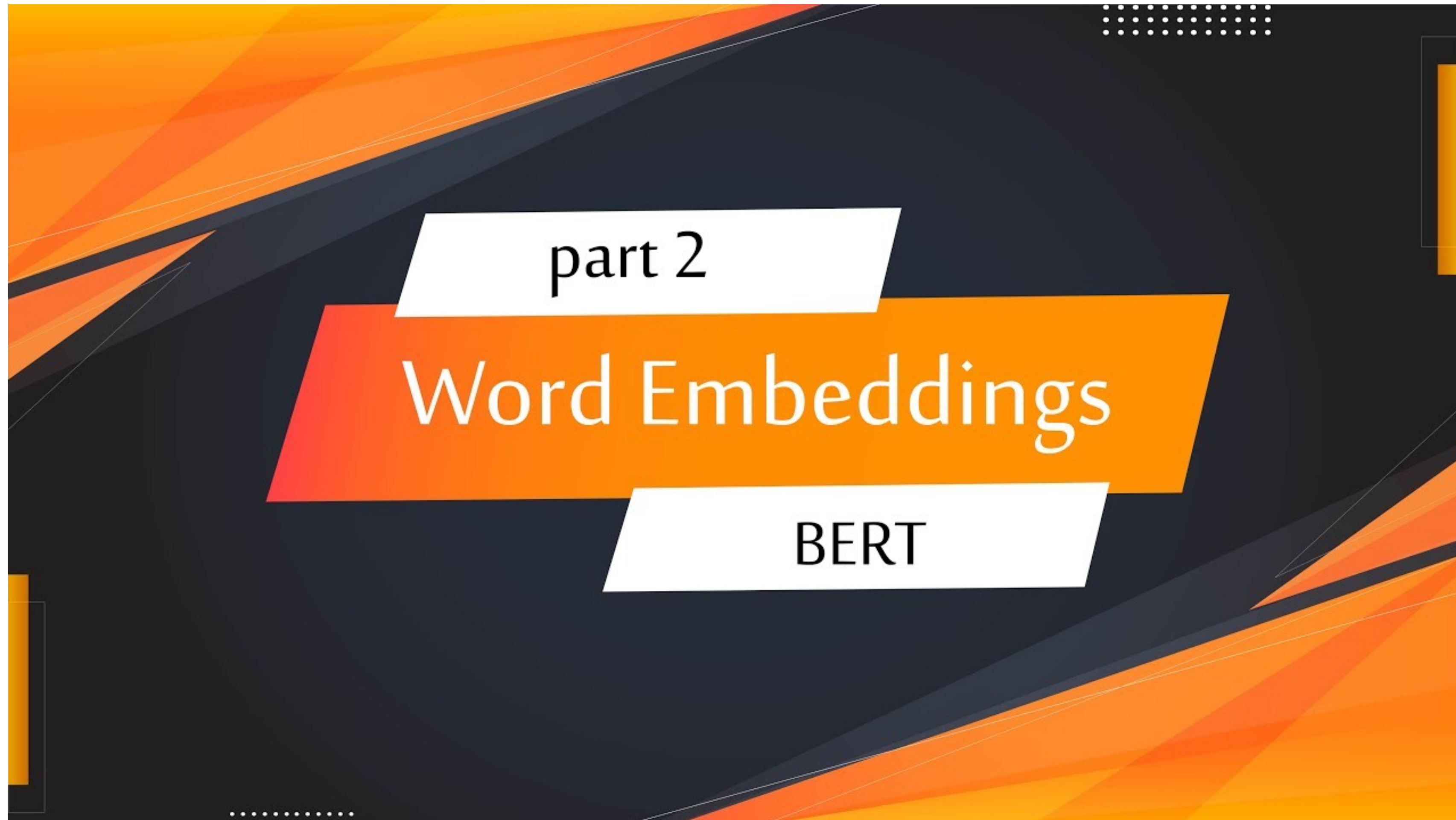
Activity 2 - Quick descriptive analysis

W_007	Descriptive analysis	<div>1. What are the top 30 locations in the dataset? Please plot them.</div> <div>2. What are the top 20 keywords in disastrous and non_disastrous tweets? Plot 2 horizontal bar charts with seaborn sorted by largest to lowest volume.</div>	10 min.
		<div>Optional:</div> <div>3. What are the distributions for: 'word_count', 'stop_words', 'mean_word_length', 'punc_count','hashtag_count', 'mention_count', 'url_count'?</div>	

Activity 2 - Data cleaning

W_008	Delete columns	Delete the following columns: 'location', 'keyword' and 'id'	2 min.
<hr/>			
W_009	Cleaning data	<div>Create a function that does the following:</div> <div><div>1. Add nltk and spacy to clean stopwords. Experimentate with both to understand which one works better.</div><div>2. Replace tweets that contain urls with a blank space</div><div>3. Remove punctuation</div><div>4. Remove extra spaces between words</div><div>5. Lemmatize words to simplify the tweets</div></div>	15 min.
<hr/>			
W_010	Cleaned data	Add cleaned data as another column to the dataframe calling the generated function.	2 min.

Word embeddings theory



Activity 2 - Model generation

BERT_001	Tokenizer	1. Import BERT tokenizer	5 min.
BERT_002	Model	<div>1. Import BERT Model (Depending on time you have to train, select a small model).</div> <div>2. Define the BERT Keras Layer to trainable=True to allow the model to be trained with our own data.</div> <div>3. Define a vocabulary_file and make the tokenizer lower case all the words.</div> <div>4. Define a class TweetClassifier with the following functionality:</div> <div>- Define the parameters that you'll need for the model generation (for example epochs, batch_size, activation, optimizer, metrics, loss, etc.)</div> <div>- Define a function encode which handles: i.the tokenization of the words, ii. Add 2 tokens at the end and the beggining (CLS and SEP), iii. Define the length of the padding based on the max length of the model, iv. Define mask (shows which part of the words are words), v.Vectorization of the text, vi. Define segment (Only zeros with the max length of the words).</div> <div>- Define a function make_model which handles: i. defines the inputs of the model, ii. give all inputs as a list to the bert layer, iii. Take out the first token of each sentence, iv. use 1 dense layer with a sigmoid activation, v. Define keras model that takes all the input, vi. Define the optimizer (depending on which one generate an if).</div> <div>- Define a train function which handles: i. fits the model, ii. Optional: Generate a checkpoint to minimize loss to save the model.</div> <div>- Define a function predict which handles: i. Load the trained model, ii. Use it to predict the x test</div>	20 min.