



# IIC 3800 Tópicos en CC NLP

<https://github.com/marcelomendoza/IIC3800>

## Clasificación con word embeddings

```
inputs = Input(shape=(max_tokens, ))
```

```
embeddings_layer = Embedding(input_dim=len(tokenizer.index_word)+1,  
output_dim=embed_len, input_length=max_tokens, trainable=False,  
weights=[glove_50_embeddings])
```

```
dense1 = Dense(128, activation="relu")
```

```
dense2 = Dense(64, activation="relu")
```

```
dense3 = Dense(len(classes), activation="softmax")
```

```
x = embeddings_layer(inputs)
```

```
x = tensorflow.reduce_sum(x, axis=1)
```

```
x = dense1(x)
```

```
x = dense2(x)
```

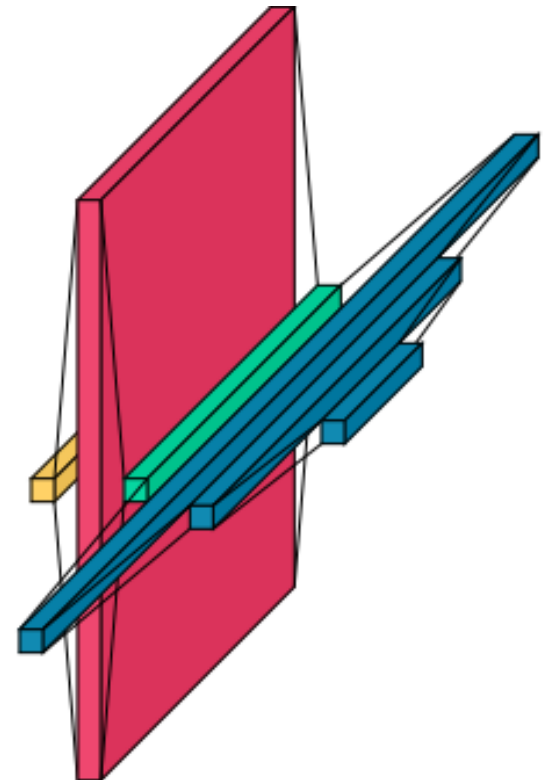
```
outputs = dense3(x)
```

```
model = Model(inputs=inputs, outputs=outputs)
```

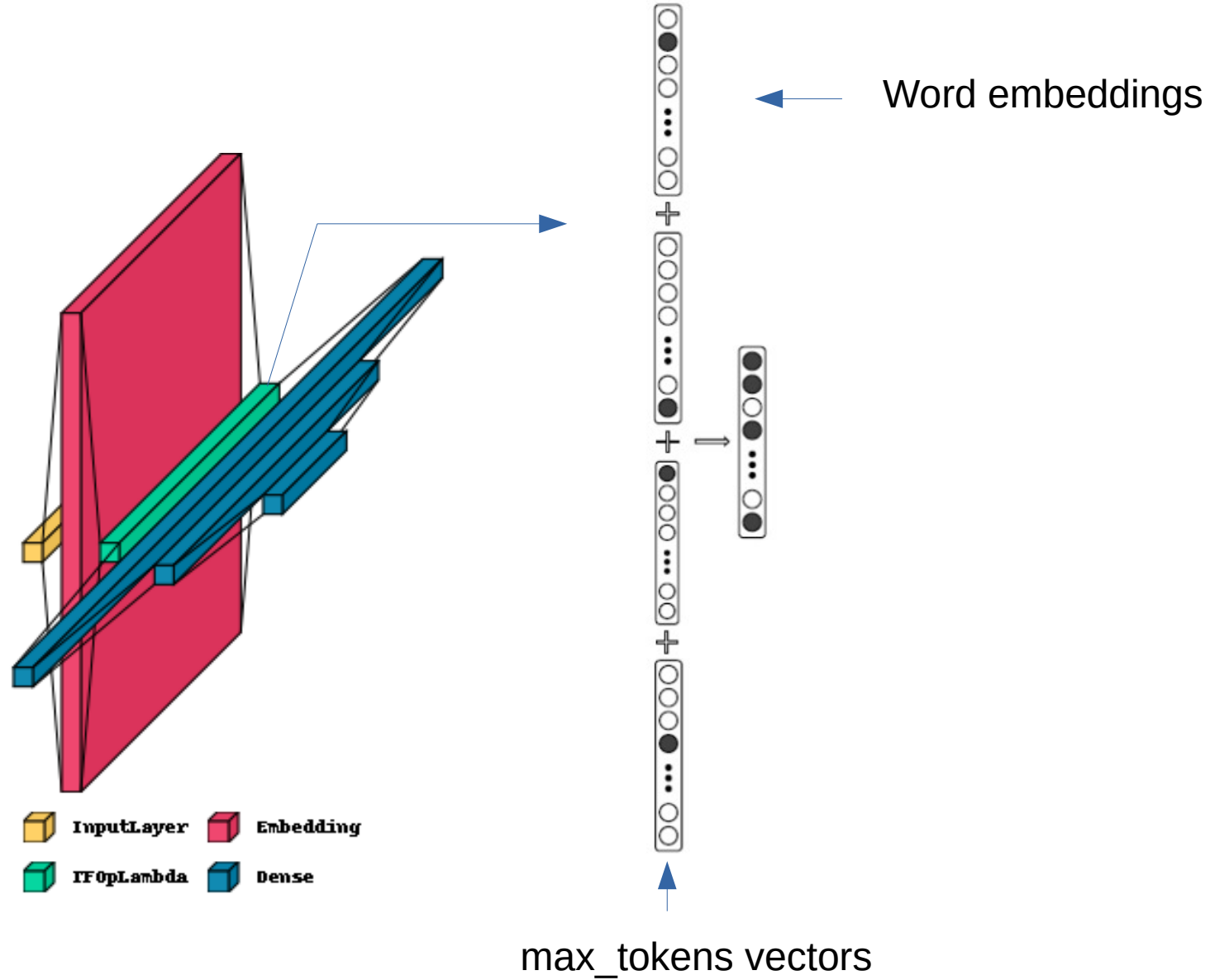
| V |



forward



## Clasificación con word embeddings



- SUBWORDS, FASTTEXT -

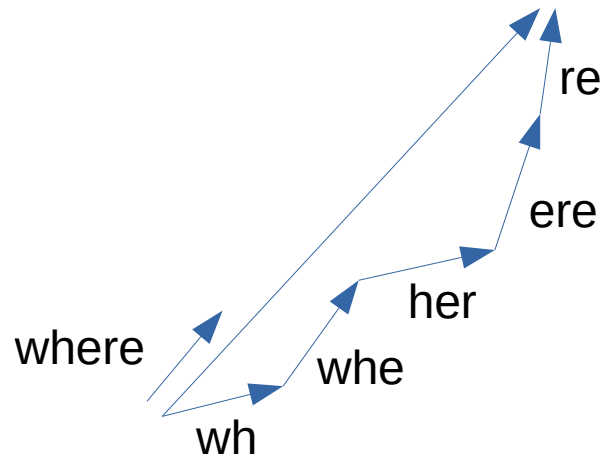
## Subwords

Las palabras se representan por un bag-of char  $n$ -grams:

Ej.:  $n = 3$ ,      where  $\rightarrow$  <wh, whe, her, ere, re>, <where>

Sea  $\mathcal{G}_w \subset \{1, \dots, G\}$  el conjunto de char  $n$ -grams de la palabra  $w$ . En FastText, cada  $n$ -gram tiene un vector  $\mathbf{z}_g$  que lo representa.

Una palabra se representa como la **suma** de los vectores  $\mathbf{z}_g$ , en  $\mathcal{G}_w \subset \{1, \dots, G\}$



## Subwords

FastText es una extensión de skip-grams basada en sub-words.

La función objetivo de skipgrams corresponde a la log verosimilitud:

$$\sum_{t=1}^T \sum_{c \in \mathcal{C}_t} \log p(w_c | w_t),$$


Podemos usar una softmax para definir la probabilidad de una palabra de contexto:

$$p(w_c | w_t) = \frac{e^{s(w_t, w_c)}}{\sum_{j=1}^W e^{s(w_t, j)}}.$$

Las palabras de contexto son ejemplos positivos. Usamos negative sampling para que el problema sea de clasificación binaria.

## Subwords

Cuando consideramos *negative sampling*, la función objetivo de clasificación binaria corresponde a una *log loss*:

$$\log \left( 1 + e^{-s(w_t, w_c)} \right) + \sum_{n \in \mathcal{N}_{t,c}} \log \left( 1 + e^{s(w_t, n)} \right)$$


## Subwords

Cuando consideramos *negative sampling*, la función objetivo de clasificación binaria corresponde a una *log loss*:

$$\log \left( 1 + e^{-s(w_t, w_c)} \right) + \sum_{n \in \mathcal{N}_{t,c}} \log \left( 1 + e^{s(w_t, n)} \right)$$

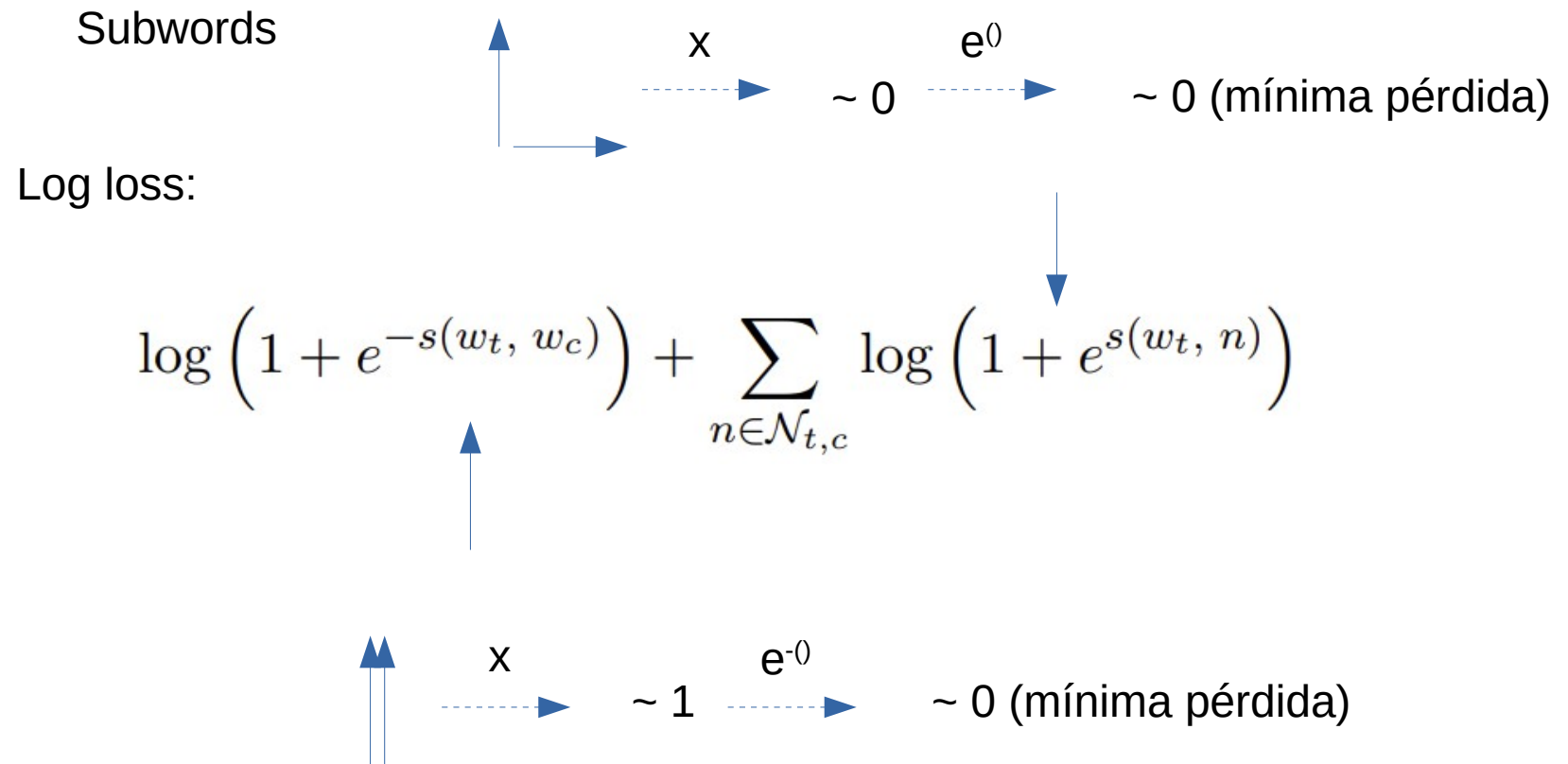
└ Negative samples (fuera del contexto de  $w_t$ )

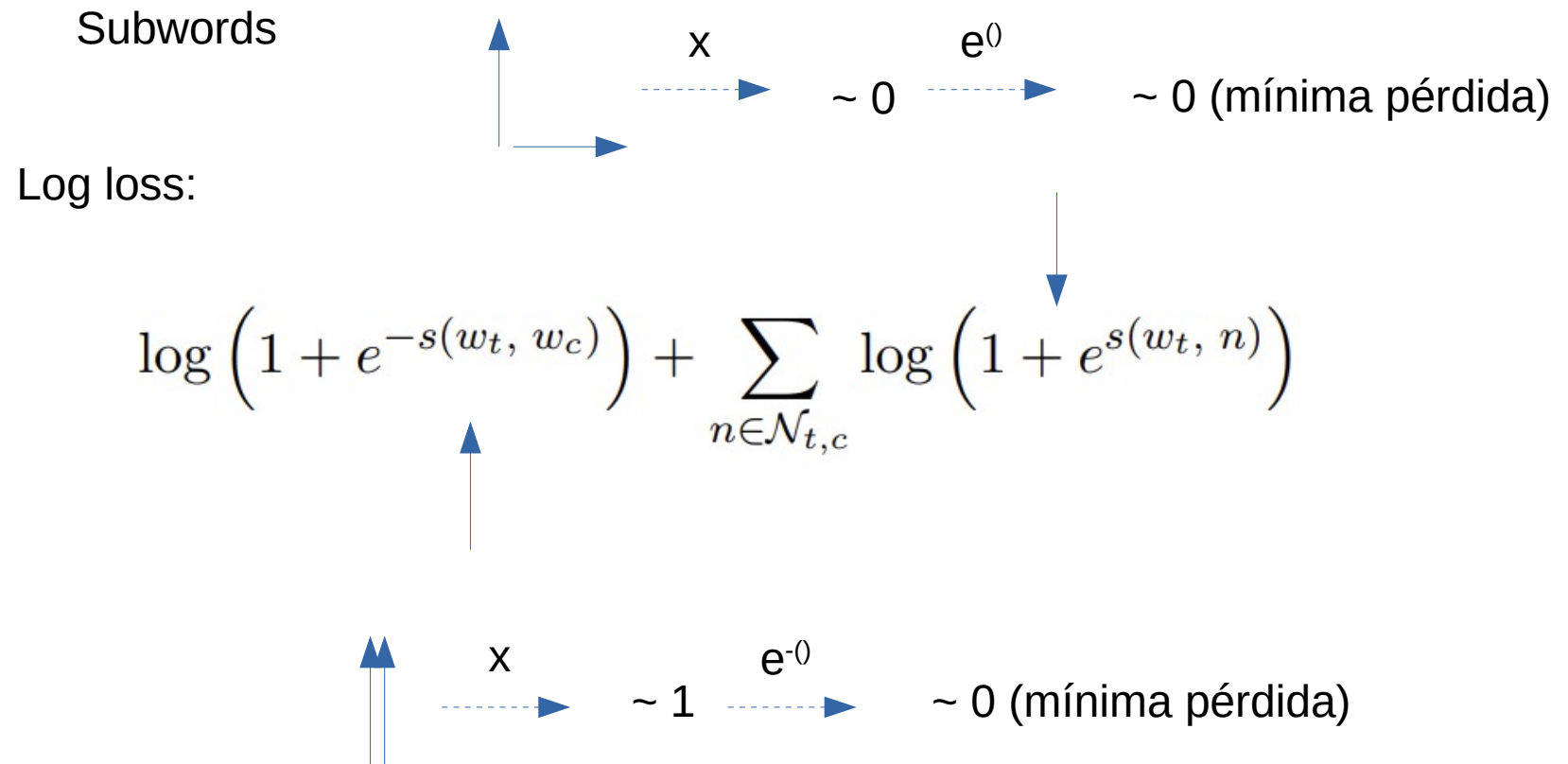
Luego, tenemos varios problemas de clasificación binaria independientes que superponemos durante el entrenamiento:

$$\sum_{t=1}^T \left[ \sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

donde:  $\ell : x \mapsto \log(1 + e^{-x})$





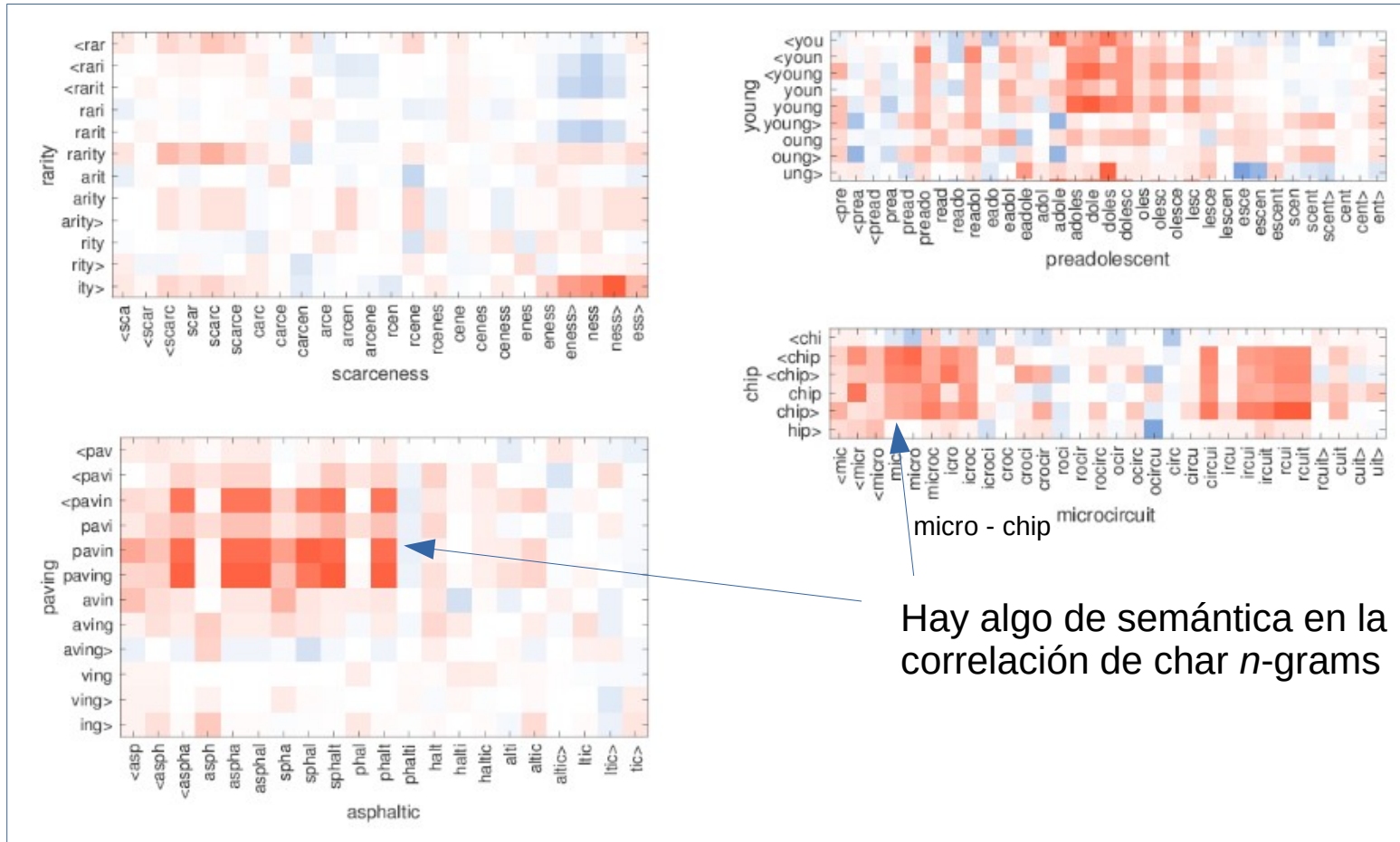


FastText se entrena usando como función de scoring:

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c.$$

## Subwords

- FastText captura palabras out of vocabulary y palabras poco frecuentes
- En algunos idiomas obtiene resultados mucho mejores que word2vec



### Aspectos prácticos:

- FastText mapea los  $n$ -grams a enteros usando hashing.
- App. 2 millones de char  $n$ -grams se usan en FastText.