Curso de

Algoritmos de Clasificación de Texto

Francisco Camacho
@el_pachocamacho

[C1] Introducción a la desambiguación

Desambiguación y etiquetado de palabras

¿Por qué es tan difícil?

El lenguaje humano es **difuso**, **ambiguo** y requiere **mucho contexto**

Debo ir al **banco** para retirar dinero

Te puedes sentar en ese **banco** para descansar

Mi hermano es una persona muy noble

El noble del castillo no quiere ayudar a su pueblo

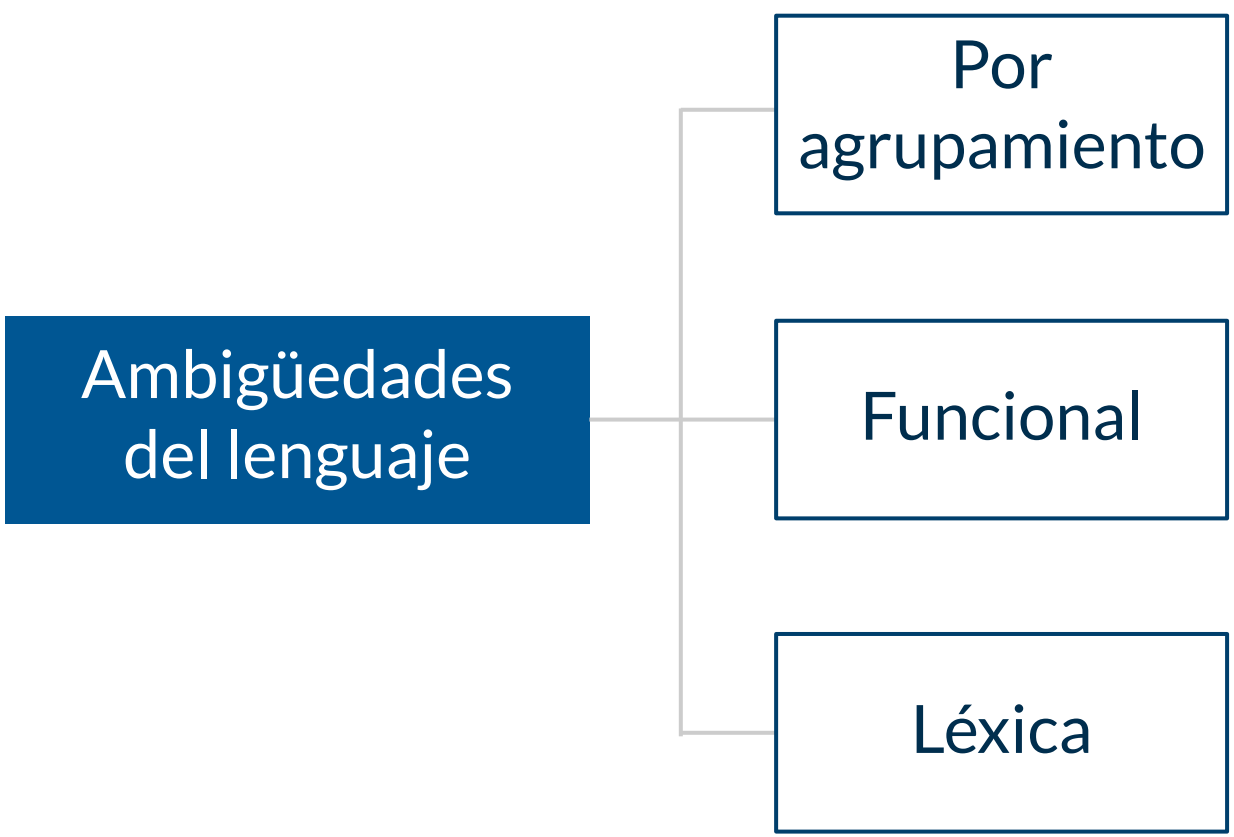
adjetivo

Mi hermano es una persona muy noble

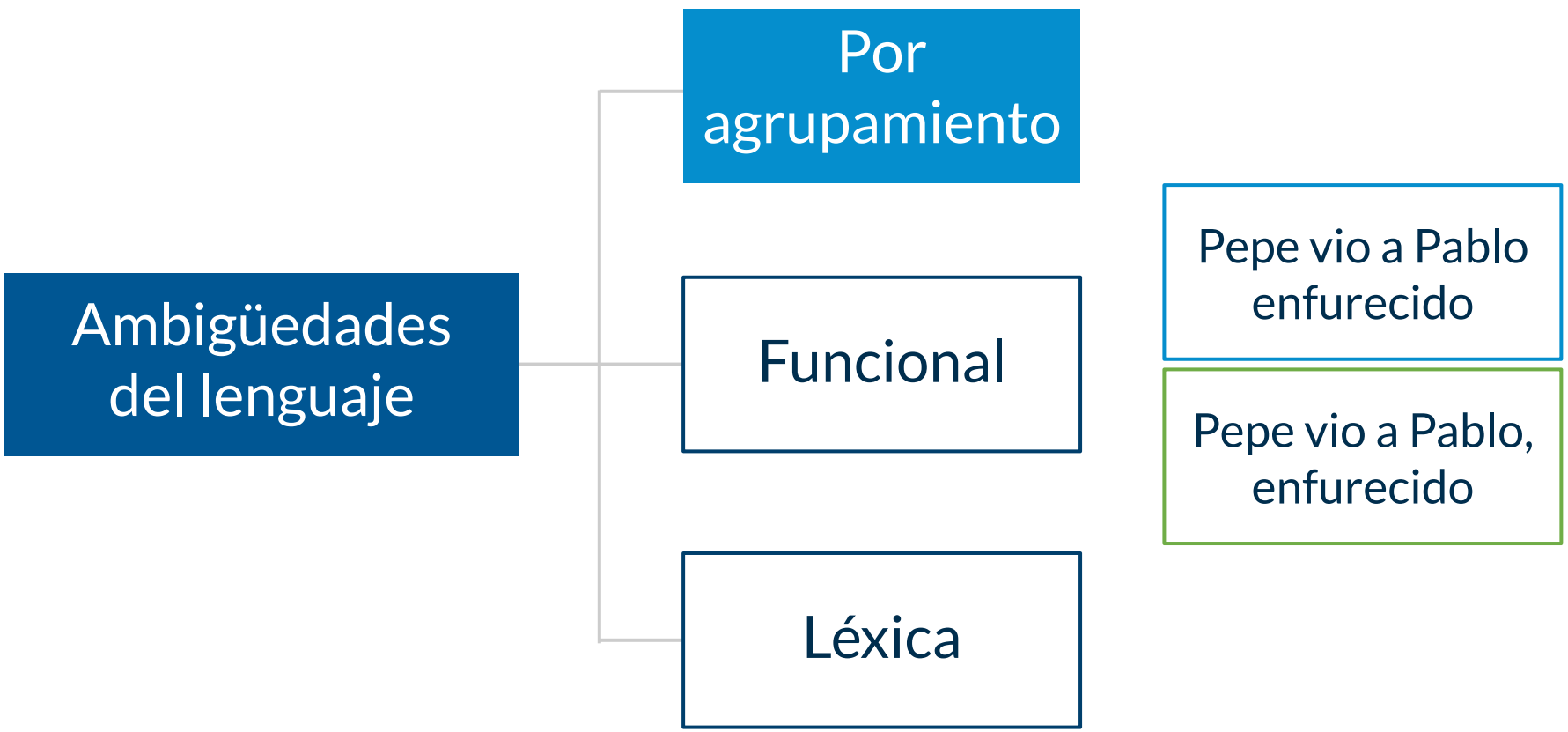
sustantivo

El noble del castillo no quiere ayudar a su pueblo

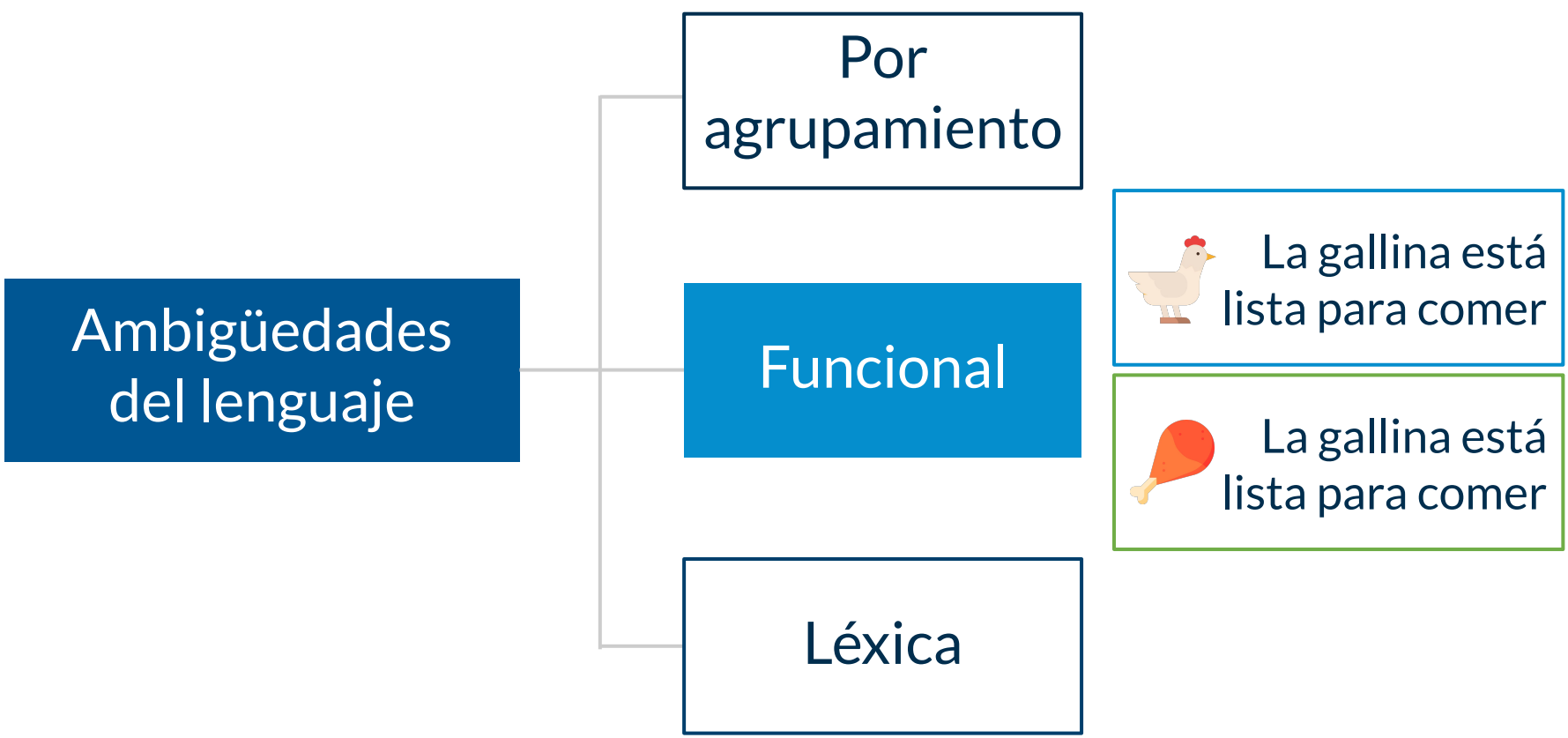
Ambigüedades del lenguaje



Ambigüedades del lenguaje



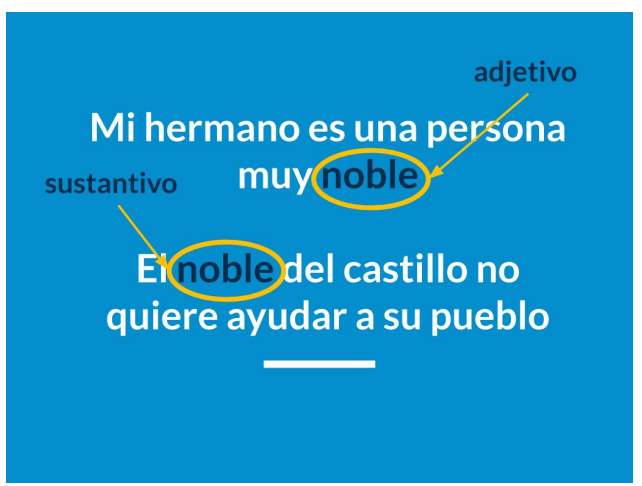
Ambigüedades del lenguaje



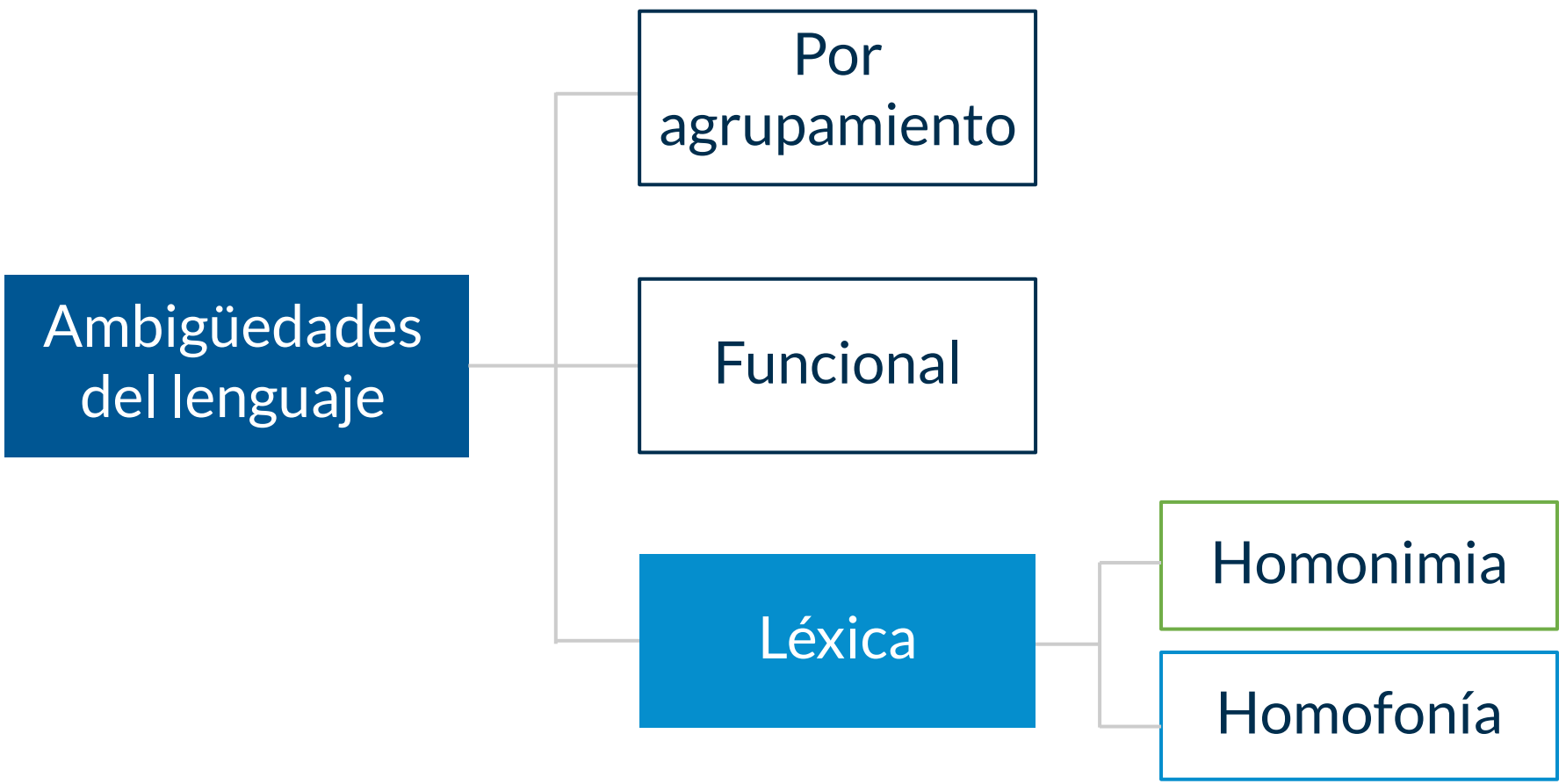
Ambigüedades del lenguaje

Ambigüedades del lenguaje

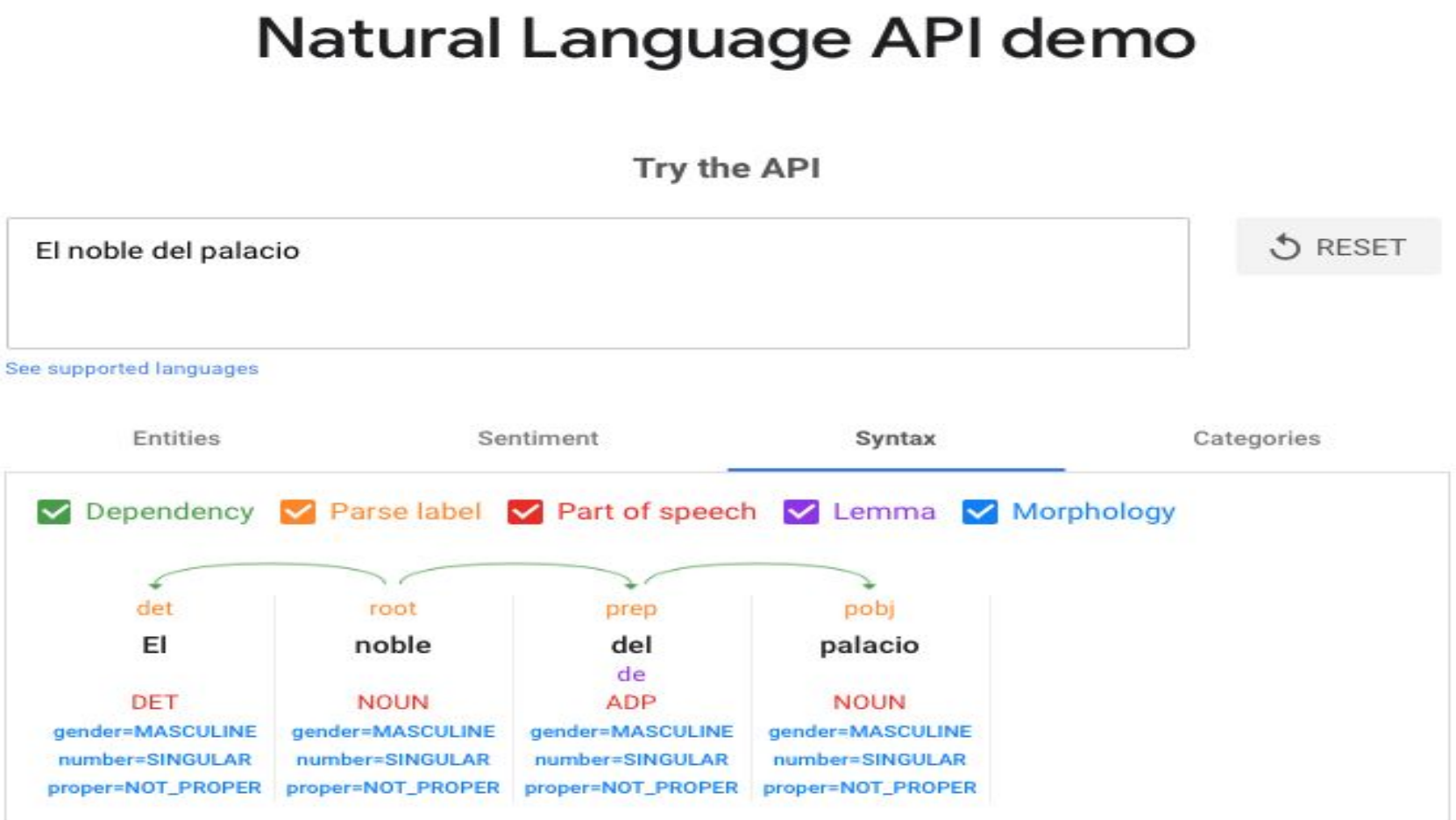
- Por agrupamiento
- Funcional
- Léxica



Ambigüedades del lenguaje



Etiquetado de palabras



https://cloud.google.com/natural-language

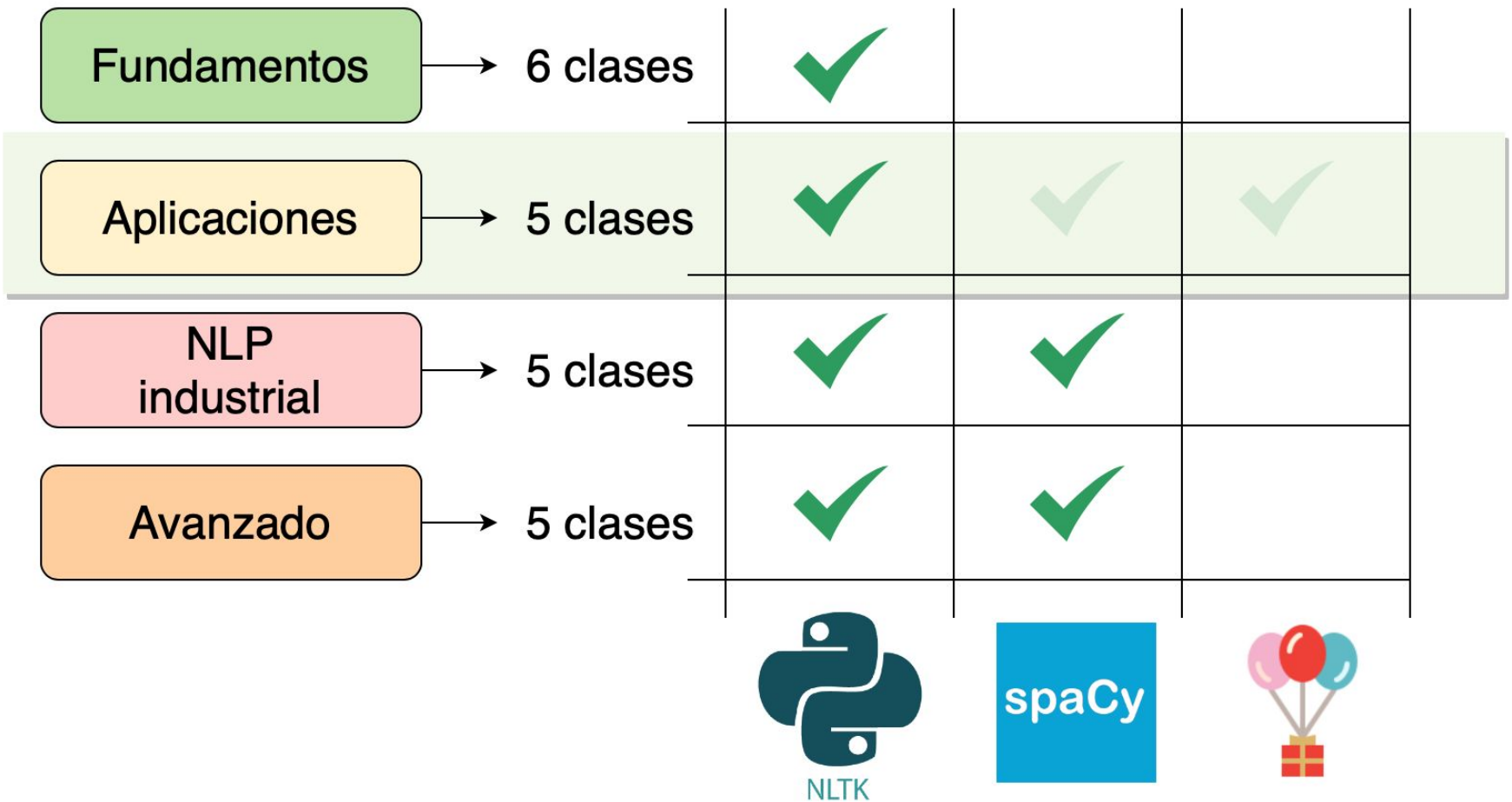
Aplicaciones

- Mejoras en motores de búsqueda, e-commerce y web.
- Automatización en manejo de CRMs.

Aplicaciones

- Censura en redes sociales.
- Orden de datos no-estructurados.

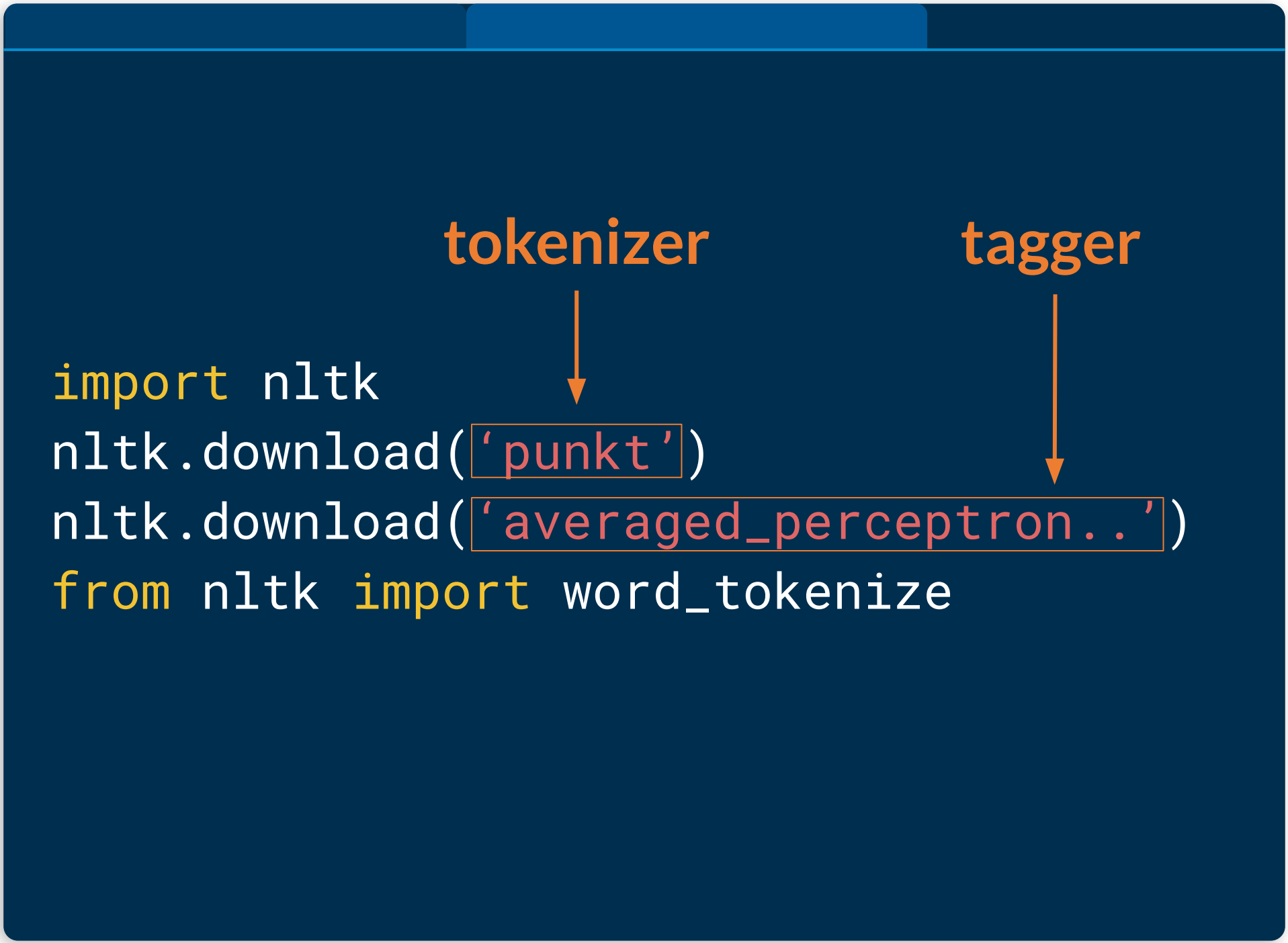
Roadmap del contenido



[C4] Cadenas de Markov

Desambiguación y etiquetado de palabras

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron..')
from nltk import word_tokenize
```


tokenizer
tagger
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron..')
from nltk import word_tokenize

Tokenizador: ¿Punkt?

Unsupervised Multilingual Sentence Boundary Detection

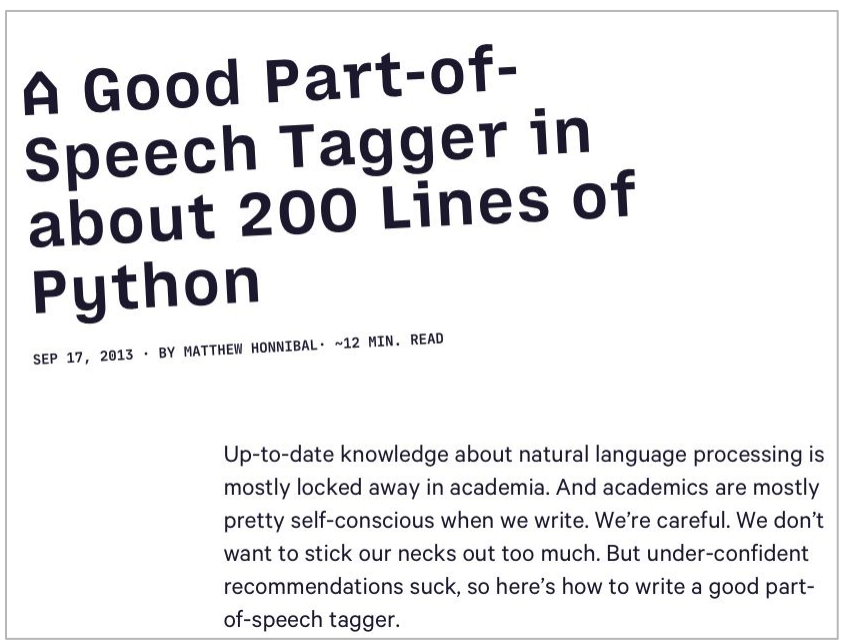
Tibor Kiss*
Ruhr-Universität Bochum

Jan Strunk**
Ruhr-Universität Bochum

In this article, we present a language-independent, unsupervised approach to sentence boundary detection. It is based on the assumption that a large number of ambiguities in the determination of sentence boundaries can be eliminated once abbreviations have been identified. Instead of relying on orthographic clues, the proposed system is able to detect abbreviations with high accuracy using three criteria that only require information about the candidate type itself and are independent of context: Abbreviations can be defined as a very tight collocation consisting of a truncated word and a final period, abbreviations are usually short, and abbreviations sometimes contain internal periods. We also show the potential of collocational evidence for two other important subtasks of sentence boundary disambiguation, namely, the detection of initials and ordinal numbers. The proposed system has been tested extensively on eleven different languages and on different text genres. It achieves good results without any further amendments or language-specific resources. We evaluate its performance against three different baselines and compare it to other systems for sentence boundary detection proposed in the literature.

https://www.aclweb.org/anthology/J06-4003.pdf

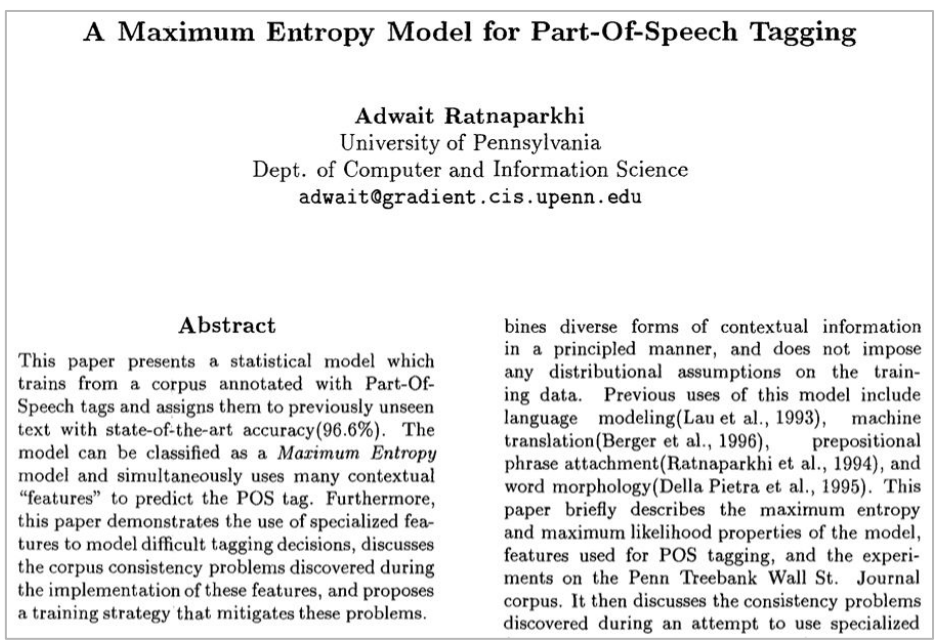
Etiquetador: ¿Averaged Perceptron Tagger?



A Good Part-of-Speech Tagger in about 200 Lines of Python

SEP 17, 2013 · BY MATTHEW HONNIBAL · ~12 MIN. READ

Up-to-date knowledge about natural language processing is mostly locked away in academia. And academics are mostly pretty self-conscious when we write. We're careful. We don't want to stick our necks out too much. But under-confident recommendations suck, so here's how to write a good part-of-speech tagger.



A Maximum Entropy Model for Part-Of-Speech Tagging

Adwait Ratnaparkhi
University of Pennsylvania
Dept. of Computer and Information Science
adwait@gradient.cis.upenn.edu

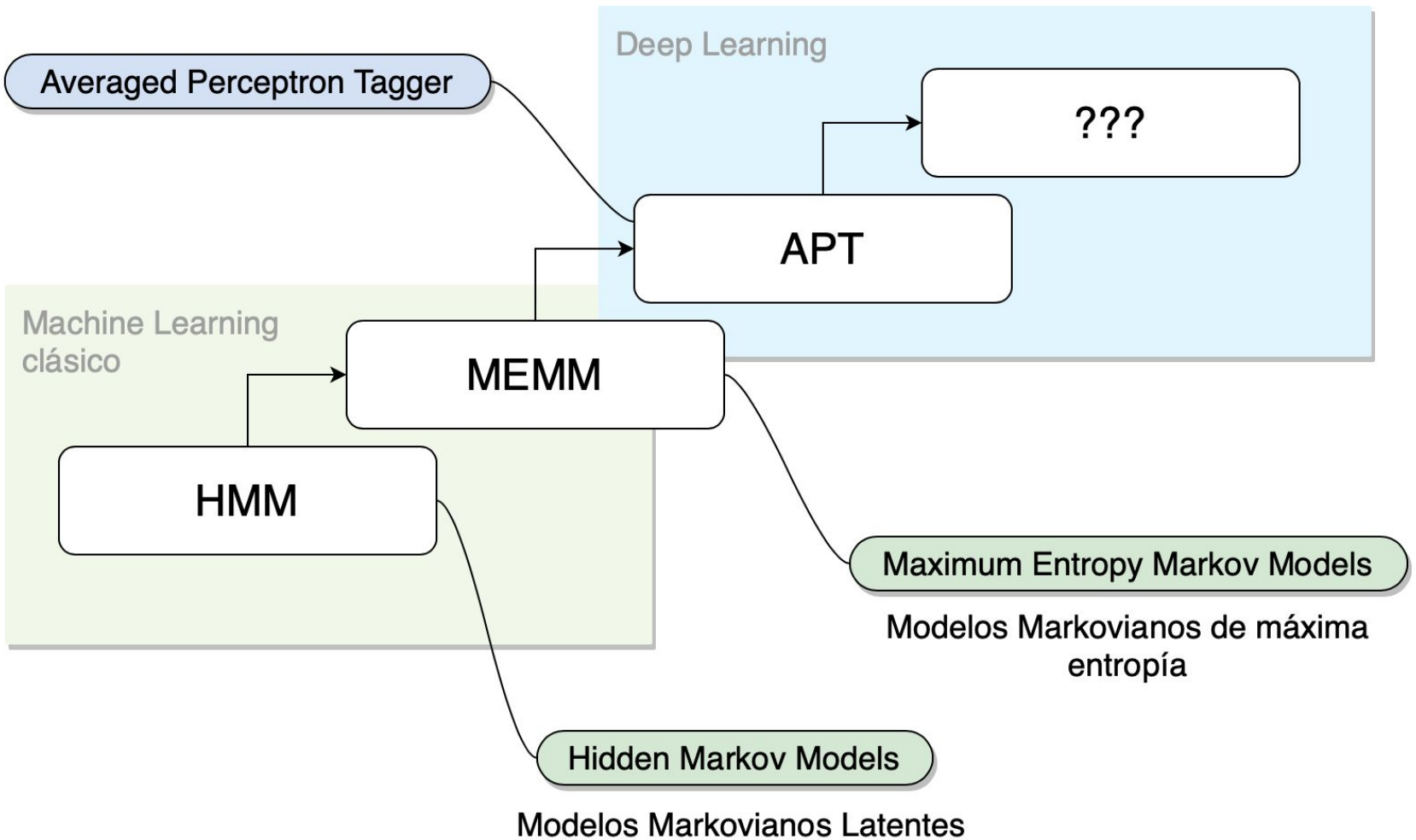
Abstract

This paper presents a statistical model which trains from a corpus annotated with Part-Of-Speech tags and assigns them to previously unseen text with state-of-the-art accuracy(96.6%). The model can be classified as a *Maximum Entropy* model and simultaneously uses many contextual "features" to predict the POS tag. Furthermore, this paper demonstrates the use of specialized features to model difficult tagging decisions, discusses the corpus consistency problems discovered during the implementation of these features, and proposes a training strategy that mitigates these problems.

bines diverse forms of contextual information in a principled manner, and does not impose any distributional assumptions on the training data. Previous uses of this model include language modeling(Lau et al., 1993), machine translation(Berger et al., 1996), prepositional phrase attachment(Ratnaparkhi et al., 1994), and word morphology(Della Pietra et al., 1995). This paper briefly describes the maximum entropy and maximum likelihood properties of the model, features used for POS tagging, and the experiments on the Penn Treebank Wall St. Journal corpus. It then discusses the consistency problems discovered during an attempt to use specialized

https://www.aclweb.org/anthology/W96-0213.pdf

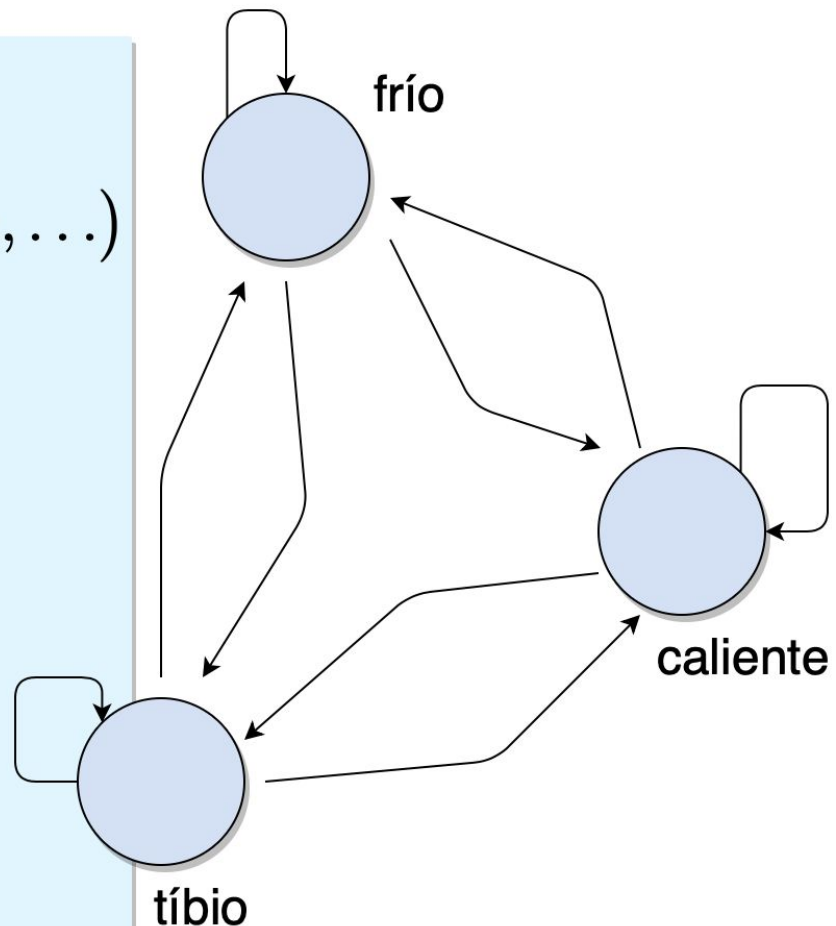
Escalera de modelos



Modelos Markovianos Latentes

MC: Markov Chain

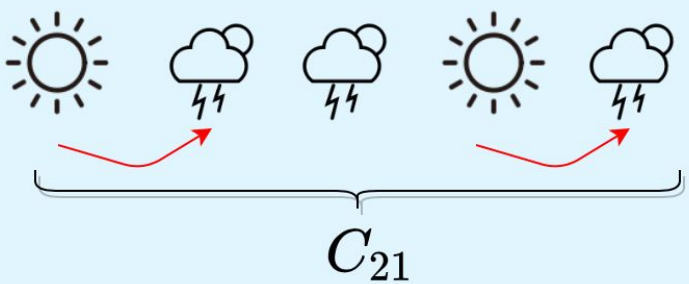
$$(\text{dia}_1, \text{dia}_2, \ldots) = (\text{frío}, \text{caliente}, \ldots)$$



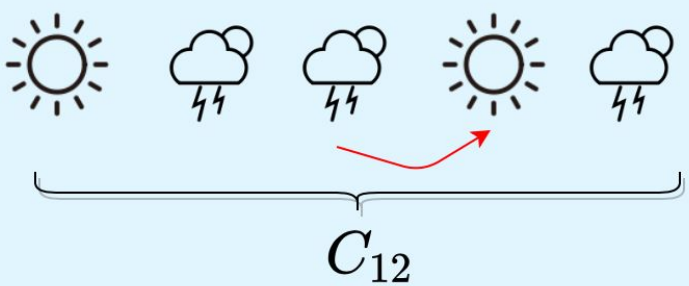
Modelos Markovianos Latentes

Probabilidades de transición

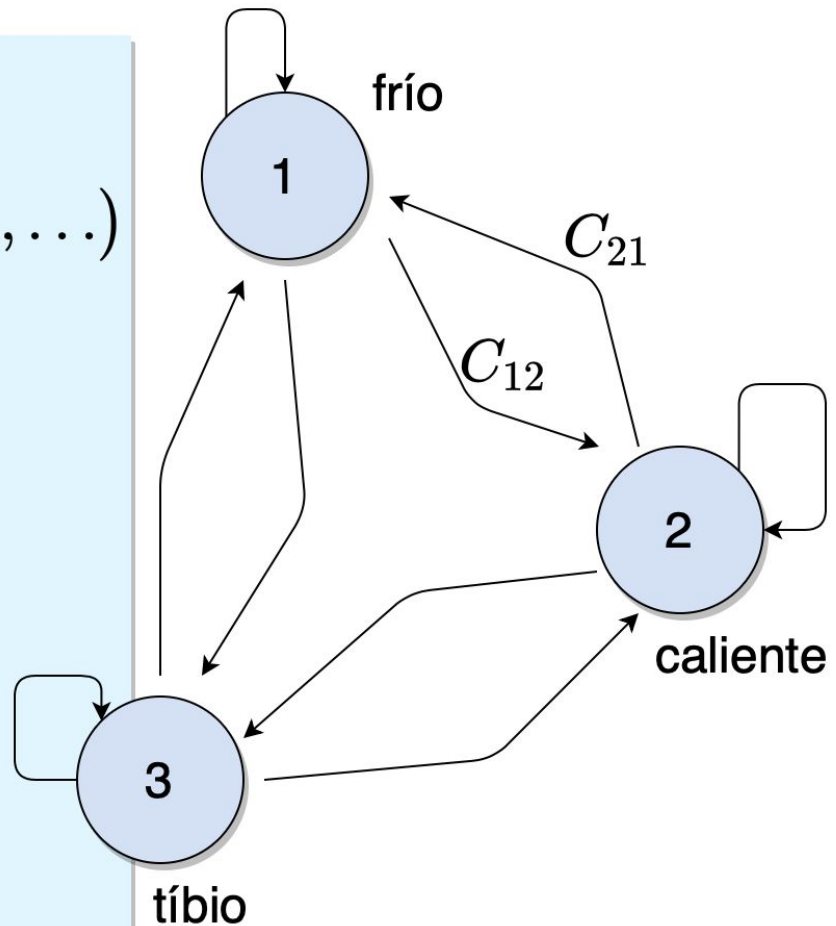
$$(\text{dia}_1, \text{dia}_2, \ldots) = (\text{frío}, \text{caliente}, \ldots)$$



C_{21}



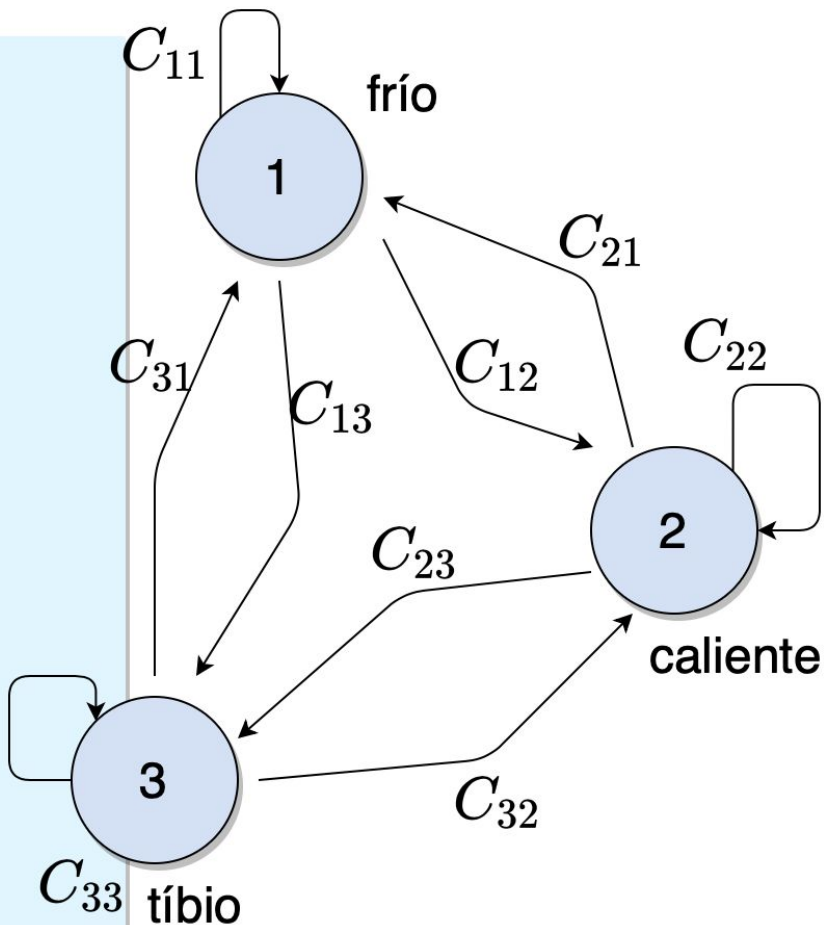
C_{12}



Modelos Markovianos Latentes

Matriz de transición

C_{11}	C_{12}	C_{13}
C_{21}	C_{22}	C_{23}
C_{31}	C_{32}	C_{33}



Modelos Markovianos Latentes

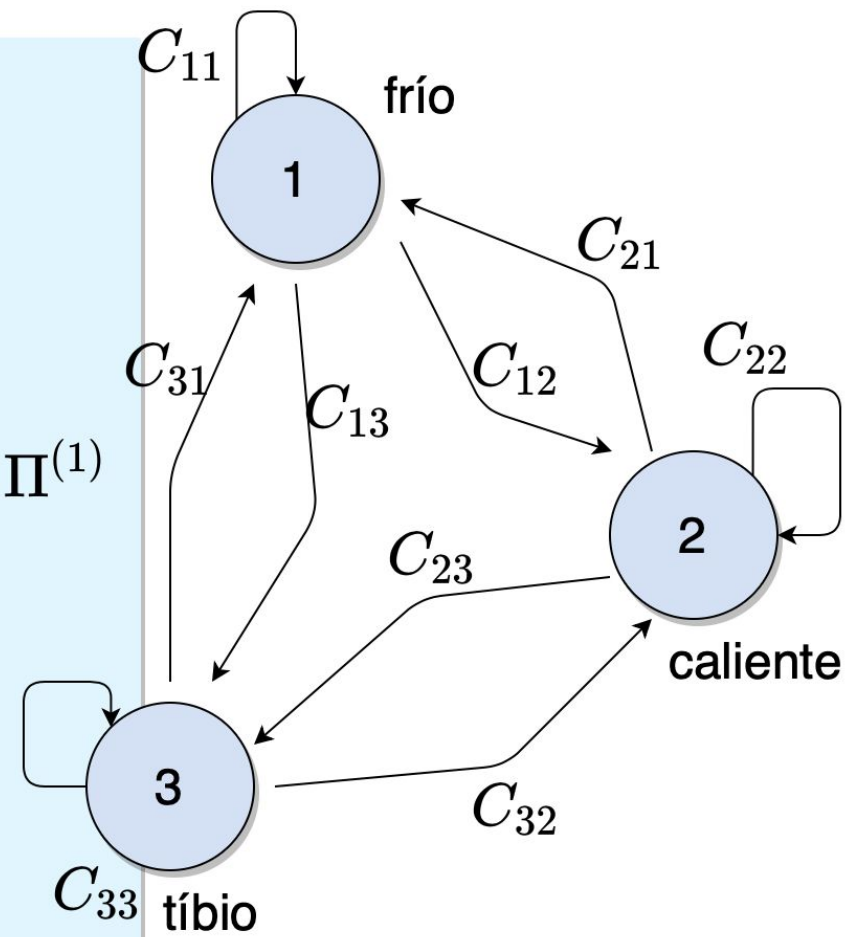
Distribución inicial de estados

$$A = \begin{array}{|c|c|c|} \hline C_{11} & C_{12} & C_{13} \\ \hline C_{21} & C_{22} & C_{23} \\ \hline C_{31} & C_{32} & C_{33} \\ \hline \end{array}$$

$$\Pi^{(0)} A = \Pi^{(1)}$$

$$\Pi^{(0)} = \begin{array}{|c|c|c|} \hline \rho_1^{(0)} & \rho_2^{(0)} & \rho_3^{(0)} \\ \hline \end{array}$$

$$\Pi^{(1)} = \begin{array}{|c|c|c|} \hline \rho_1^{(1)} & \rho_2^{(1)} & \rho_3^{(0)} \\ \hline \end{array}$$



[C5] Modelos Markovianos Latentes

Desambiguación y etiquetado
de palabras

Modelos Markovianos Latentes

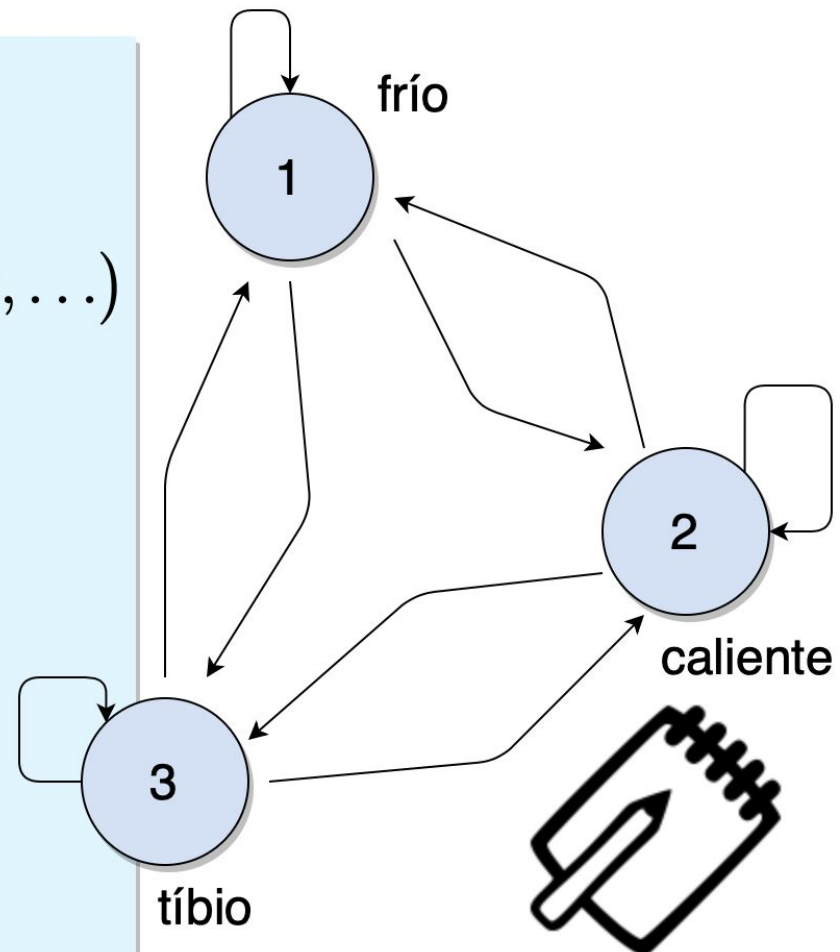
Ingredientes de MC

$$(\text{dia}_1, \text{dia}_2, \ldots) = (\text{frío}, \text{caliente}, \ldots)$$

A : matriz transición

Π : distribución de estados

$$\Pi^{(0)} A = \Pi^{(1)}$$



Modelos Markovianos Latentes

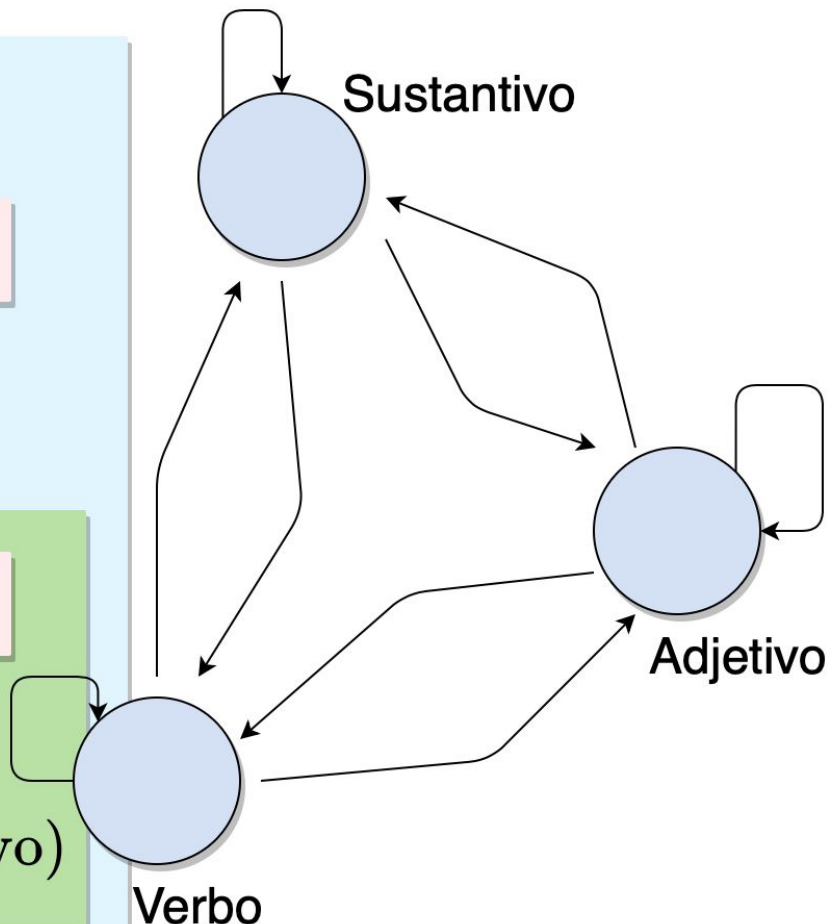
HMM: Hidden Markov Model

Sequencia de palabras

$$(w_1, w_2, \ldots) =$$
$$(\text{Pedro}, \text{es}, \text{ingeniero})$$

Sequencia de etiquetas

$$(t_1, t_2, \ldots) =$$
$$(\text{Sustantivo}, \text{Verbo}, \text{Sustantivo})$$



[C6] Entrenando un HMM

Desambiguación y etiquetado
de palabras

Modelos Markovianos Latentes

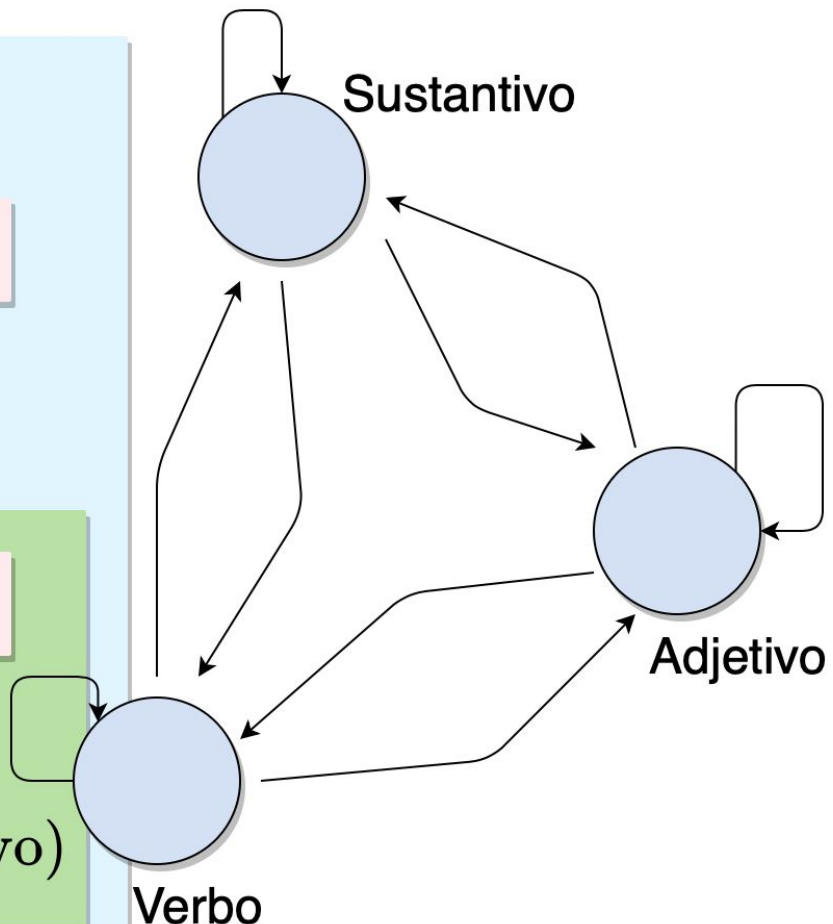
HMM: Hidden Markov Model

Sequencia de palabras

$$(w_1, w_2, \ldots) =$$
$$(\text{Pedro}, \text{es}, \text{ingeniero})$$

Sequencia de etiquetas

$$(t_1, t_2, \ldots) =$$
$$(\text{Sustantivo}, \text{Verbo}, \text{Sustantivo})$$



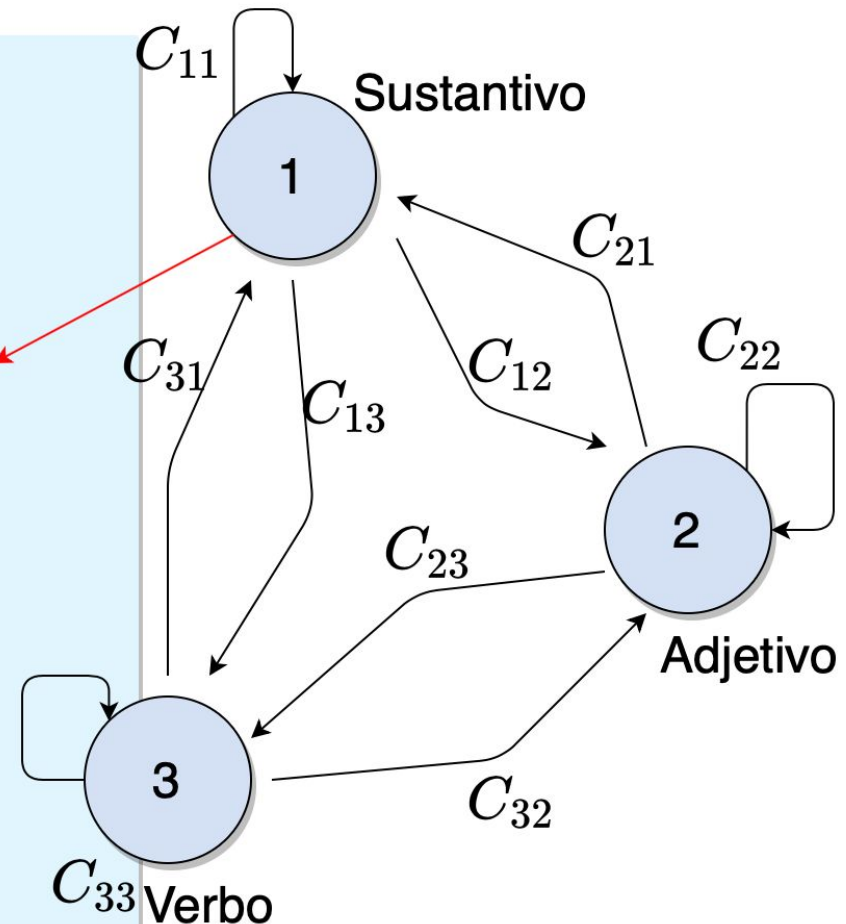
Modelos Markovianos Latentes

Probabilidades de emisión

$$(w_1, w_2, \ldots) \rightarrow B_i = P(w_i|t_i)$$

$$P(Pedro|Sustantivo)$$
$$P(es|Sustantivo)$$
$$\ldots$$

$$(t_1, t_2, \ldots) \rightarrow C_{ij} = P(t_i|t_{i-1})$$



Modelos Markovianos Latentes

Ingredientes de HMM

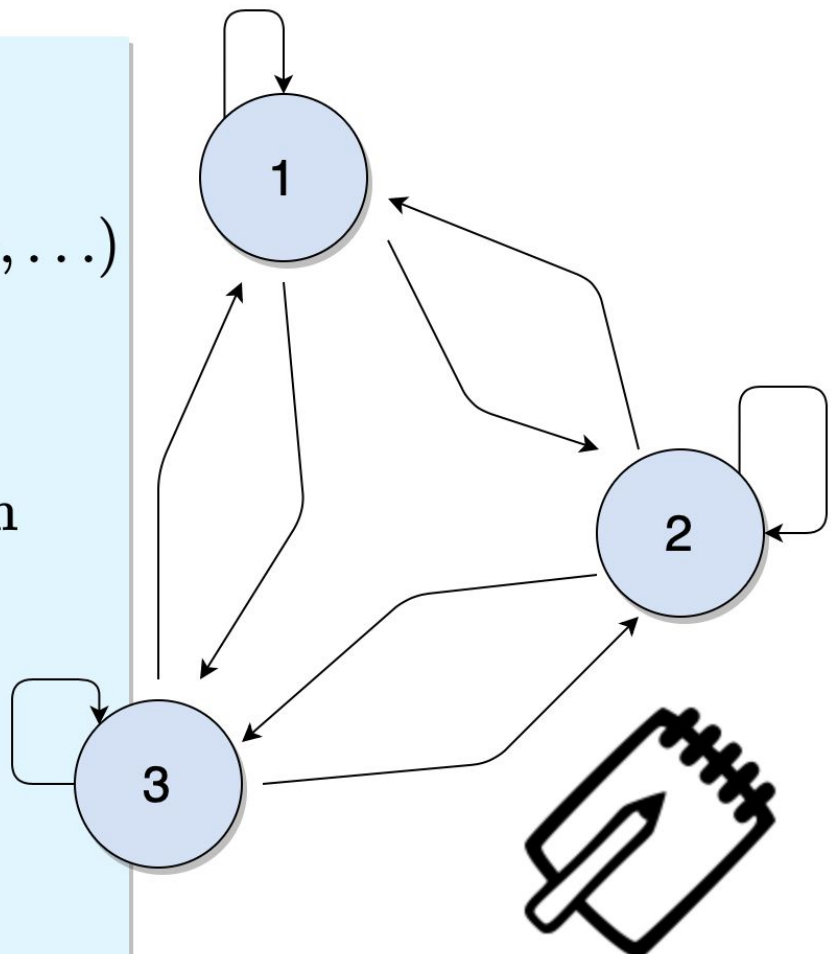
$(t_1, t_2, \ldots) = (\text{sustantivo}, \text{verbo}, \ldots)$

A : matriz transición

Π : distribución de estados

B : probabilidades de emisión

$$\arg\max_{(t^n)} P(t^n | w^n)$$



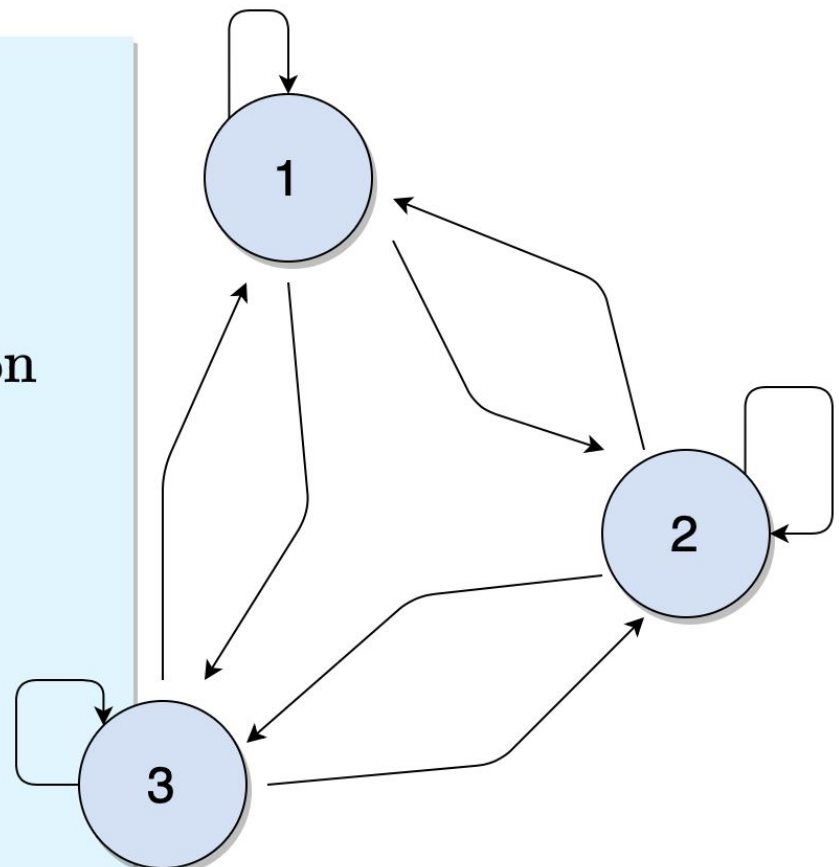
¿Que significa entrenar un HMM?

Aprender probabilidades ...

A : matriz transición

B : probabilidades de emisión

$$(A, B)$$



[C9] El algoritmo de Viterbi

Desambiguación y etiquetado
de palabras

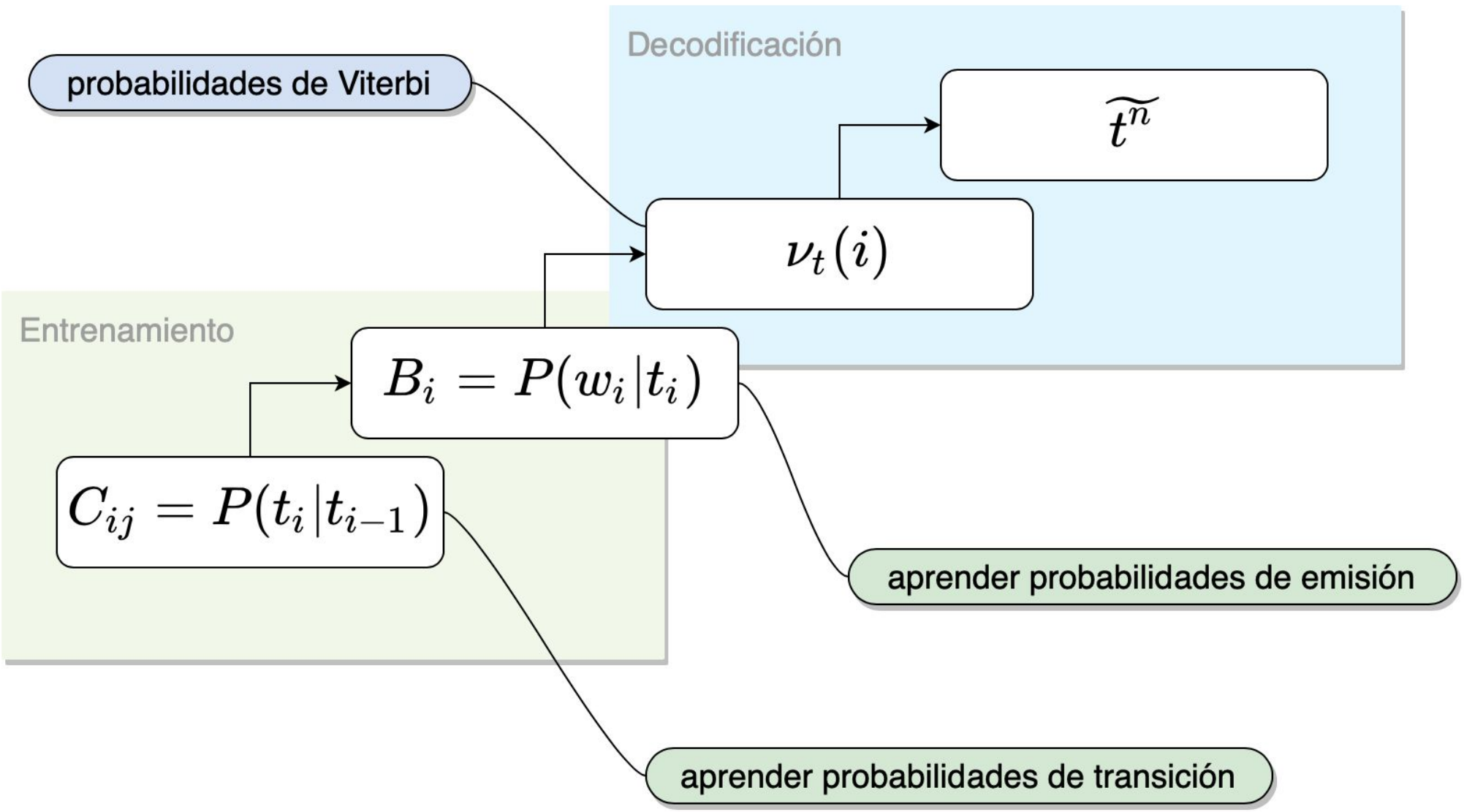
Etiquetado con HMM

- **Entrenamiento del HMM**

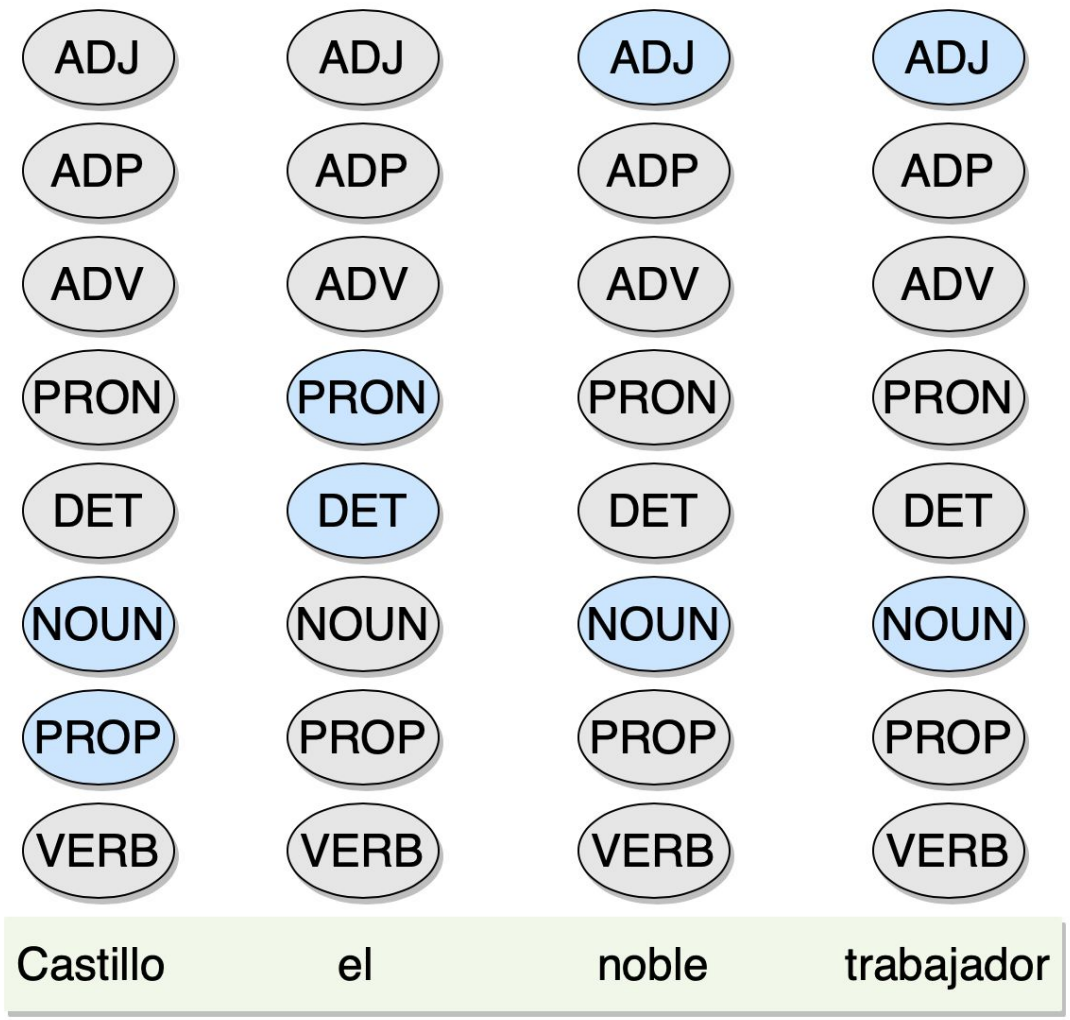
  El modelo aprende las probabilidades de emisión y transición.
- **Decodificación**
  El modelo calcula la secuencia de etiquetas más probable.

$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i \underbrace{P(w_i|t_i)}_{emisión} \overbrace{P(t_i|t_{i-1})}^{transición}$$

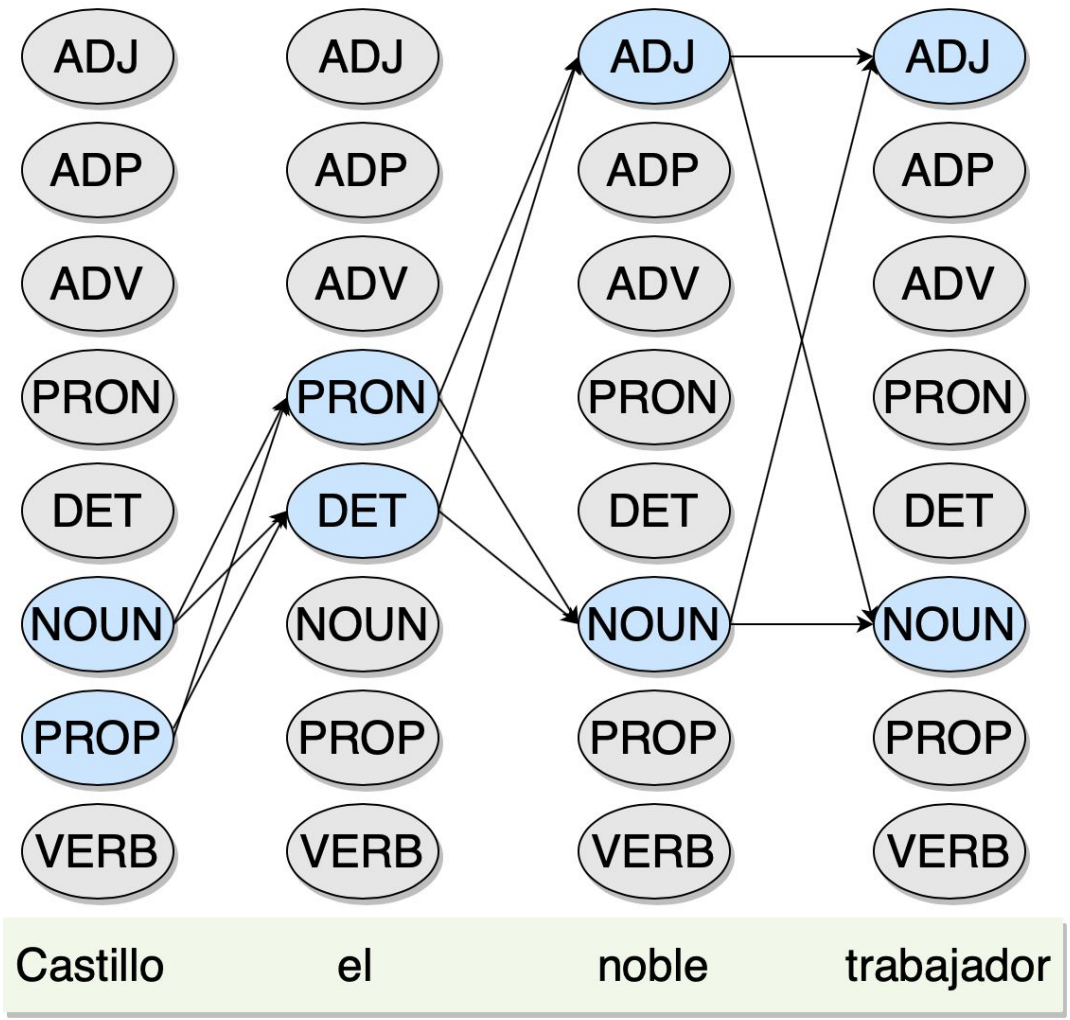
Etiquetado con HMM



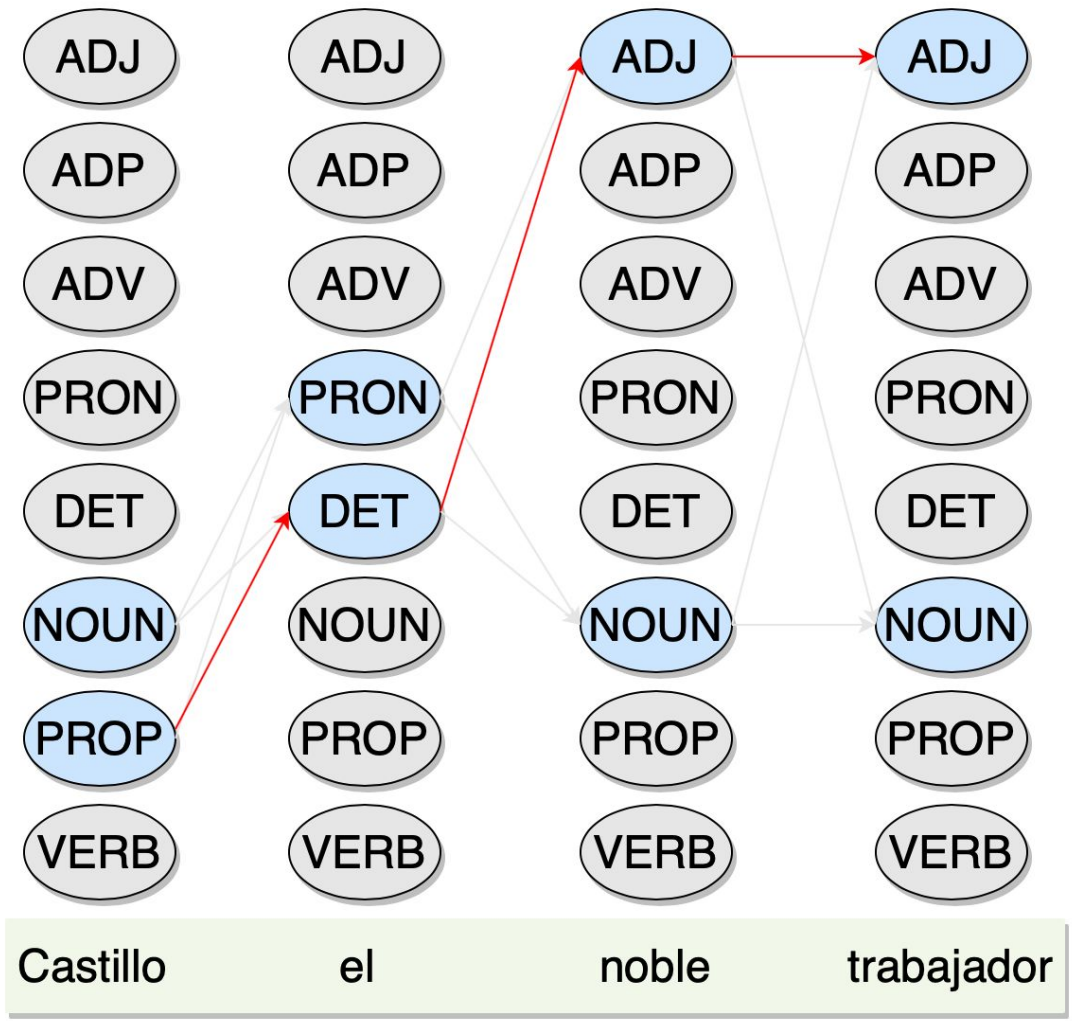
Decodificación de Viterbi



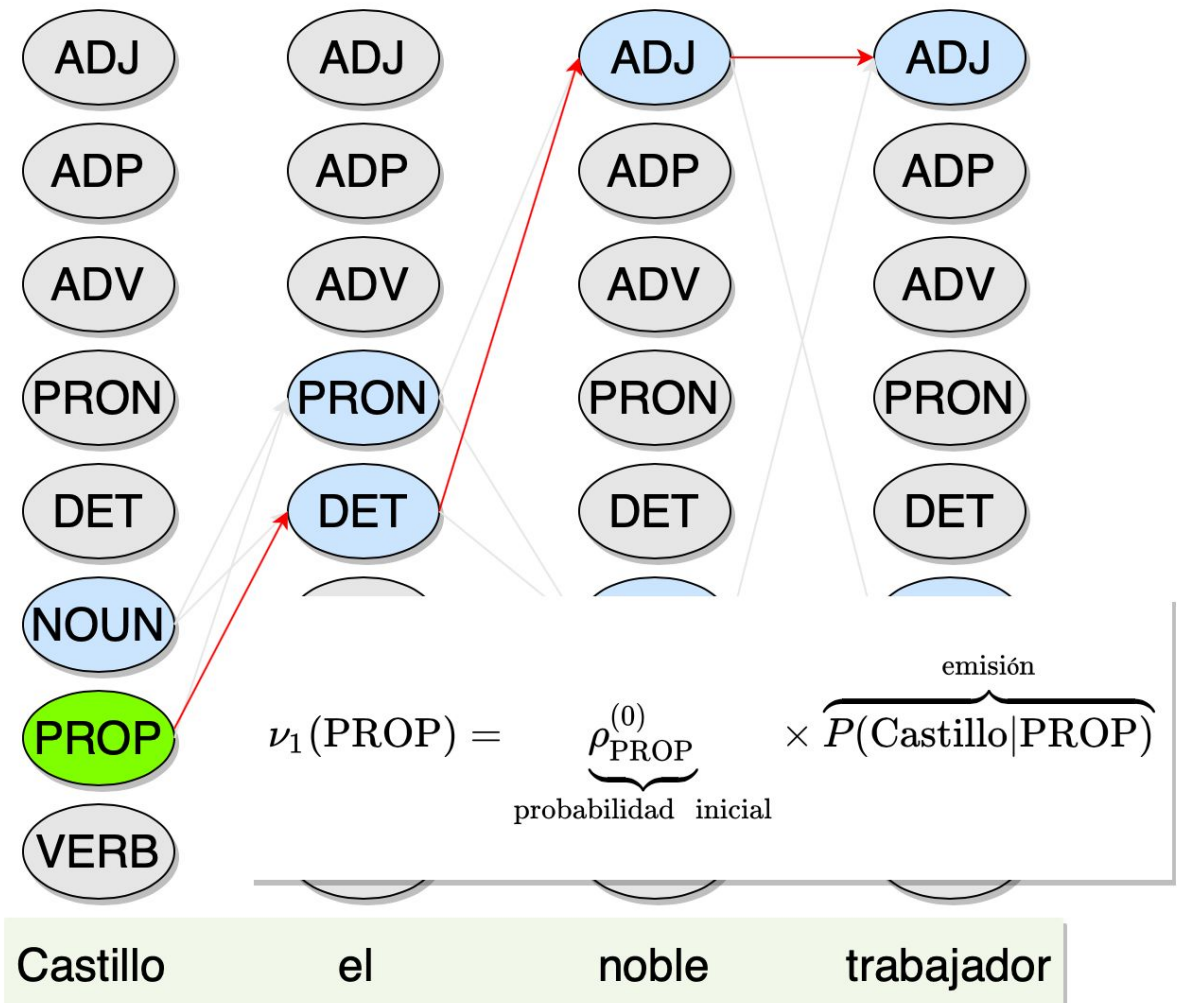
Decodificación de Viterbi



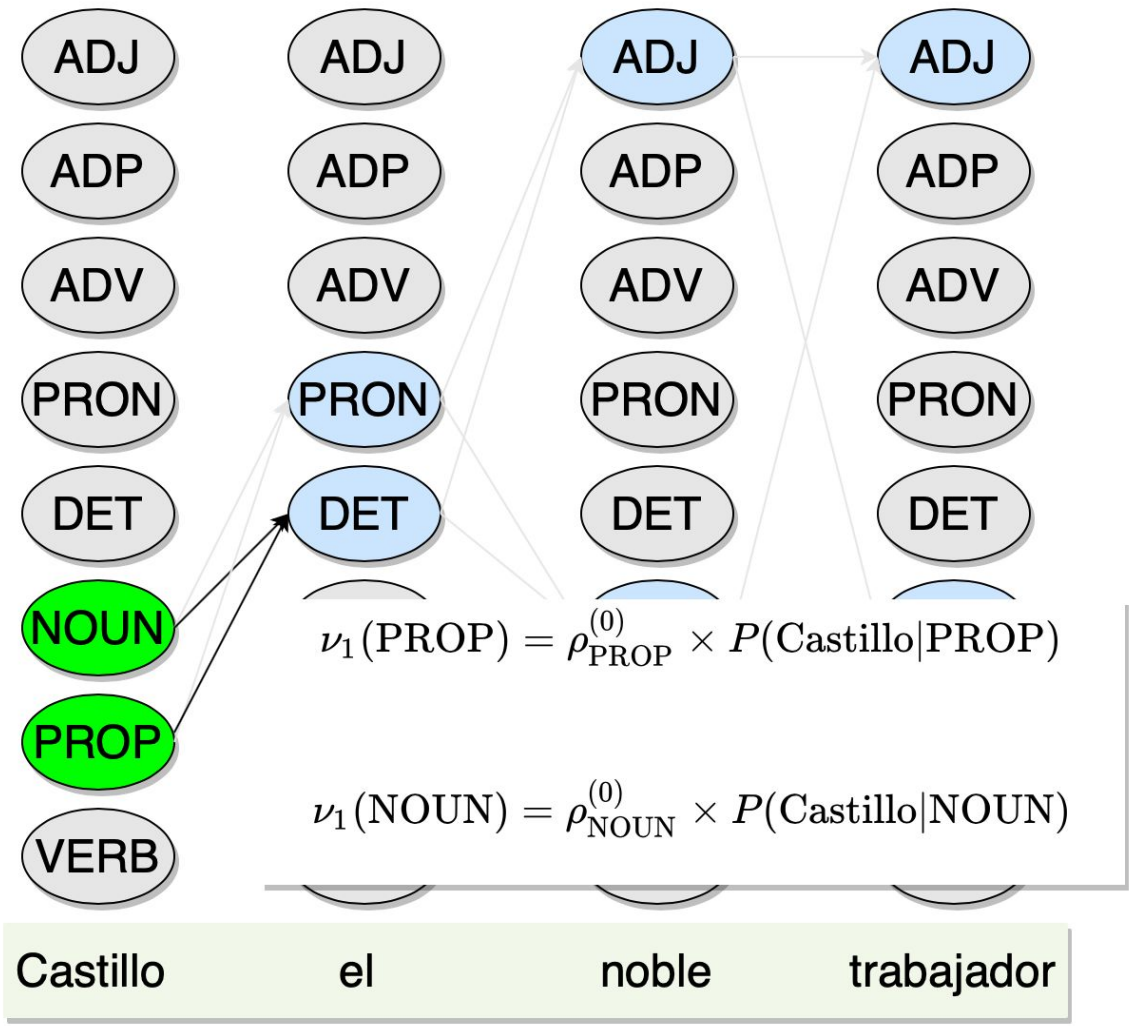
Decodificación de Viterbi



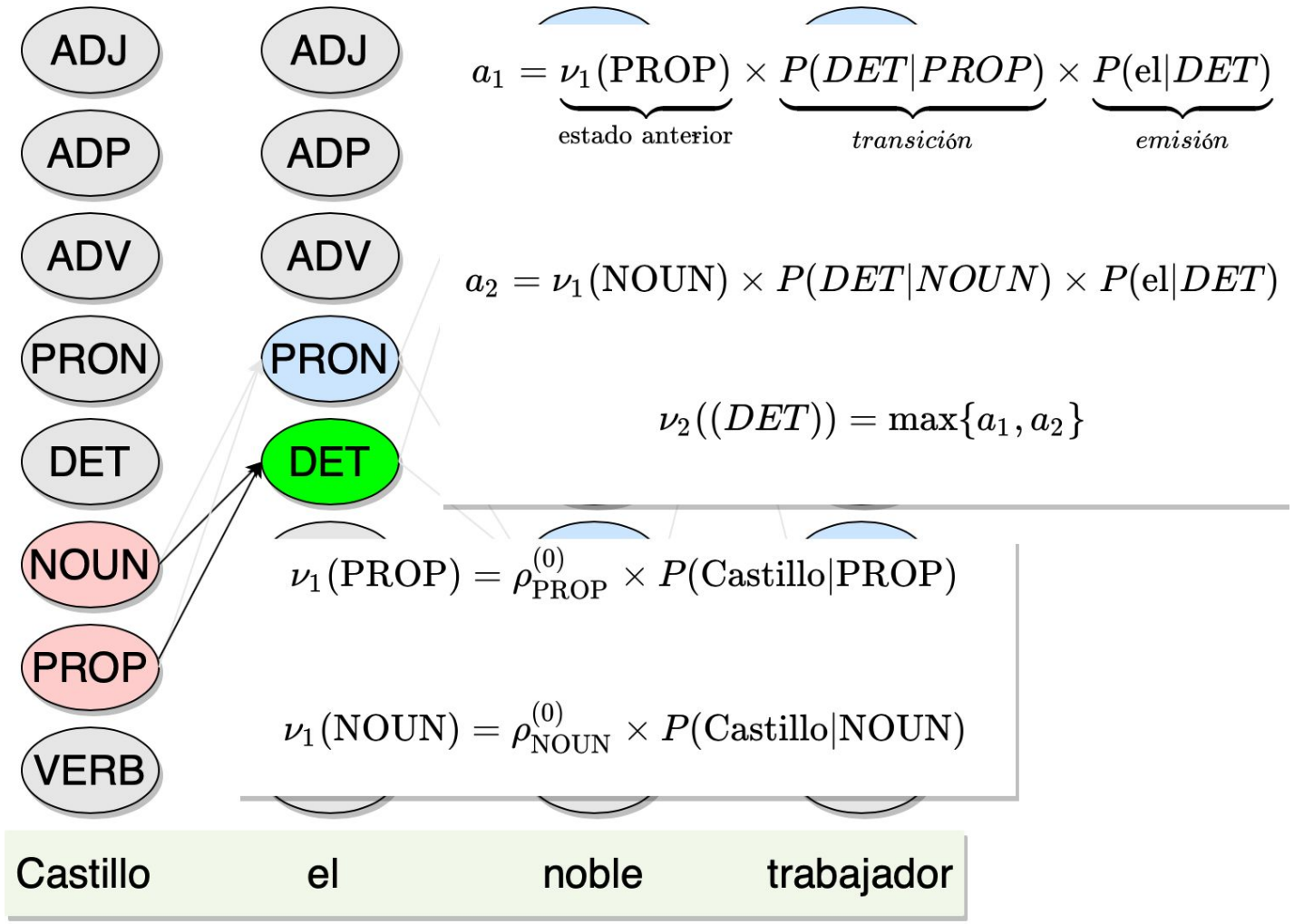
Decodificación de Viterbi



Decodificación de Viterbi



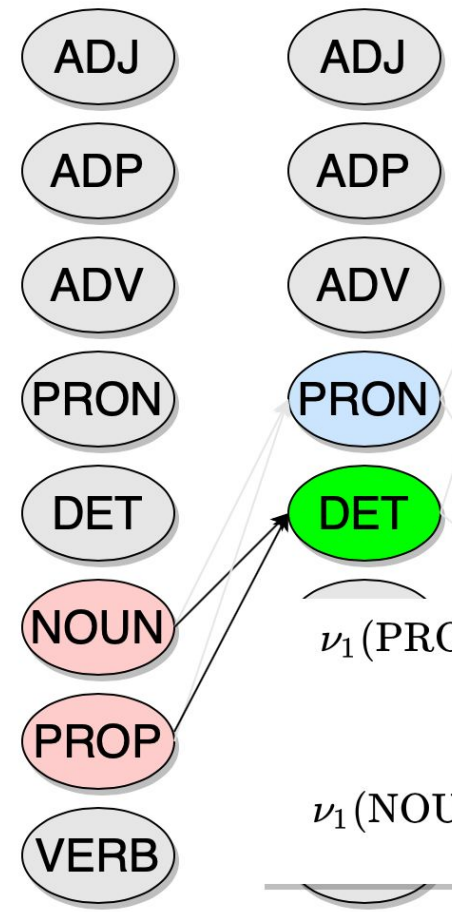
Decodificación de Viterbi



[C10] Cálculo de las probabilidades de Viterbi

Desambiguación y etiquetado de palabras

Decodificación de Viterbi



$$a_1 = \underbrace{\nu_1(\text{PROP})}_{\text{estado anterior}} \times \underbrace{P(DET|PROP)}_{\textit{transición}} \times \underbrace{P(\text{el}|DET)}_{\textit{emisión}}$$

$$a_2 = \nu_1(\text{NOUN}) \times P(DET|NOUN) \times P(\text{el}|DET)$$

$$\nu_2((DET)) = \max\{a_1, a_2\}$$

$$\nu_1(\text{PROP}) = \rho^{(0)}_{\text{PROP}} \times P(\text{Castillo}|\text{PROP})$$

$$\nu_1(\text{NOUN}) = \rho^{(0)}_{\text{NOUN}} \times P(\text{Castillo}|\text{NOUN})$$

Castillo el noble trabajador

Decodificación de Viterbi

$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i \underbrace{P(w_i|t_i)}_{emisión} \overbrace{P(t_i|t_{i-1})}^{transición}$$

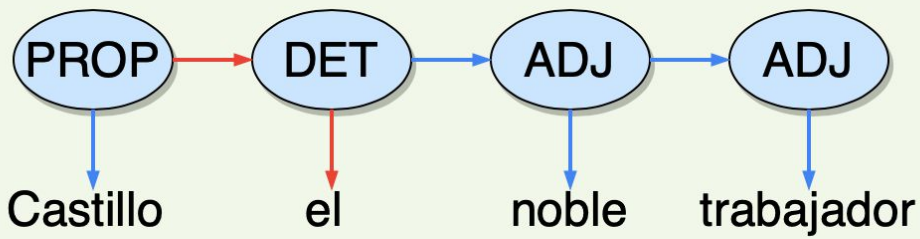
$$\nu_t(j) = \max_i \{\nu_{t-1}(i) \times C_{ij} \times P(\text{palabra}|j)\}$$

[C14] Modelos Markovianos de Máxima Entropía

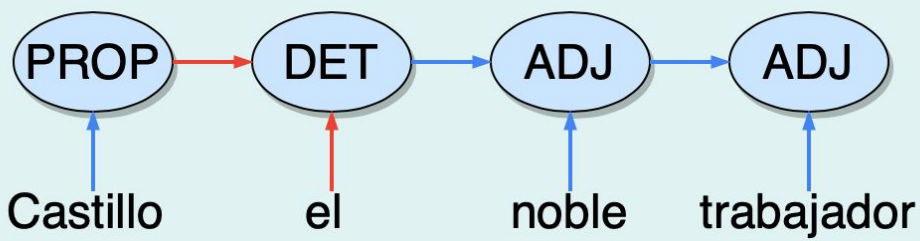
Desambiguación y etiquetado de palabras

HMM

$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(w_i|t_i)P(t_i|t_{i-1})$$



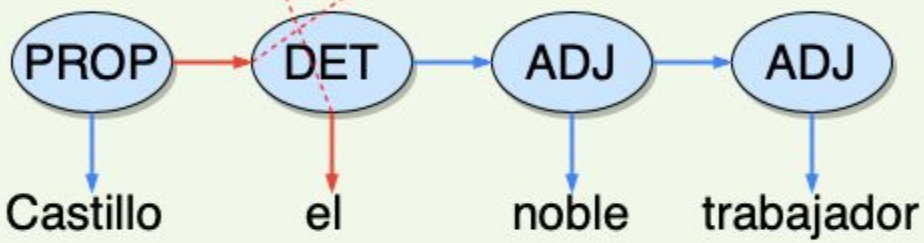
MEMM



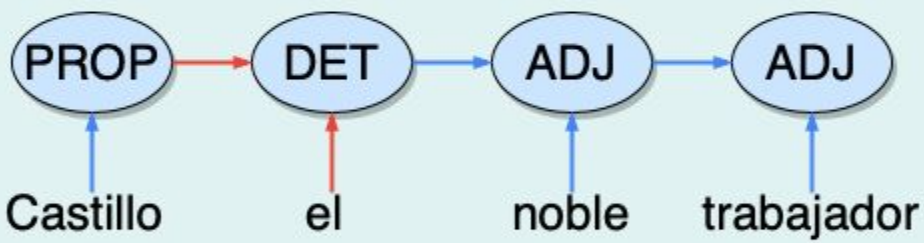
$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i|w_i, t_{i-1})$$

HMM

$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(w_i|t_i)P(t_i|t_{i-1})$$



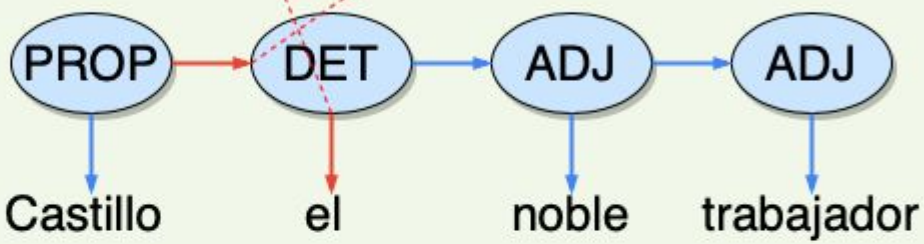
MEMM



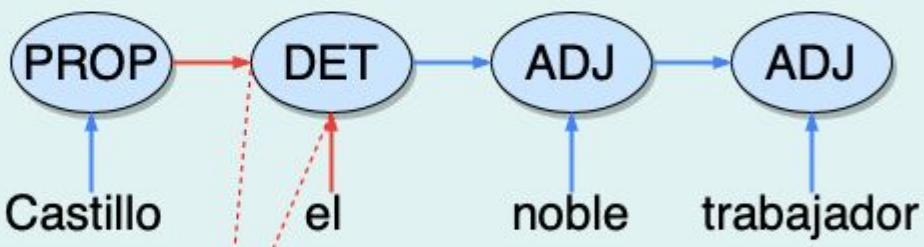
$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i|w_i, t_{i-1})$$

HMM

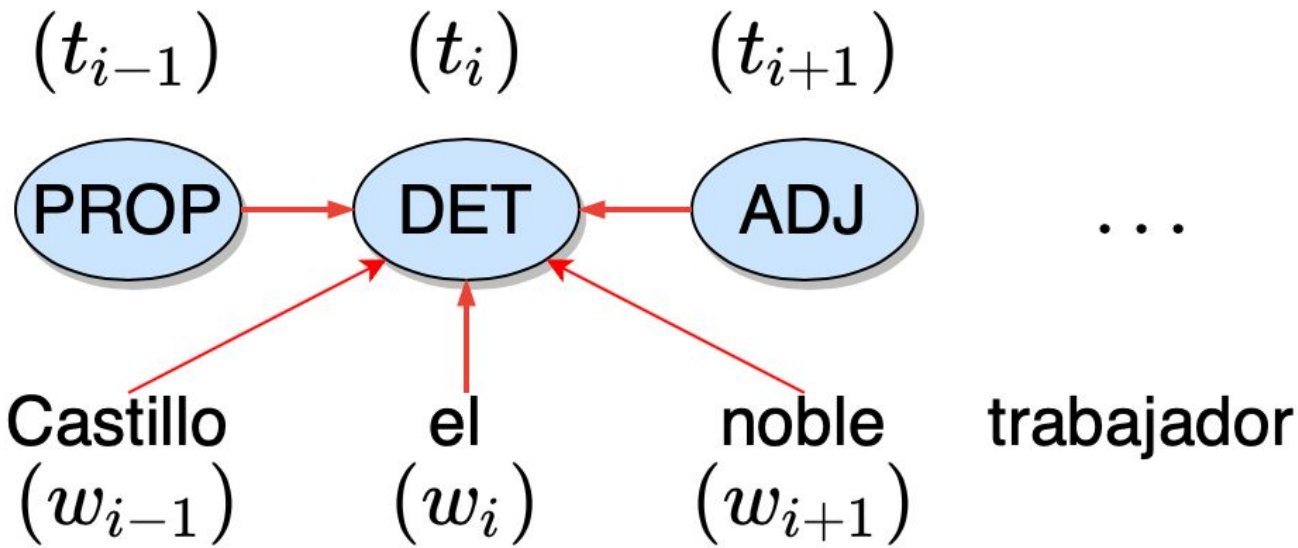
$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(w_i|t_i)P(t_i|t_{i-1})$$



MEMM



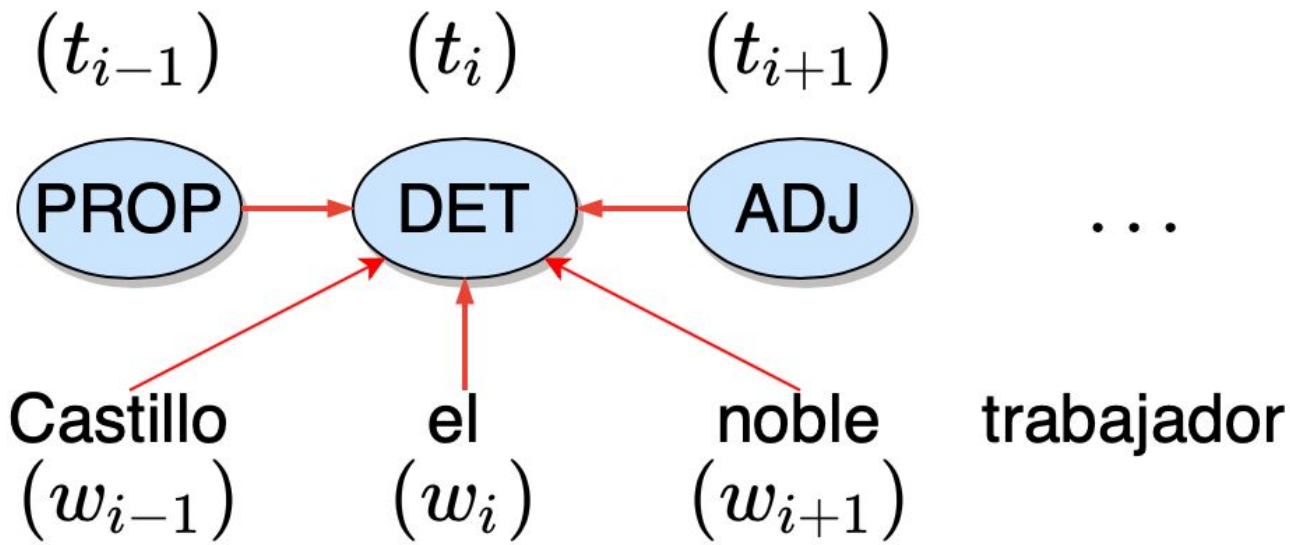
$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i|w_i, t_{i-1})$$



$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i | w_{i-1}, \ldots, w_{i+1}, t_{i-1}, t_{i+1})$$

[C15] Algoritmo de Viterbi para MEMM

Desambiguación y etiquetado
de palabras



$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i | w_{i-1}, \ldots, w_{i+1}, t_{i-1}, t_{i+1})$$

Decodificación de Viterbi

$$\widetilde{t^n} = \arg\max_{(t^n)} \prod_i P(t_i | w_i, t_{i-1}, \ldots)$$

$$\nu_t(j) = \max_i \{\nu_{t-1}(i) \times P(j | palabra, i, \ldots)\}$$

[C16] RETO 1: Construye un MEMM en Python

Desambiguación y etiquetado de palabras

```
import nltk
...

import pickle

...
```

“

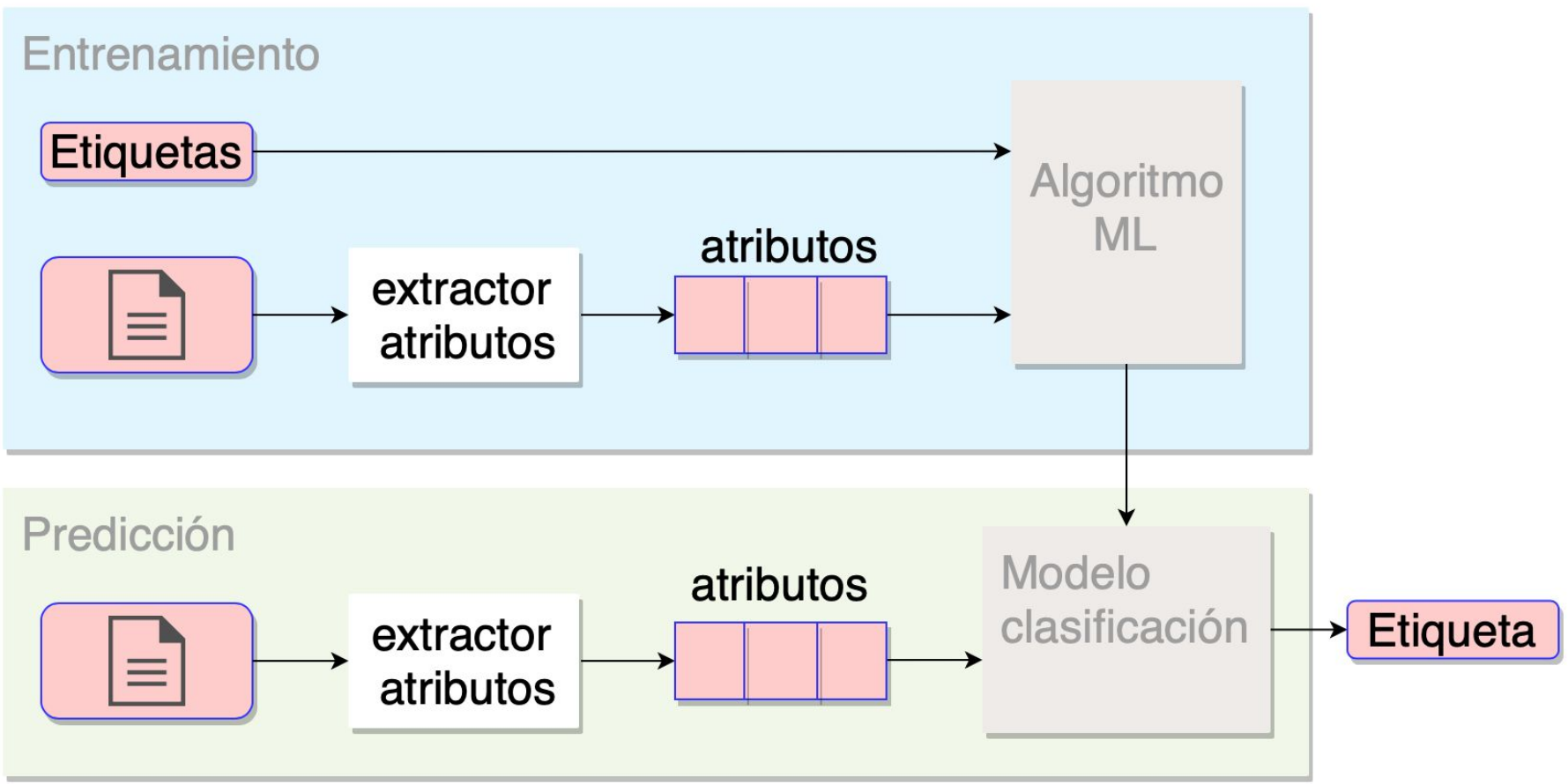
Vamos a un notebook en Google Colab ...

”

[C17] El problema general de la clasificación de texto

Clasificación de texto

Clasificación: ML supervisado



Técnicas de clasificación

- Basadas en teoría de la probabilidad
- Basadas en teoría de la información
- Basadas en espacios vectoriales

Clasificación de palabras

- Identificación de género de nombres
- Etiquetado POS (categorías gramaticales)
- Bloqueo de palabras ofensivas

Clasificación de documentos

- Análisis de sentimiento
- Tópicos de conversación
- Priorización en CRMs

[C18] Tareas de clasificación con NLTK

Clasificación de texto

Clasificación de género

Por palabra

```
import nltk, random
from nltk.corpus import names

def atributos(palabra):
   return {'letra_f': palabra[-1]}

tagset = [
 (n, 'M'/'F') for n in names.words('.txt')
     ]

random.shuffle(tagset)
...
```

```
...

fset = [
(atributos(n),'g') for (n, 'g') in tagset)
]

train, test = fset[num:], fset[:num]

classifier =
     nltk.NaiveBayesClassifier.train(train)
```

Ingeniería de atributos

¿Cómo escoger los atributos relevantes de un nombre?

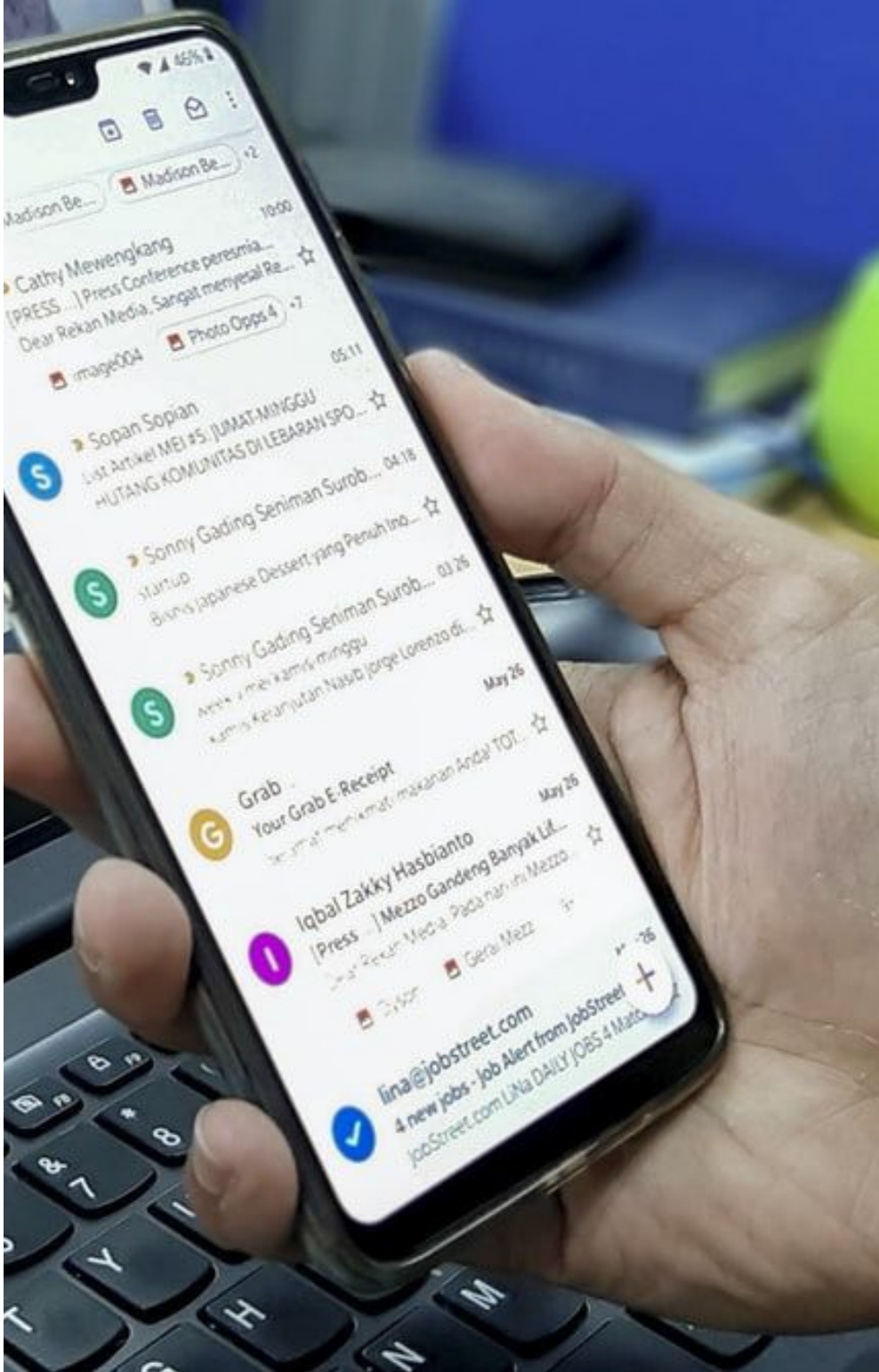
```
import nltk, random
from nltk.corpus import names

def atributos(palabra):
    '???'
    return {'atr_1': ?,'atr_2': ?, ...}

...
```

Clasificación de emails

Por documento



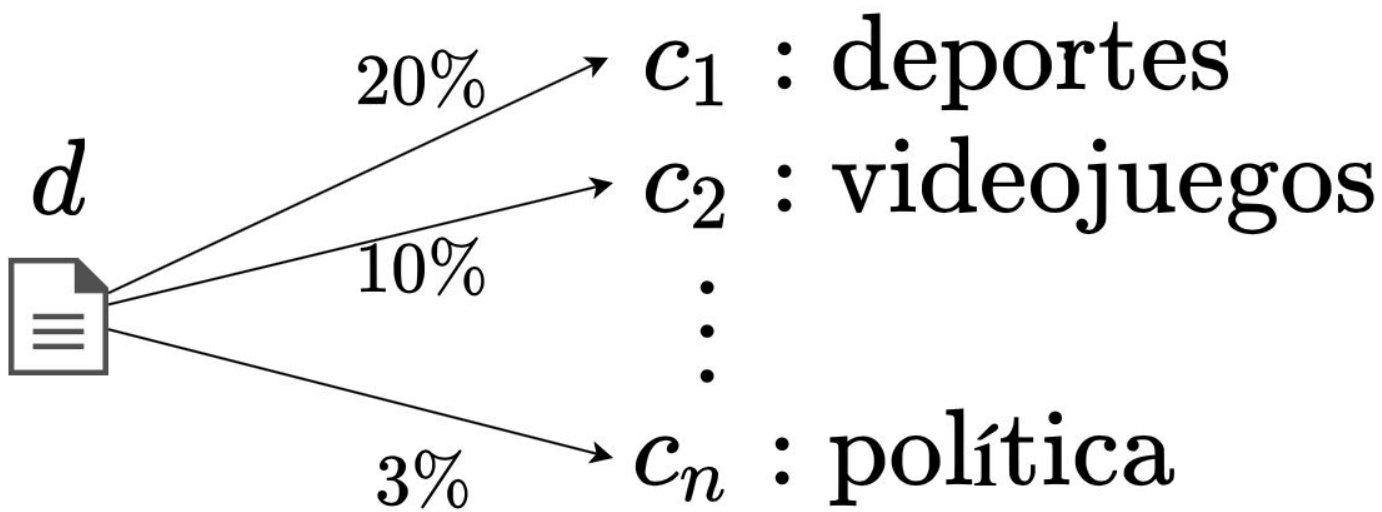
¡Alto!

Hasta el final del curso

[C21] Naive Bayes

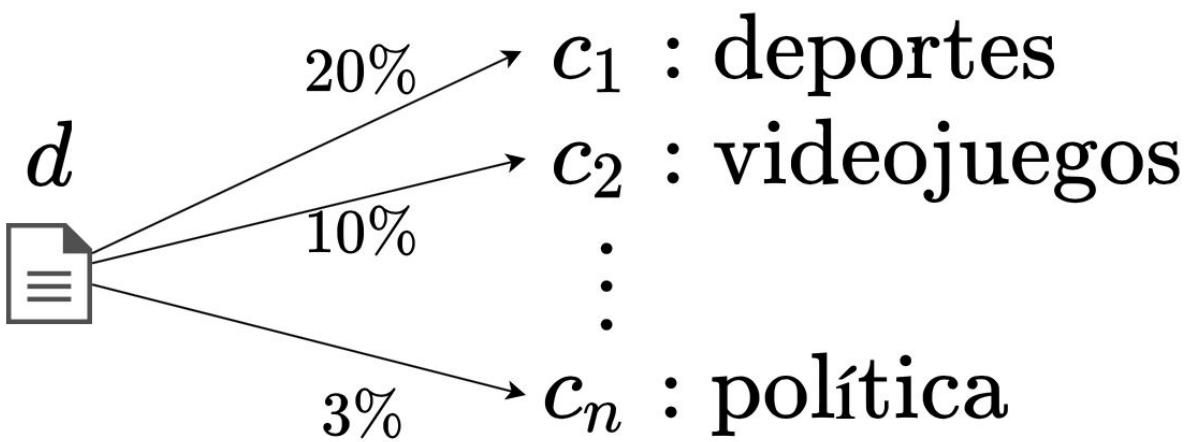
Clasificación de texto

NB: clasificador probabilístico



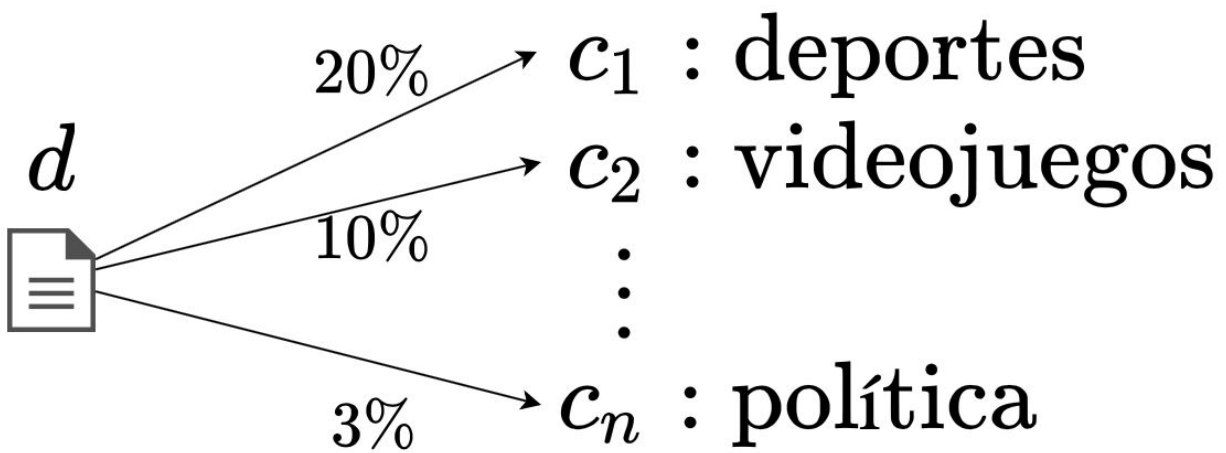
$$\hat{c} = \arg\max_{(c)} P(c|d)$$

NB: clasificador probabilístico



$$\hat{c} = \arg\max_{(c)} \underbrace{\frac{P(d|c)P(c)}{P(d)}}_{\text{Regla de Bayes}}$$

NB: clasificador probabilístico

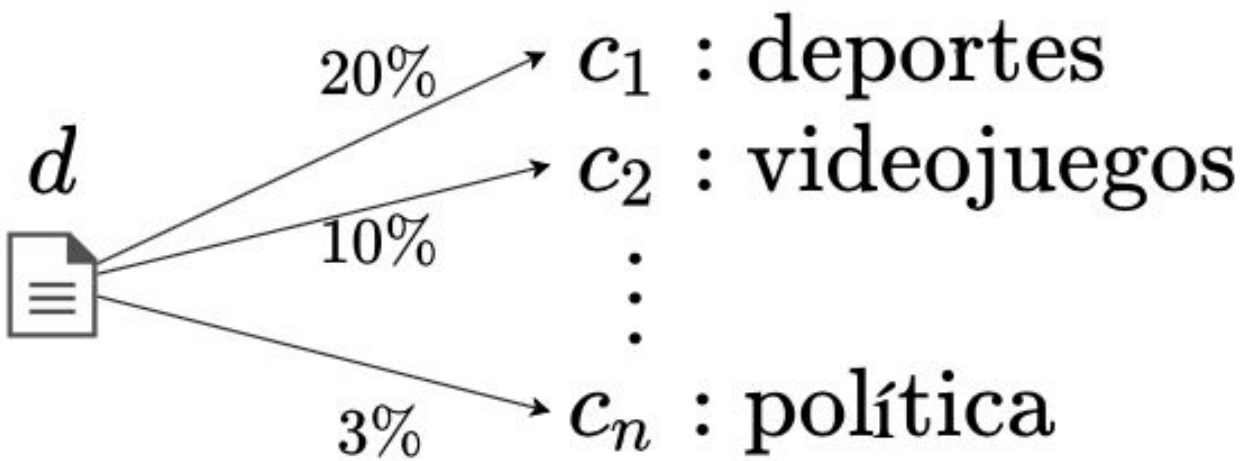


$$\hat{c} = \arg\max_{(c)} \underbrace{P(d|c)P(c)}_{P(d) \text{ igual para toda clase}}$$

[C23] Naive Bayes en python: construcción del modelo

Clasificación de texto

NB: clasificador probabilístico



$$\hat{c} = \arg\max_{(c)} \log P(c) + \sum_{i=1}^{n} \log P(f_i|c)$$

“

Vamos a un notebook en Google Colab ...

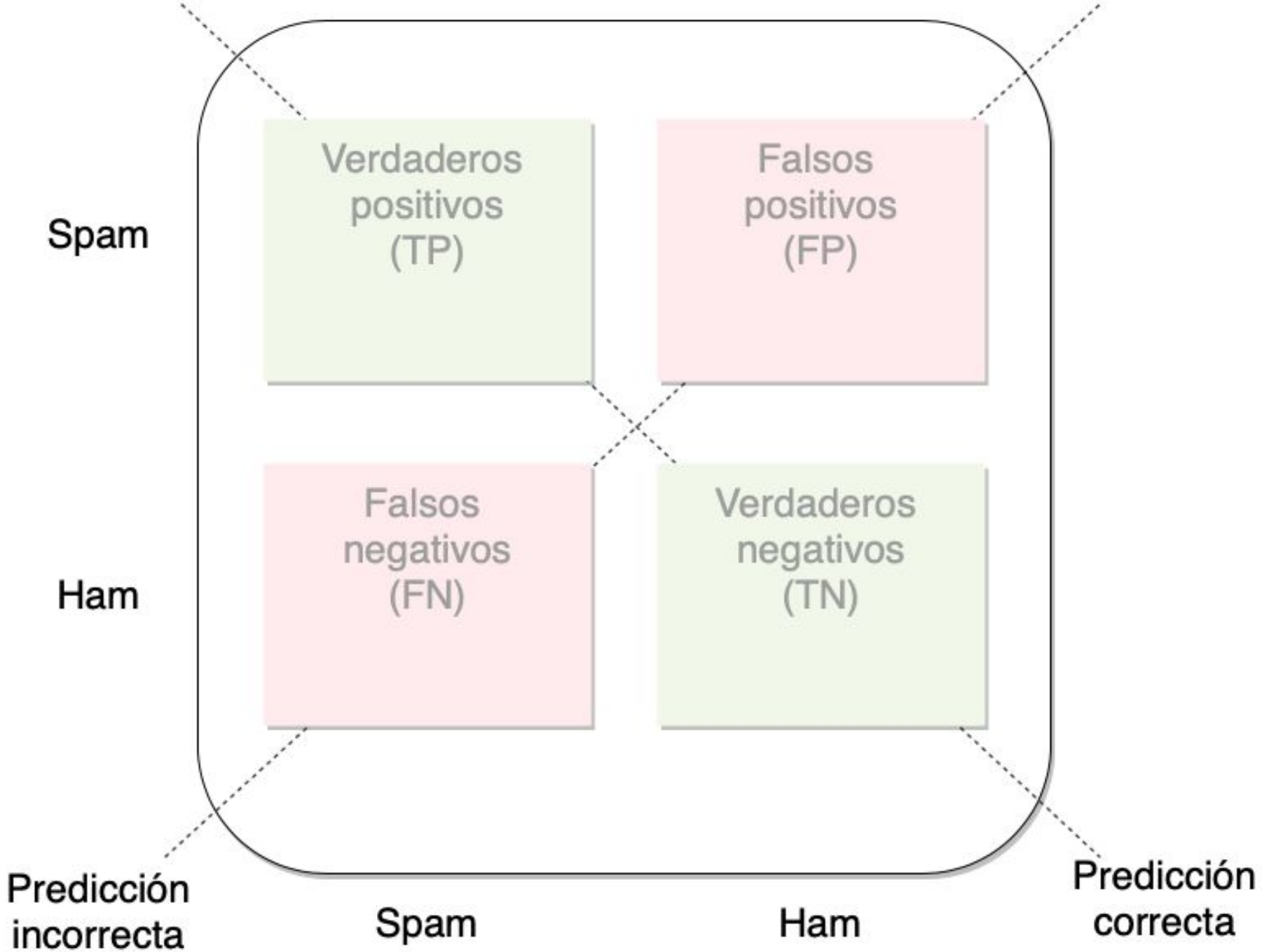
”

[C25] Métricas para algoritmos de clasificación

Clasificación de texto

Accuracy

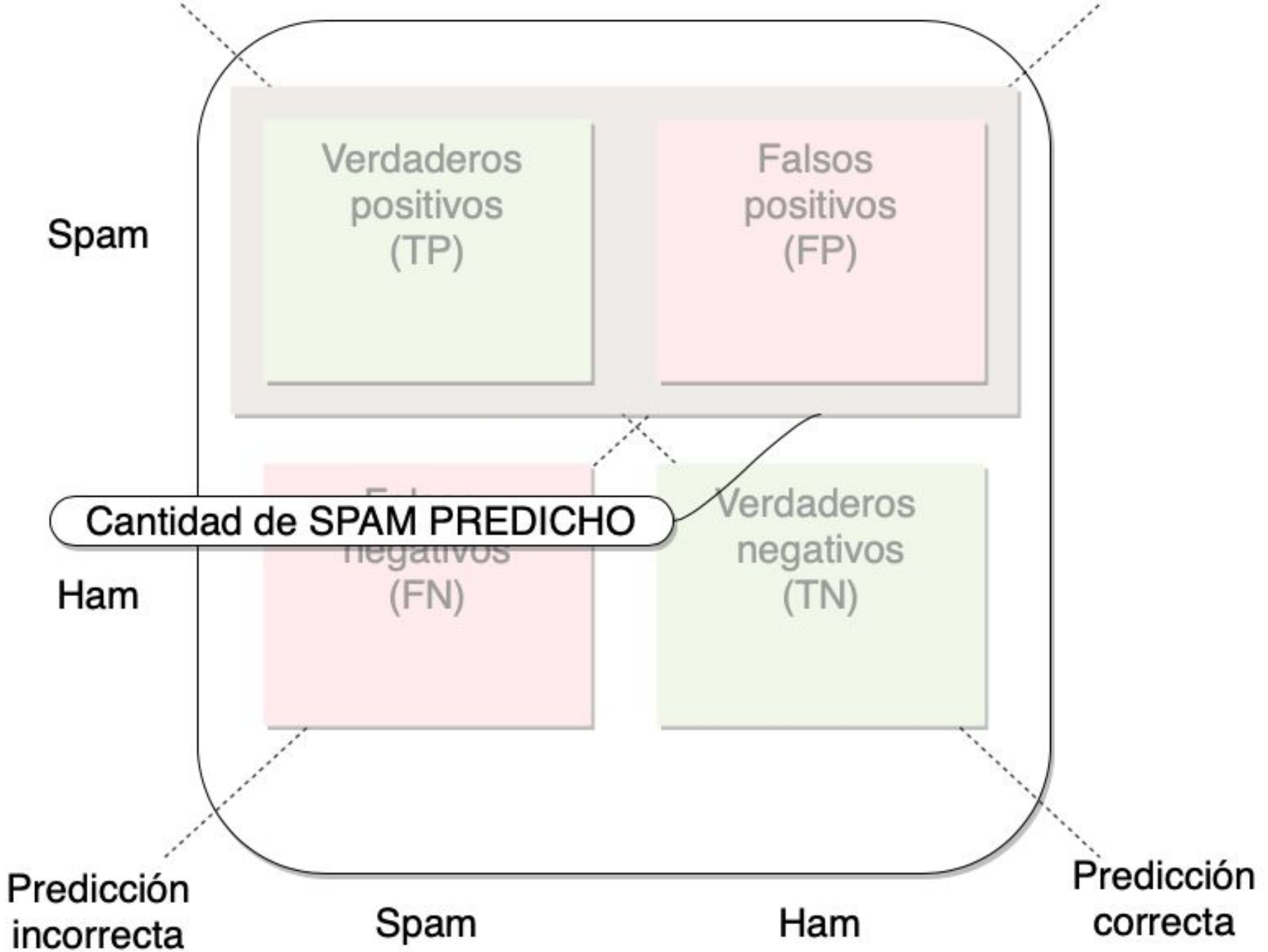
$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i, \hat{y}_i}$$


Verdaderos positivos (TP)
Falsos positivos (FP)
Falsos negativos (FN)
Verdaderos negativos (TN)
Spam
Ham
Predicción incorrecta
Spam
Ham
Predicción correcta

Precision

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

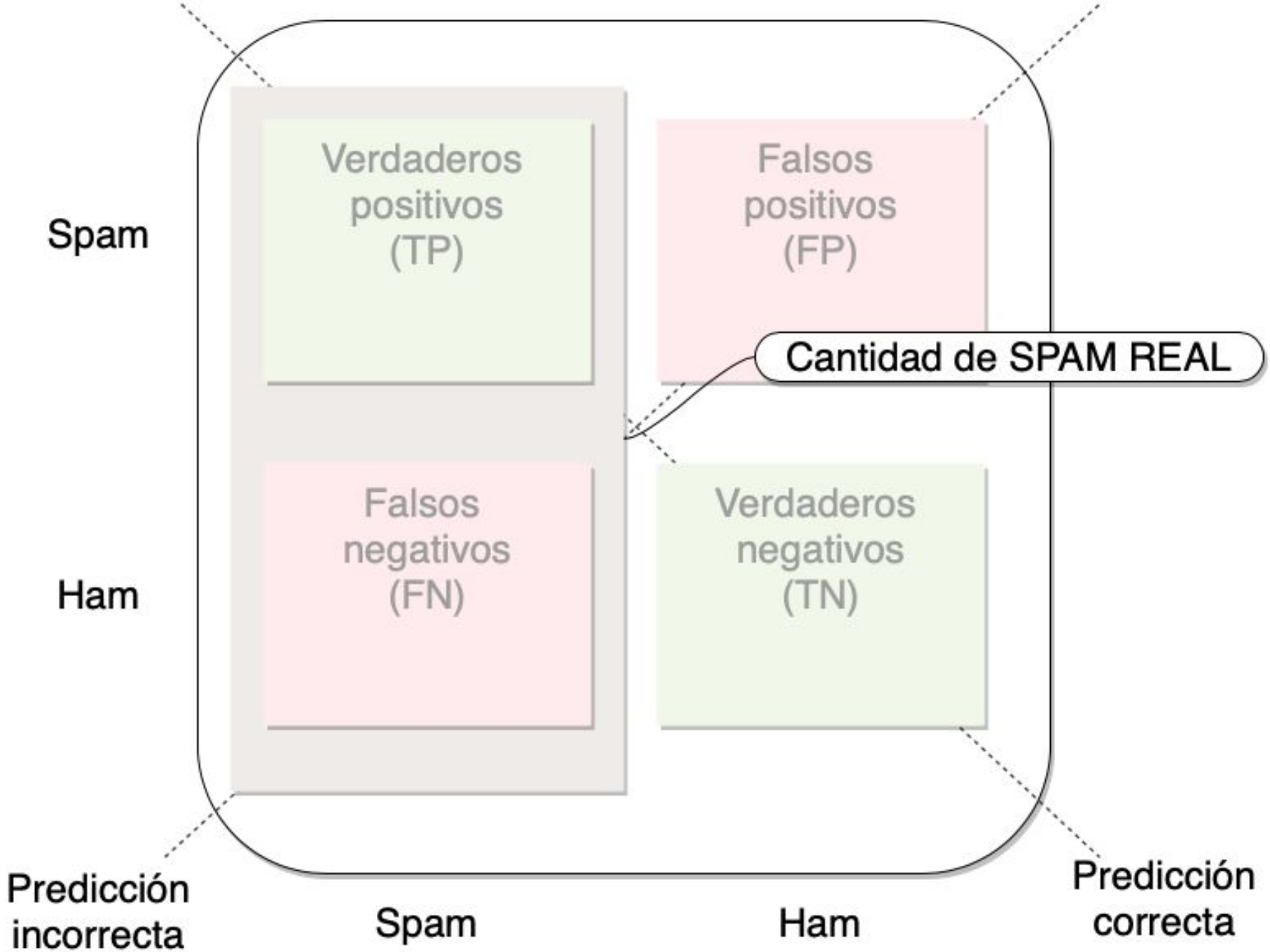
¿Cuántos de los correos clasificados como spam, realmente lo son?


Verdaderos positivos (TP)
Falsos positivos (FP)
Spam
Cantidad de SPAM PREDICHO
Verdaderos negativos (TN)
Ham
Predicción incorrecta
Spam
Ham
Predicción correcta

Recall

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

¿Cuántos correos SPAM logramos identificar?


Verdaderos positivos (TP)
Falsos positivos (FP)
Cantidad de SPAM REAL
Falsos negativos (FN)
Verdaderos negativos (TN)
Spam
Ham
Predicción incorrecta
Spam
Ham
Predicción correcta

“

Vamos a un notebook en Google Colab ...

”

[C26] Reto Final: Construye un modelo de sentimiento

Clasificación de texto

```
from nltk.corpus import movie_reviews as
mr

docs =[
   (list(mr.words(fileid)), category)
    for category in mr.categories()
    for fileid in mr.fileids(category)]

random.shuffle(docs)
```

```
from nltk.corpus import movie_reviews as
mr

all_words = nltk.FreqDist(
   w.lower() for w in mr.words()
)

word_features = list(all_words)[:2000]

...
```

Clasificación de sentimiento

- Dataset: http://www.cs.jhu.edu/~mdredze/datasets/sentiment/

- Naive Bayes, Árboles de decisión, Máxima entropía, etc.

Imágenes de uso libre

unsplash.com

pexels.com

Pixabay.com

Íconos de uso libre

thenounproject.com

flaticon.com

Gifs

giphy.com

Íconos de Redes sociales