# Maximum d'entropie appliqué aux données textuelles

El Mehdi Issouani
joint works with Patrice Bertail

Laboratoire **LMAC** - Université de technologie de Compiègne **(UTC)**

**Entropies et divergences : modélisation . statistique . algorithmique**

Tuesday 14th May, 2024

# Natural Language Processing (NLP)

1. Tokenization
2. Part Of Speech Tagging (**POS tagging**)

## Natural Language Processing

### Tokenization example

*[He called Mr. Green at 2 p.m. in St. Louis, Mr. White did not answer. He then left him a voice mail message.]*

- Sentence tokenization

  *[He called Mr. Green at 2 p.m. in St. Louis, Mr. White did not answer.]*
  *[He then left him a voice mail message.]*

- Word tokenization

  *[He], [then], [left], [him], [a], [voice], [mail], [message], [.]*

# Natural Language Processing

## POS tagging examples

| Time | flies | like | an | arrow | . |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| NN | VB | IN | DT | NN | . |

| Fruit | flies | like | a | banana | . |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| JJ | NN | VB | DT | NN | . |

**VB**: Verb - **DT**: Determiner - **NN**: Noun singular or mass - **IN**: Preposition - **NNS**: Noun plural - **JJ**: Adjective

## Notations and vocabulary

- We denote a sentence $s$ with $N_s$ words

$$s = w_1, \dots, w_i, \dots, w_{N_s}$$

  where $w_i$ represents the $i$ th word and $N_s \in \mathbb{N}^*$ is random.
- $t_i$ is the tag of the $i$ th word with $t_i \in \mathrm{T} = \{NN, NNS, VB, \dots\}$ representing the tagset of finite size (example: 7, 36, 87)
- $n$ : Number of words in the whole dataset $n = \sum_{s \in \mathcal{S}} N_s$

$$\begin{tikzcd}
 & w_1 & w_2 & \dots & w_{N-1} & w_N & . \\
\text{sent} \rightarrow & \text{Time} \arrow[u] \arrow[d, leftrightarrow] & \text{flies} \arrow[u] \arrow[d, leftrightarrow] & \text{like} \arrow[u] \arrow[d, leftrightarrow] & \text{an} \arrow[u] \arrow[d, leftrightarrow] & \text{arrow} \arrow[u] \arrow[d, leftrightarrow] & . \arrow[u] \arrow[d, leftrightarrow] \\
\text{tags} \rightarrow & \text{NN} \arrow[d] & \text{VB} \arrow[d] & \text{IN} \arrow[d] & \text{DT} \arrow[d] & \text{NN} \arrow[d] & . \arrow[d] \\
 & t_1 & t_2 & \dots & t_{N-1} & t_N & .
\end{tikzcd}$$

## Mathematical Models for POS Tagging

- POS Tagging task can be considered as a **classification** problem.

$$\begin{cases} cl \; : \quad \mathcal{X} \longrightarrow \boldsymbol{Y} \\ \qquad\quad x \longmapsto y \end{cases}$$

So for a given sentence $w_1, ..., w_N$ (containing a random number of words $N$), the goal of POS tagging is to find the best tag sequence $t_1^*, ..., t_N^*$.

## Mathematical Models for POS Tagging

- Goal : we aim to obtain the best tag sequence

$$(t_1^*, \ldots, t_N^*) = \underset{(t_1,\ldots,t_N)\in \mathrm{T}^{\mathrm{N}}}{\arg\max} \left[p\left(t_1, \ldots, t_N \mid w_1, \ldots, w_N\right)\right]$$

- Denote $x_1, \ldots, x_N$ the contexts of each word

$$p\left(t_1, \ldots, t_N \mid w_1, \ldots, w_N\right) = \prod_{i=1}^{N} p\left(t_i \mid x_i\right)$$

## Mathematical Models for POS Tagging

- Goal : we aim to obtain the best tag sequence

$$\left(t_1^*, \ldots, t_N^*\right) = \underset{(t_1, \ldots, t_N) \in \mathrm{T}^{\mathrm{N}}}{\arg\max} \left[p\left(t_1, \ldots, t_N \mid w_1, \ldots, w_N\right)\right]$$

- Denote $x_1, \ldots, x_N$ the contexts of each word

$$p\left(t_1, \ldots, t_N \mid w_1, \ldots, w_N\right) = \prod_{i=1}^{N} p\left(t_i \mid x_i\right)$$

## Feature-Based Models

Features:

- Définition : encode le lien binaire qu'il y a entre le tag et le contexte (ou environnement du mot)
- Les features permettrent d'organiser le corpus et d'extraire les informations contextuelles utiles à la tâche considérée
- Exemple :

$$
\begin{array}{cccccccc}
\text{features:} & \dots & \dots & \dots & (\text{flies, NNS}) & (\text{flies, VB}) & \dots & \dots \\
 & \downarrow & & & \downarrow & \downarrow & \downarrow & \downarrow \\
\text{f(flies, NNS)} & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
f(\text{ flies, VB}) & 0 & \dots & 0 & 0 & 1 & \dots & 0 \\
f(\text{ flies, NN}) & 0 & \dots & 0 & 0 & 0 & \dots & 0
\end{array}
$$

## Maximum entropy principle (MaxEnt)

A method to infer a measure $p(z)$ defined on a given set $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ under some constraint $\mathcal{P}$.

$$p^* = \arg\max_{p \in \mathcal{P}} \{\mathcal{H}(p)\} = \arg\max_{p \in \mathcal{P}} \left\{ -\int p(z) \log p(z) l(dz) \right\},$$

$$\text{where } \mathcal{P} = \left\{ p : \int f_k(z) p(z) l(dz) = \mu_k, k = 1, \ldots, q \right\}$$

.

## Link to divergences

If we have access to a given default distribution $p_0 \in \mathcal{P}$

$$\begin{aligned} p^* = \arg\max_{p\in\mathcal{P}}\{\mathcal{H}(p)\} = &\quad \arg\max_{p\in\mathcal{P}}\{\mathcal{H}(p_0) - D(p, p_0)\} \\ = &\quad \arg\min_{p\in\mathcal{P}}\{D(p, p_0)\} \end{aligned}$$

$$\text{where}\quad D(p, p_0) = \int \left[ p(z) \log \frac{p(z)}{p_0(z)} - p(z) + p_0(z) \right] l(dz)$$

.

## MaxEnt solution

Consider $\lambda_i$'s the Kuhn & Tucker coefficients in the optimization program with each $\lambda_i$ corresponding to a constraint $\mu_i$,

$$p(z) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(z)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(u)\right) l(du)}.$$

## MaxEnt solution

Consider $\lambda_i$'s the Kuhn & Tucker coefficients in the optimization program with each $\lambda_i$ corresponding to a constraint $\mu_i$,

$$p(z) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(z)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(u)\right) l(du)}.$$

$\longrightarrow$ Notice that $p(z)$ doesn't depend on $\mu_k$.

## Back to POS Tagging

By putting $z = (x, t)$ we can rewrite

### Log-linear models

$$p(z) = p(x,t) = p(t \mid x)p(x) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(x,t)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k(u)\right) l(du)},$$

### Conditional probabilities

$$p(t \mid x) = \frac{\exp\left(\sum_{k=1,..,q} \lambda_k f_k(x,t)\right)}{\int \exp\left(\sum_{k=1,..,q} \lambda_k f_k\left(x,t'\right)\right) l\left(dt'\right)}$$

## POS Tagging solution

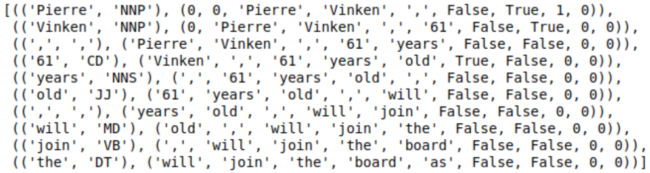
### POS Tagging solution

$$\hat{p}\left(t_i|x_i\right) = \frac{e^{-\hat{\lambda}' f\left(x, t_i\right)}}{\sum\limits_{t_i \in \mathcal{T}} e^{-\hat{\lambda}' f\left(x, t_i\right)}}, \text{ where } \hat{\lambda} \text{ is obtained by MEL or by GEL.}$$

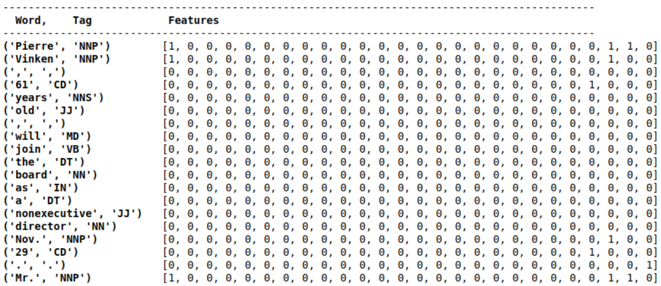
En pratique, dans la construction des modèles utilisant le MaxEnt pour le POS Tagging, ils ne calculent pas la solution analytique, mais s'arrêtent plutôt à cette étape et estiment $\lambda$ par la méthode du maximum de vraisemblance.

## Application to Penn Treebank Corpus

- Corpus : **Penn Treebank Corpus**
- Number of sentences : **3914**



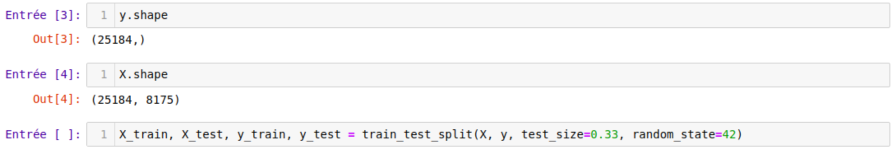
```
[(('Pierre', 'NNP'), (0, 0, 'Pierre', 'Vinken', ',', False, True, 1, 0)),
 (('Vinken', 'NNP'), (0, 'Pierre', 'Vinken', ',', '61', False, True, 0, 0)),
 ((',', ','), ('Pierre', 'Vinken', ',', '61', 'years', False, False, 0, 0)),
 (('61', 'CD'), ('Vinken', ',', '61', 'years', 'old', True, False, 0, 0)),
 (('years', 'NNS'), (',', '61', 'years', 'old', ',', False, False, 0, 0)),
 (('old', 'JJ'), ('61', 'years', 'old', ',', 'will', False, False, 0, 0)),
 ((',', ','), ('years', 'old', ',', 'will', 'join', False, False, 0, 0)),
 (('will', 'MD'), ('old', ',', 'will', 'join', 'the', False, False, 0, 0)),
 (('join', 'VB'), (',', 'will', 'join', 'the', 'board', False, False, 0, 0)),
 (('the', 'DT'), ('will', 'join', 'the', 'board', 'as', False, False, 0, 0))]
```



```
-----------------------------------------------------------------------------------
  Word,    Tag            Features
-----------------------------------------------------------------------------------
('Pierre', 'NNP')        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
('Vinken', 'NNP')        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
(',', ',')               [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('61', 'CD')             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
('years', 'NNS')         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('old', 'JJ')            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
(',', ',')               [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('will', 'MD')           [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('join', 'VB')           [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('the', 'DT')            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('board', 'NN')          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('as', 'IN')             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('a', 'DT')              [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('nonexecutive', 'JJ')   [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('director', 'NN')       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
('Nov.', 'NNP')          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
('29', 'CD')             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
('.', '.')               [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
('Mr.', 'NNP')           [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
```

## Application to Penn Treebank Corpus

- Corpus : **Penn Treebank Corpus**
- Number of sentences : **3914**

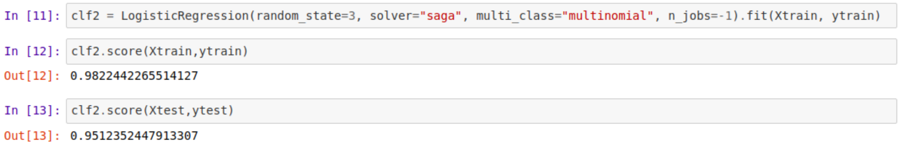


```
Entrée [3]:  y.shape
   Out[3]:  (25184,)

Entrée [4]:  X.shape
   Out[4]:  (25184, 8175)

Entrée [ ]:  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

- **Results**



```
In [11]: clf2 = LogisticRegression(random_state=3, solver="saga", multi_class="multinomial", n_jobs=-1).fit(Xtrain, ytrain)

In [12]: clf2.score(Xtrain,ytrain)
Out[12]: 0.9822442265514127

In [13]: clf2.score(Xtest,ytest)
Out[13]: 0.9512352447913307
```

## A simple illustrating example for translation

Cet exemple est issu de Adam L. Berger et al. (1996), où ils s'intéressent à la traduction du mot "in" par les cinq possibilités suivantes {dans, en, à, au cours de, pendant}. Sans aucune information externe :

$$p(\text{ dans }) + p(en) + p(à) + p(\text{ au cours de }) + p(\text{ pendant }) = 1.$$

Notons $p_i = p\left(w_i\right)$ for $w_i \in \{dans, en, à, au\ cours\ de, pendant\}$. Ainsi

$$p = \underset{p \in \mathcal{P}_1}{\arg\max} \{H(p)\}$$

$$\text{où } \mathcal{P}_1 = \left\{p = (p_1, \ldots, p_5) \,\middle|\, \sum p_i = 1\right\} \text{ et } H(p) = -\sum p_i \log p_i.$$

$$\frac{\partial H(p)}{\partial p_i} = 0 \Leftrightarrow \frac{\partial}{\partial p_i}\left(-\sum p_i \log p_i + \lambda\left(\sum p_i - 1\right)\right) = 0$$

$$\Leftrightarrow p_1 = \ldots = p_5 = e^{\lambda - 1} \Rightarrow \forall i,\ p_i = \frac{1}{5} \text{ and } \lambda = 1 - \log 5$$

## A simple illustrating example for translation

Supposons maintenant que les mots "dans" et "en" apparaissaient 30% de fois. Nous voulons que le modèle prenne en compte la contrainte supplémentaire suivante :

$$\begin{aligned} p(\text{ dans }) + p(\text{ en }) &= 3/10 \\ p(\text{ dans }) + p(en) + p(à) + p(\text{ au cours de }) + p(\text{ pendant }) &= 1 \end{aligned}$$

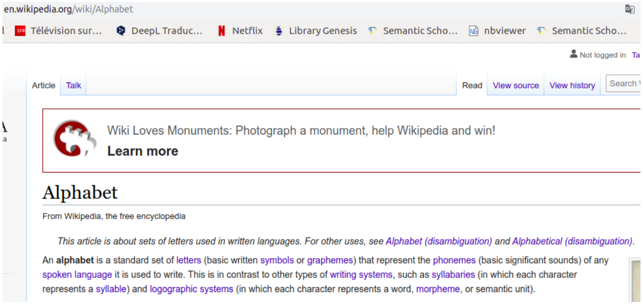
Le programme devient : $p = \arg\max_{p \in \mathcal{P}_2} \{H(p)\}$

$$\text{où} \quad \mathcal{P}_2 = \left\{ p = (p_1, \ldots, p_5) \text{ under } \sum p_i = 1 \text{ and } p_1 + p_2 = \frac{3}{10} \right\}$$

Le même calcul que ci-dessus conduit à

$$p_1 = p_2 = e^{\lambda_1 + \lambda_2 - 1} \text{ and } p_3 = p_4 = p_5 = e^{\lambda_1 - 1}$$

$\Rightarrow p(dans) = p(en) = 3/20 \; ; \; p(à) = p(au\ cours\ de) = p(pendant) = 7/30$



en.wikipedia.org/wiki/Alphabet

Télévision sur... DeepL Traduc... Netflix Library Genesis Semantic Scho... nbviewer Semantic Scho...

Not logged in

Article Talk Read View source View history Search

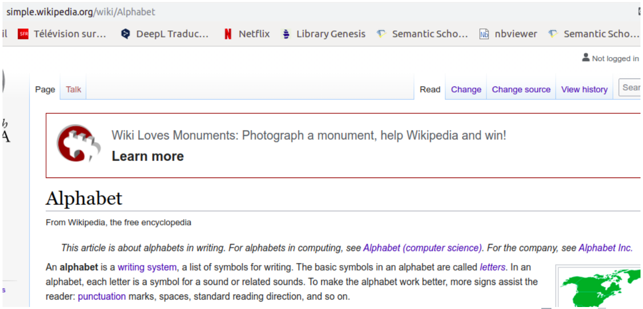
Wiki Loves Monuments: Photograph a monument, help Wikipedia and win!
**Learn more**

Alphabet

From Wikipedia, the free encyclopedia

*This article is about sets of letters used in written languages. For other uses, see Alphabet (disambiguation) and Alphabetical (disambiguation).*

An **alphabet** is a standard set of letters (basic written symbols or graphemes) that represent the phonemes (basic significant sounds) of any spoken language it is used to write. This is in contrast to other types of writing systems, such as syllabaries (in which each character represents a syllable) and logographic systems (in which each character represents a word, morpheme, or semantic unit).



simple.wikipedia.org/wiki/Alphabet

Télévision sur... DeepL Traduc... Netflix Library Genesis Semantic Scho... nbviewer Semantic Scho...

Not logged in

Page Talk Read Change Change source View history Sear

Wiki Loves Monuments: Photograph a monument, help Wikipedia and win!
**Learn more**

Alphabet

From Wikipedia, the free encyclopedia

*This article is about alphabets in writing. For alphabets in computing, see Alphabet (computer science). For the company, see Alphabet Inc.*

An **alphabet** is a writing system, a list of symbols for writing. The basic symbols in an alphabet are called *letters*. In an alphabet, each letter is a symbol for a sound or related sounds. To make the alphabet work better, more signs assist the reader: punctuation marks, spaces, standard reading direction, and so on.

## Corpus

- In our case, we use Wikipedia, the simple version and the standard version in English.
- A file with all titles, for each of 5000 random titles the two versions of the article are extracted (simple vs. complex). And we keep only the pairs that do not have an empty component, with texts that have more or less the same sizes.

## Representations

Deux manières

- High dimensional sparse vectors (discrete representation)

**MaxEnt and Feature based models**

$$\begin{cases} x = (0,1,0,....,1,0) \in \mathbb{R}^q \\ q \text{ very large, } O(q) \approx \text{hundreds of thousands or millions.} \end{cases}$$

- Low dimensional dense vectors (continuous representation)

**Word2vec**

$$\begin{cases} x = (0.54,\ -0.312,\ 3.1,\ ....,\ -2.344,\ 0.543) \in \mathbb{R}^q \\ q \text{ small, } O(q) \approx \text{hundreds or thousands.} \end{cases}$$

## Representations

- The interest of the first method: easy to interpret, the disadvantage heavy to implement and slows down the programs
- For the second method, it is difficult to interprete the role of each component, on the other hand vectors are able to capture semantics

## Complexity Measure

- Classification: Simple Text vs Complex Text (Wiki)
- Feature extraction process :
  $n$ observation $x_i \in \mathbb{R}^q$ with $q \approx 80000$ and $n \approx 12000$ ; where $n$ is the number of sentences and $q$ the dimension of these sentences vectors obtained after sent_tokenizing about 3000 texts
- L1-L2 Penalised Logistic Regression
- Obtained results : **Precision of 85%**

## Thank you!