

# **Institut Supérieur des Etudes Technologiques de Bizerte**

## **Département GE**

### **Mastère RAIA**

# Traitement Automatique des Langues (TAL)

# Dr. Ing. Hela mahersia

Email: helamahersia@yahoo.fr



## Partie 1

# Introduction

# Plan

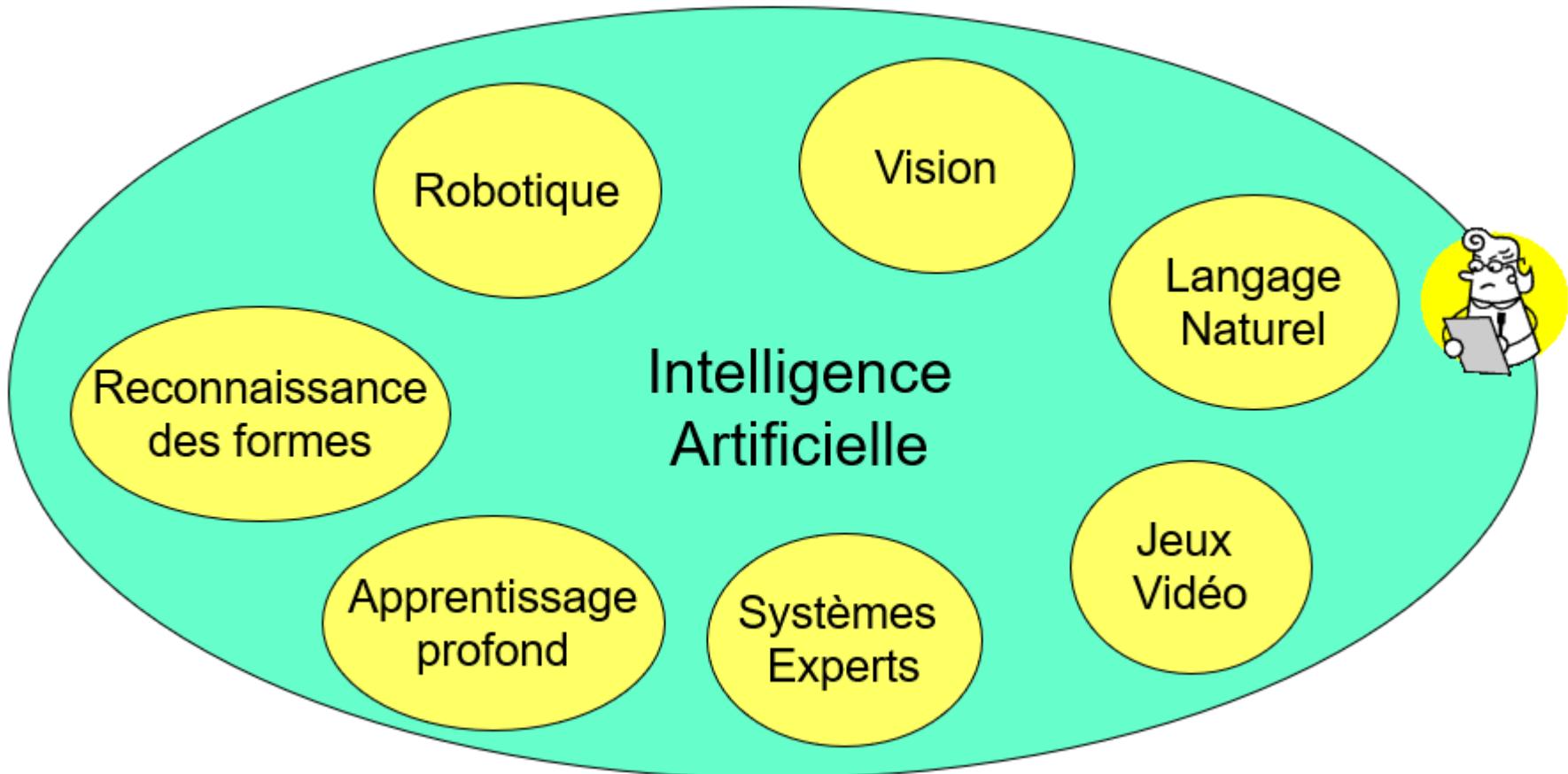




# Motivation TAL ?



# L'intelligence artificielle



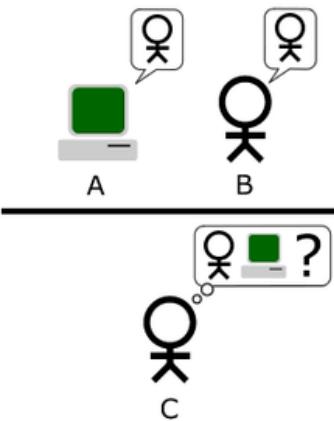
# TAL : Objectif



L'objectif du TAL est la conception de logiciels capables de traiter de façon automatique des données exprimées dans une **langue «naturelle»**

✿ Le traitement automatique des langues (TAL) étudie la composante langagière de l'intelligence artificielle

✿ Composante primordiale  
dans le **test de Turing**



# TAL : Définition

## Définition (Traitement Automatique du Langage Naturel (TALN))

*On regroupe sous le vocable de traitement automatique du langage naturel (TALN) l'ensemble des recherches et développements visant à modéliser et reproduire, à l'aide de machines, la capacité humaine à produire et à comprendre des énoncés linguistiques dans des buts de communication*



Discipline à cheval entre la linguistique,  
l'informatique et les mathématiques

# Exemple : Question-réponse avec Watson



# Exemple : Question-réponse avec Watson





# Applications du TAL

# Reconnaissance de caractères (OCR)



## ✿ Principe général :

- ❖ Numérisation de documents écrits (scanner) en images
- ❖ Application de techniques de reconnaissance de formes (lettres) à l'aide d'apprentissage (réseaux de neurones, HMM)
- ❖ Exploitation d'un modèle de langage (dont des ressources : dictionnaires, grammaires, etc.) pour déterminer l'hypothèse la plus probable

## ✿ Applications pratiques : dématérialisation de documents (bibliothèques), formulaires (chèques, administration), adresses pour le tri postal, identification d'immatriculation



# Correction orthographique / grammaticale



## ✿ Principe général :

- ❖ Identifier les mots (tokenization)
- ❖ Correction orthographique : mots qui n'appartiennent pas au dictionnaire et qui ne sont pas en langue étrangère, ni des noms propres, ni des chiffres, ni des sigles...
- ❖ Correction grammaticale : déterminer la fonction des mots au sein de la phrase (déterminant, nom, verbe, adverbe, etc.) puis réaliser une analyse syntaxique à l'aide de grammaires

## ✿ Applications pratiques : correction de document rédiger par des étudiants (exemple de fautes !)



# Traduction automatique

## ✿ Principe général :

- ◊ Sélection des langues source et cible
- ◊ Utilisation d'un modèle de langage :
  - pour la source et pour la cible
  - Recherche des traductions possibles et probables



## ✿ Applications pratiques : traduction de documents, dictionnaires bilingues, recherche d'informations multilingue

# Extraction et recherche d'informations



## ✿ Principe général :

- ◊ Enregistrer des documents (ou leurs adresses) et déterminer un ensemble de caractéristiques selon leur analyse
- ◊ Construire des indices accessibles et régulièrement mis à jour
- ◊ Répondre à la demande aux requêtes par sélection des documents les plus pertinents

## ✿ Applications pratiques : recherche en ligne, veille, surveillance, résumé automatique, classification de documents



# Reconnaissance de la parole



## ✿ Principe général :

- ❖ Traitement acoustique du flux audio
- ❖ Analyse du signal (transformée de Fourier)
- ❖ Reconnaissance par modèles (apris : HMM ou réseaux de neurones), avec implémentation de modèle de langage qui donne la séquence la plus probable

## ✿ Applications pratiques : dictaphones (smartphones), serveurs vocaux (hotline), transcriptions automatiques (sous-titres, notamment pour les malentendants), synthèse vocale, ...



# Chatbots

✿ Chatbots = agents conversationnels





# Défis du TAL

# Difficultés à surmonter



- ✿ Interpréter une phrase/un document correctement est une tâche très complexe à automatiser
- ✿ Pour un être humain, c'est une tâche qui ne requiert aucun effort
- ✿ Mais même pour nous, certains cas peuvent être une casse-tête
- ✿ Quelques sources de difficultés à surmonter:
  - ◆ Ambiguïté
  - ◆ Métaphores
  - ◆ Variations dans le temps

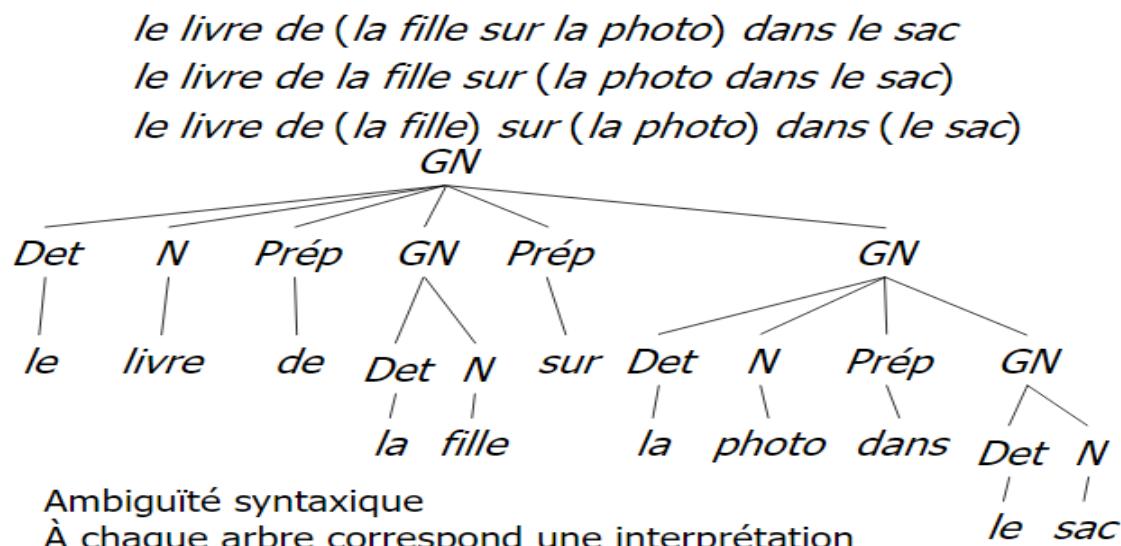


# Ambiguïté

- ✿ Les langues naturelles étant ambiguës, pour une même phrase plusieurs analyses sont possibles  
→ plusieurs interprétations syntaxiques

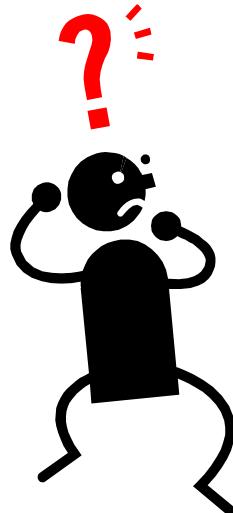
La **syntaxe** est l'étude de la façon avec laquelle les mots sont composés pour construire des phrases

- ✿ Exemple :



# Métaphores

- ✿ Certains mots sont utilisés d'une façon métaphorique
- ✿ Exemple:



... supermassive black hole eating star - CNN

→ Cette utilisation de mots rend une approche à base de règles prédéfinies très difficile

# Variations dans le temps



## ✿ L'utilisation des mots peut varier dans le temps

### ✿ Exemple : sort of

- ◊ Utilisation originale: what sort of animal did you see
- ◊ Utilisation plus moderne: he sort of understood what was going on

### ✿ Apparition de nouveaux mots: internet, wifi...



# Pour résoudre ces problèmes...



## ✿ NLP : tâche difficile, Quels pré-requis?

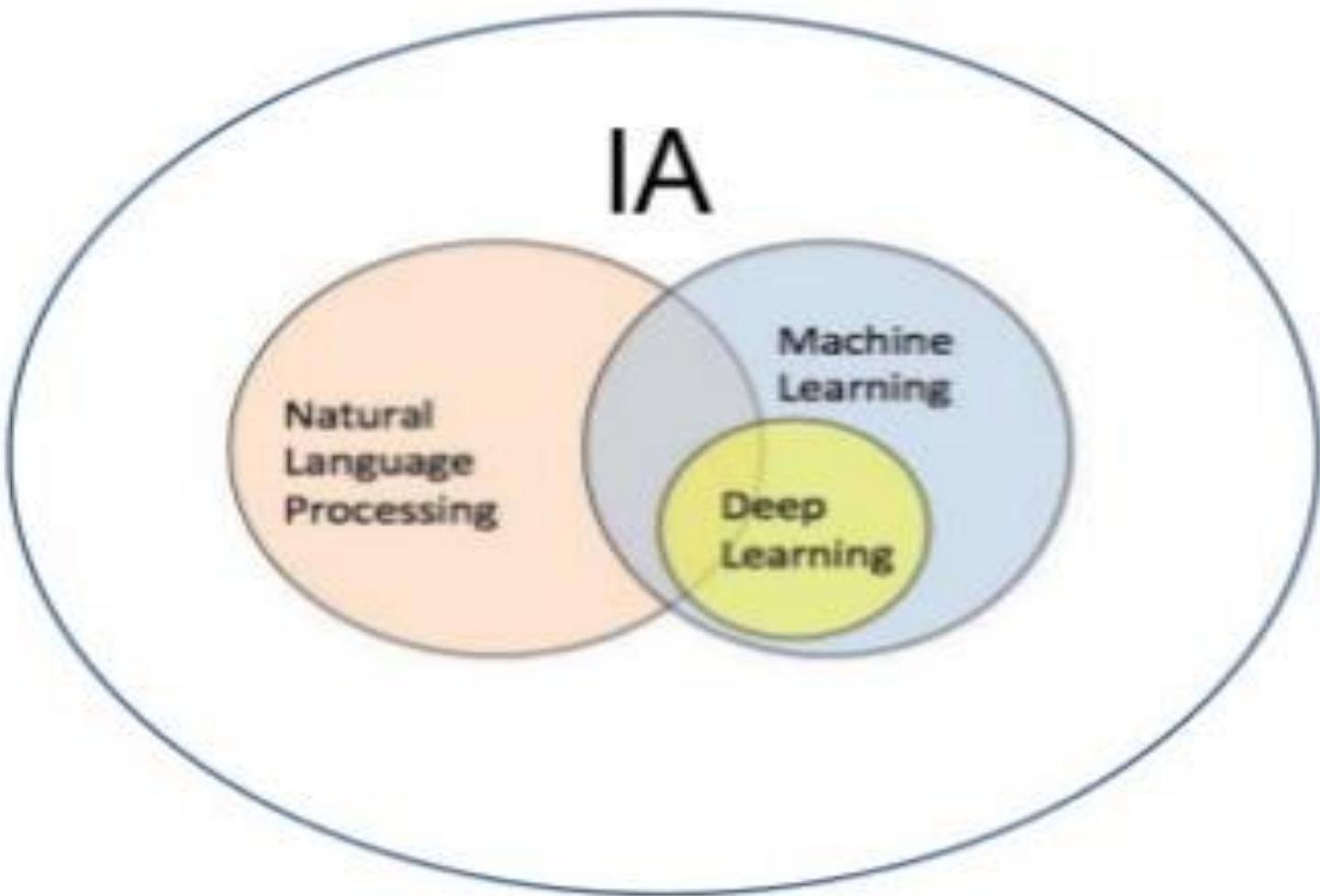
- ◊ Connaissance du langage
- ◊ Connaissance du mot

## ✿ Comment on reconnaît ?

- ◊ Modèles probabiliste construits à partir des données du langage
  - $P(\text{« house »} \rightarrow \text{« maison »})$  élevée
  - $P(\text{« avocat général »} \rightarrow \text{« the general avocado »})$  faible



# L'avenir...





## Partie 2

# Connaissances linguistiques en TAL

# Plan



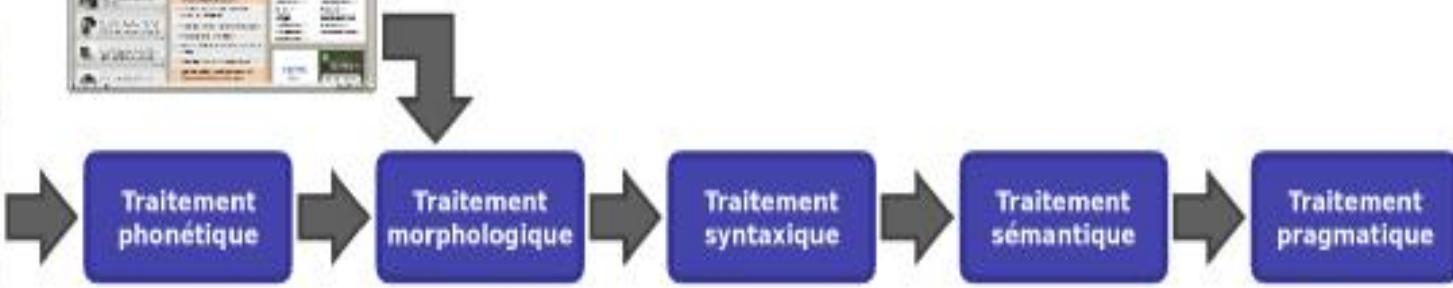
# Niveaux de traitement d'une application TAL (1/2)

✿ Pour un système TAL, une véritable analyse linguistique nécessite la capacité de :

- ❖ reconnaître
- ❖ structurer
- ❖ comprendre
- ❖ contextualiser

# Niveaux de traitement d'une application TAL (2/2)

- phonologie: les sons
- morphologie: les mots et leur forme
- syntaxe: l'organisation des mots en phrase
- sémantique: le sens hors contexte
- pragmatique: le sens en contexte



# Problème de l'ambiguïté



- ✿ Comment déterminer le genre du mot livre dans les phrases suivantes :
  - ❖ J'ai lu un livre
  - ❖ Il ne s'agit pas de livres mais de lires
    - par un traitement morphologique ?
- ✿ Pour la première phrase, il faut repérer que livre est précédé de l'article un
  - traitement syntaxique !
- ✿ Pour la seconde, il faut intégrer des connaissances sur le monde et la situation de communication (*livre* et *lire* sont deux monnaies)
  - traitement pragmatique !



# Traitement phonétique

✿ À partir d'une entrée vocale il faut essentiellement extraire deux informations linguistiques :

- ◊ Les phonèmes - sons successifs qui constituent les mots
  - (ex : chapeau comprend quatre phonèmes ch / a / p / eau)
- ◊ La prosodie - intonation, rythme et intensité permettant, par exemple, de distinguer une question ou une réponse

✿ Les phonèmes doivent être regroupés pour constituer des mots

1

# Morphologie TAL ?

Chapitre 2

30



# Traitement morphologique



## Problèmes à résoudre :

✿ qu'est-ce qu'un mot ? Comment segmenter un texte ?  
Quels types de séparateurs faut-il prendre ?

- ◊ pomme de terre, au fur et à mesure, tel que
- ◊ aujourd'hui, rock'n roll
- ◊ O.N.U, 32.5c
- ◊ porte-monnaie, voulez-vous
- ◊ c'est-à-dire

✿ qu'est-ce qu'une phrase ?

- ◊ La balance des U.S.A est de 3.4 M. de dollars.



# Traitement morphologique



- ✿ Morphologie : étude de la formation des mots à partir d'unités plus petites appelées morphèmes
  - ✿ Par exemple, le mot lapins est composé de deux morphèmes :
    - ❖ la base ou racine (lapin)
    - ❖ un suffixe, la désinence du pluriel (s)
    - ❖ *chanteurs* : 3 morphèmes : *chant-* « chant », *-eur-* « celui qui fait » et *-s*
- Morphème : sous-chaîne portant une signification ( $\neq$  lettres)
- 
- ✿ Morphème radical ou racine ou lexèmes (stem en anglais): morphème principal
  - ✿ Morphèmes grammatical : ou affixe (préfixe, suffixe, infixé, circonfix) n'apparait jamais isolé mais se combine aux lexèmes



# Traitement morphologique

→ Un mot doit avoir un seul radical mais plusieurs affixes

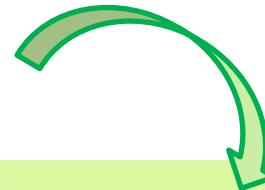
\* Exemple : chanteurs

chant (chant) → radical

eur (celui qui fait)

S (pluriel)

→ affixes



- **préfixe** : se place avant le radical
- **suffixe** : se place près le radical
- **circonfixe** : peut se placer avant ou après le radical
- **infixe** : se place à l'intérieur de le radical

# Phénomènes morphologiques



## ✿ Il existe 3 types de combinaisons des morphèmes :

◊ la **flexion** : modifier un radical afin de modifier sa forme linguistique (genre, nombre, temps, ...)

- Flexion cas des noms : déclinaison

|          | Regular Nouns |          | Irregular Nouns |      |
|----------|---------------|----------|-----------------|------|
| Singular | cat           | thrush   | mouse           | ox   |
| Plural   | cats          | thrushes | mice            | oxen |

- Flexion cas des verbes : conjugaison
  - Flexion productive : classe des verbes réguliers car facile à inclure de nouveaux mots
  - Pas de règles générales pour les verbes irreguliers



# Phénomènes morphologiques

- ✿ Il existe 3 types de combinaisons des morphèmes :
  - ◊ la **dérivation** : permet de créer de nouveaux mots, souvent d'une classe différentes
    - Exemple : happy → happiness, unhappy
    - Exemples :
      - Nominalization : creation d'un nom à partir d'un verbe ou d'un adjectif : suffix(ation) + radical (computerize) → computerization
      - creation d'un adjectif à partir d'un verbe ou d'un nom: suffix(al) + radical (computation) → computational

# Phénomènes morphologiques



✿ Il existe 3 types de combinaisons des morphèmes :

◊ La formation de mots composés correspond à former un mot par concaténation de plusieurs radicaux

- Exemple : doghouse : en un seul mot

◊ Radicaux juxtaposés dont le sens a une signification différente des radicaux qui le composent :

- Radicaux séparés : « pomme de terre », « cordon bleu »,...
- Radicaux séparés par un symbole de ponctuation : « abat-jour », « aujourd'hui », ...
- Radicaux unifiés (collés) : « gentilhomme », « monsieur », « lorsque », « toutefois », « vinaigre », « autobus »



# Tokenization : Segmentation

- ✿ Segmenter un texte en « unités minimales » pour le traiter :
- ✿ Un premier prétraitement souvent utilisé consiste à couper chaque document en une liste d'occurrences : tokens
  - ◊ Cette étape est appelée segmentation (tokenization)
- ✿ Séparer les mots en fonction des espaces ne suffit pas
- ✿ Exemple : Jean mange des pommes de terre.

U1 Jean  
U2 mange  
U3 des  
U4 pommes de terre  
U5 .

# Tokenization

## Exemple 2:

Les étudiants, n'ont-ils pas tous 15,3 de moyenne ?

Tokenization

Les | étudiants | , | n' | ont | -ils | pas | tous | 15,3 | de | moyenne | ?

# TAL et morphologie?

✿ Dans un système de TAL, l'analyse morphologique a pour objectif de :

- ◊ Reconnaître les morphèmes
- ◊ Proposer une lemmatisation (forme canonique ou lemme) associée à un mot :
  - pour un verbe : ce verbe à l'infinitif,
  - pour les autres mots : le mot au masculin singulier.
  - Exemple : L'adjectif **petit** existe sous quatre formes : petit, petite, petits et petites → la forme canonique de tous ces mots est petit
- ◊ Reconnaître les entités nommées (noms de personnes, d'organisations, d'entreprises, de lieux, quantités, distances, valeurs, dates...)



# Syntaxe et TAL ?

Chapitre 2

# Traitement syntaxique



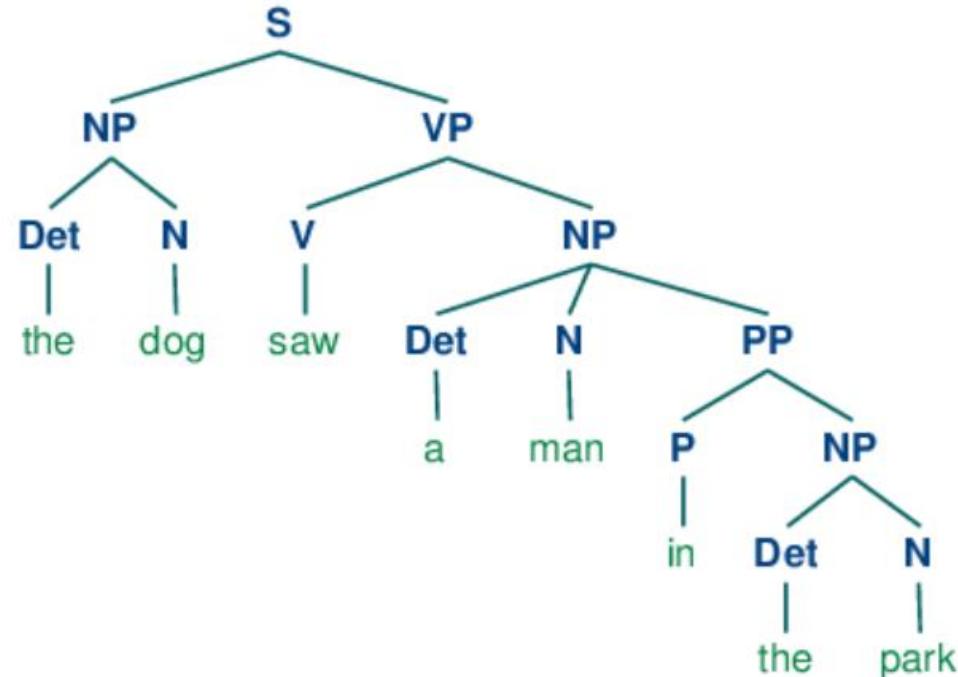
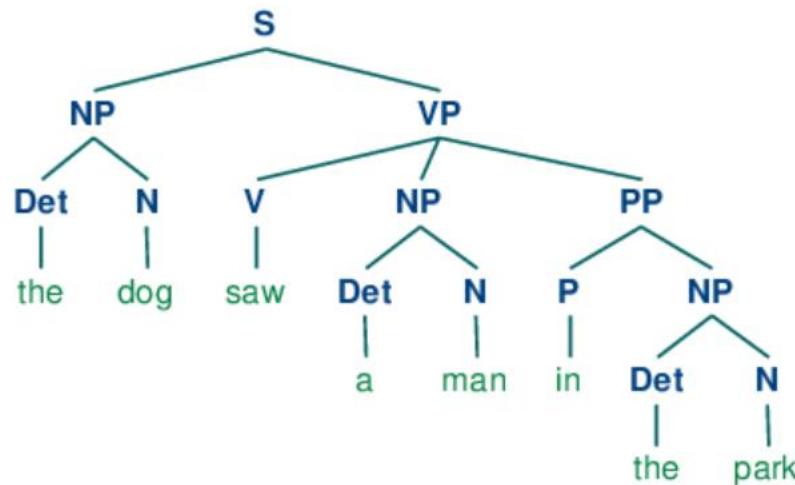
- ✿ **Syntagme : mot ou une suite de mots consécutifs auxquels on peut associer une catégorie syntaxique**
  
- ✿ **Dans un système de TAL, l'analyse syntaxique peut avoir pour objectif de :**
  - ◊ Découper la phrase en syntagmes (chunks)
  - ◊ Reconnaître les termes
  - ◊ Repérer des dépendances syntaxiques
  - ◊ **Proposer une organisation hiérarchique des syntagmes**



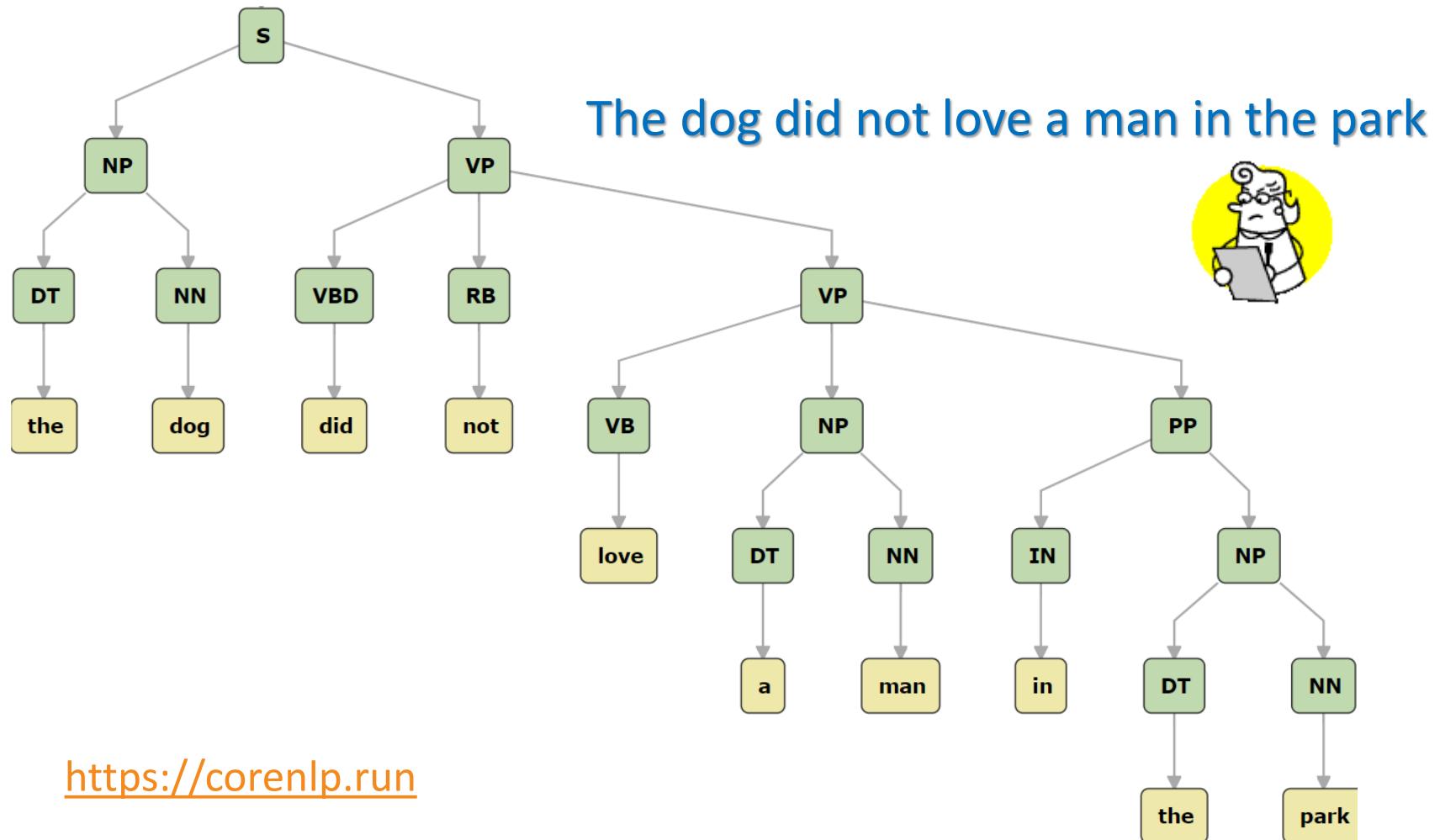
# Arbre syntaxique : ambiguïté

The dog saw a man in the park

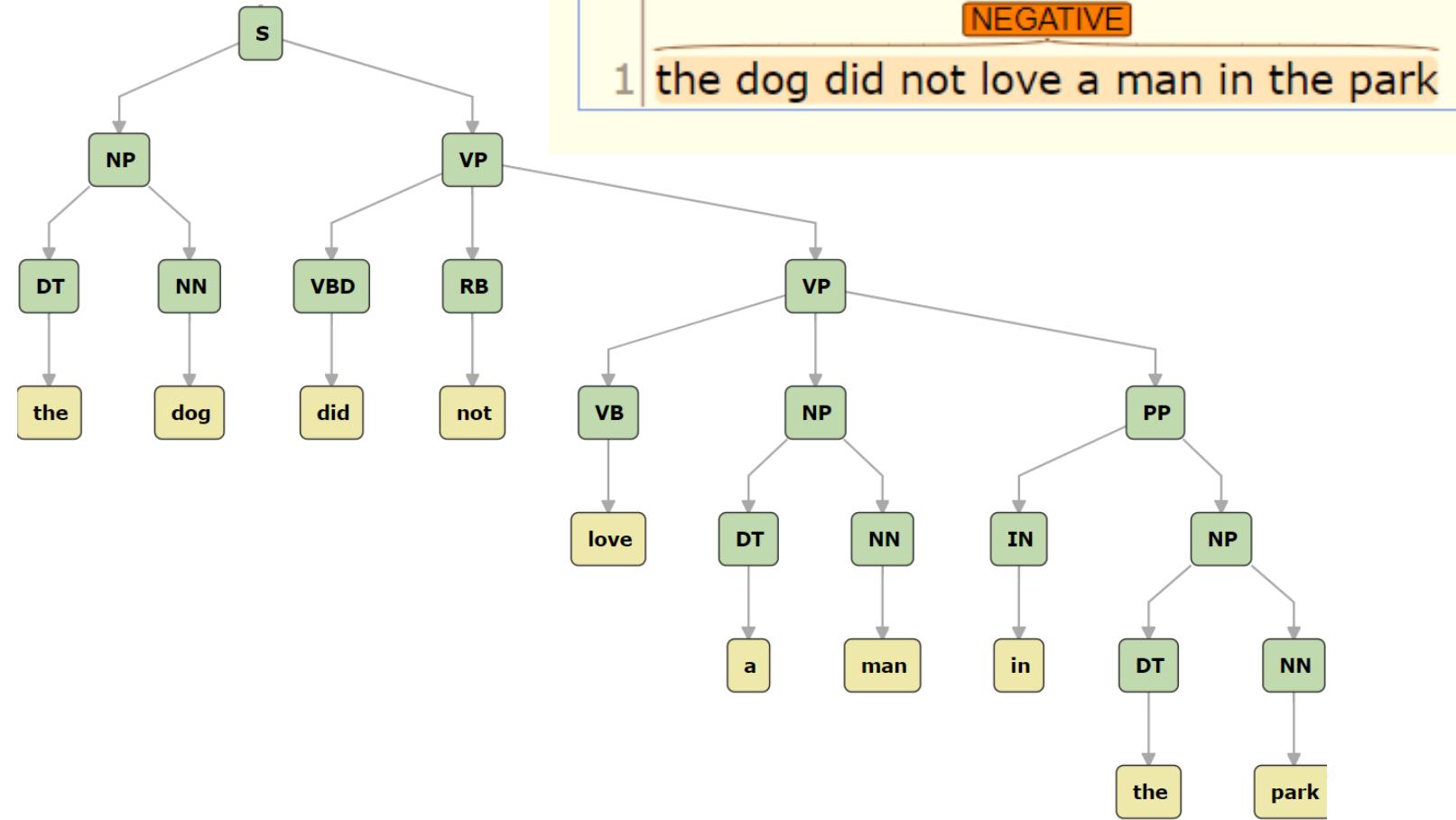
The dog hate a man in the park?



# Arbre syntaxique : Exemple



# Arbre syntaxique : utilité

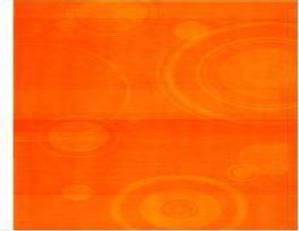




# La Sémantique et TAL?

Chapitre 2

# La sémantique



- ✿ La sémantique vise à étudier le sens hors contexte.
- ✿ Le traitement sémantique prend comme unité d'analyse la phrase, et conduit à représenter sa partie significative.
- ✿ L'analyseur sémantique décrit le sens des mots de la phrase, ces mots:
  - ❖ sont identifiés par l'analyse morphologique,
  - ❖ et regroupés en structures par l'analyse syntaxique.



# La pragmatique

- ✿ La pragmatique vise à l'étude du sens en contexte.
- ✿ Trouver la signification réelle des phrases liées aux conditions situationnelles et contextuelles.
- ✿ La sémantique est liée à la pragmatique
  - ❖ la sémantique traite la signification du langage en isolation
  - ❖ la pragmatique traite la signification du langage en usage

# Comment faire du TAL ?



- ✿ Approche classique : Rule-based approach
  - ◊ Regular expression
  - ◊ Context-free grammars
- ✿ L'approche à base de corpus : Probabilistic modeling & Machine Learning
  - ◊ Modèles automatiquement extraits de volumes importants de données textuelles caractéristiques de la classe d'application envisagée
- ✿ Approche par apprentissage profond : Deep learning approach





## Partie 3

# Prétraitements de textes (Dans la pratique)

# Un texte ?

- ✿ Nous pouvons considérer le texte comme étant une séquence de :
- ✿ Caractères
- ✿ Mots
- ✿ Groupements nominaux et verbaux
- ✿ Phrases
- ✿ Paragraphes

# Un mot ?

✿ Il semble naturel de penser à un texte comme une séquence de mots

- ◊ Un mot est une séquence significative de caractères

✿ Comment trouver les limites des mots?

- ◊ En anglais, nous pouvons diviser une phrase par des espaces ou une ponctuation

**Input:** Friends, Romans, Countrymen, lend me your ears;

**Output:** Friends      Romans      Countrymen      lend      me      your      ears

◊ En allemand, il y a des mots composés écrits sans espaces :

- «Rechtsschutzversicherungsgesellschaften» signifie sociétés d'assurance qui assurent la protection juridique !!

◊ En japonais, il n'y a pas d'espaces du tout !!

# Tokenization

## ✿ Tokenization est un processus qui divise une séquence d'entrée en des tokens

- ❖ Vous pouvez considérer un token comme une unité utile pour le traitement sémantique :
- ❖ Peut être un mot, une phrase, un paragraphe, etc.

## ✿ Tokenization par l'espace

- ❖ `nltk.tokenize.WhitespaceTokenizer`

This    is    Andrew's    text,    isn't    it?

- ❖ Problème : « it » et « it? » sont des tokens différents avec le même sens

# Tokenization

## ✿ Tokenization par la ponctuation :

- ◊ nltk.tokenize.WordPunctTokenizer

This is Andrew ' s text , isn ' t it ?

- ◊ Problème: « s », « isn », « t » ne sont pas très significatifs

## ✿ Tokenization par ensemble de règles :

- ◊ nltk.tokenize.TreebankWordTokenizer

This is Andrew 's text , is n't it ?

- ◊ « s » et « 'nt » sont plus significatifs pour le traitement

# Exemples en python

```
import nltk  
text = "This is Andrew's text, isn't it?"
```

```
tokenizer = nltk.tokenize.WhitespaceTokenizer()  
tokenizer.tokenize(text)
```

```
['This', 'is', "Andrew's", 'text,', "isn't", 'it?']
```

```
tokenizer = nltk.tokenize.TreebankWordTokenizer()  
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'s", 'text', ',', 'is', "n't",  
'it', '?']
```

```
tokenizer = nltk.tokenize.WordPunctTokenizer()  
tokenizer.tokenize(text)
```

```
['This', 'is', 'Andrew', "'", 's', 'text', ',', 'isn',  
"', 't', 'it', '?']
```

# Normalisation des tokens (1/2)

- ✿ Dans certaines applications, les variations morphologiques d'un même mot ne sont pas utiles
  - ❖ En recherche d'information : si je cherche exemple, il faut que le mot exemples soit aussi traité
- ✿ Il est souhaitable donc de normaliser la forme prise par les mots afin d'éliminer ces variations inutiles :
  - ① Racinisation ou stemming : retirer les affixes d'un mot pour obtenir la forme la plus proche du radical/racine
    - ❖ généralement seuls les suffixes sont enlevés
    - ❖ Le token obtenu n'est pas forcément un mot

# Exemple de stemmers

## ✿ Le stemmer de Porter

- ◊ 5 phases heuristiques de réductions de mots, appliquées séquentiellement
- ◊ Exemple de règles de la phase 1 :

| Rule      | Example           |
|-----------|-------------------|
| SSES → SS | caresses → caress |
| IES → I   | ponies → poni     |
| SS → SS   | caress → caress   |
| S →       | cats → cat        |

## ✿ En python : nltk.stem.PorterStemmer

- ◊ Examples:
  - feet → feet                      cats → cat
  - wolves → wolv                 talked → talk

## ✿ Problème: échoue sur les formes irrégulières, produit des non-mots

# Normalisation des tokens (2/2)

② Lemmatisation ou lemmatization : renvoie la forme de base ou du dictionnaire d'un mot = lemme

- ❖ Contrairement à la racinisation, le token trouvé doit être un mot
- ❖ C'est une tâche plus difficile

# Exemple de Lemmatizer

- ✿ WordNet Lemmatizer : Utilise la base de données WordNet pour rechercher les lemmas
- ✿ En python : nltk.stem.WordNetLemmatizer
  - ◊ Exemples :
    - pieds → pied              chats → chat
    - loups → loup              parlé → parlé
- ✿ Problèmes : toutes les formes ne sont pas réduites

Stemmer ou lemmatizer ? nous devons essayer les deux pour choisir la meilleure solution qui convient à notre tâche

# Exemples en python

```
import nltk  
text = "feet cats wolves talked"  
tokenizer = nltk.tokenize.TreebankWordTokenizer()  
tokens = tokenizer.tokenize(text)
```

```
stemmer = nltk.stem.PorterStemmer()  
" ".join(stemmer.stem(token) for token in tokens)
```

u'feet cat wolv talk'

```
stemmer = nltk.stem.WordNetLemmatizer()  
" ".join(stemmer.lemmatize(token) for token in tokens)
```

u'foot cat wolf talked'

# D'autres normalisations



## ✿ Normalisation des majuscules

- ◊ Us, us → us (si les deux sont pronoms)
- ◊ us, US (pourrait être pronom ou pays)
- ◊ Nous pouvons utiliser des heuristiques :
  - Mettre au minuscule le début de la phrase
  - Mettre au minuscule les mots dans les titres
  - Laisser les mots à mi-phrase tels qu'ils sont
- ◊ Ou nous pouvons utiliser l'apprentissage automatique pour trancher → mais c'est un peu dur

## ✿ Acronymes

- ◊ eta, e.t.a., E.T.A. → E.T.A.
- ◊ Nous pouvons utiliser les expressions régulières





**MERCI POUR VOTRE  
ATTENTION**

**DES QUESTIONS ? —**



## Partie 4

# Approche statistique en TAL

# Rappel : TAL

Les tâches TAL sont partout autour de nous:

- suggestion dans la recherche,
- réponses automatiques Gmail,
- traduction automatique.

Busy as usual, but fine :)

How can I help you?

I am looking for a cheap  
trip for the holidays...

Do you want to relax  
on the beach?

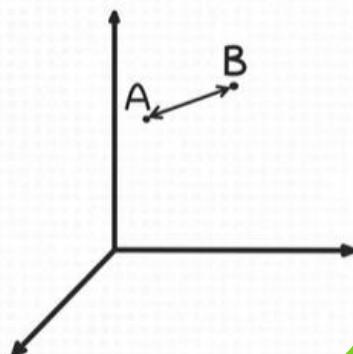
Oh no please

What kind of vacation  
do you want?

A



B



# Contexte : approches principales du TAL



# Plan





# Propriétés des données textuelles ?



Chapitre 3

# Corpus

## ✿ Dans un texte, certaines structures sont typiques

### ✿ Corpus :

- ◊ ensemble de données textuelles
- ◊ Collection de textes ou de documents

### ✿ Un corpus n'est jamais aléatoire

- ◊ Corpus d'entraînement
- ◊ /corpus de test

| Word | Freq. | Use                                   |
|------|-------|---------------------------------------|
| the  | 3332  | determiner (article)                  |
| and  | 2972  | conjunction                           |
| a    | 1775  | determiner                            |
| to   | 1725  | preposition, verbal infinitive marker |
| of   | 1440  | preposition                           |
| was  | 1161  | auxiliary verb                        |
| it   | 1027  | (personal/expletive) pronoun          |
| in   | 906   | preposition                           |
| that | 877   | complementizer, demonstrative         |
| he   | 877   | (personal) pronoun                    |
| I    | 783   | (personal) pronoun                    |
| his  | 772   | (possessive) pronoun                  |
| you  | 686   | (personal) pronoun                    |
| Tom  | 679   | proper noun                           |
| with | 642   | preposition                           |

Statistique des mots du roman Tom Sawyer

# Corpus : Définitions



- ✿ Collection de données langagières qui sont sélectionnées et organisées selon des critères linguistiques explicites pour servir d'échantillons dans une langue. [Habert et al. 1998]
  - ◊ Pour être viable aujourd'hui, un corpus doit exister sous un format électronique
- ✿ Le corpus spécialisé : restreint à une situation de communication, un domaine, très constraint du point de vue lexical, que l'on trouve dans les domaines scientifiques et techniques.
- ✿ Le corpus de référence : conçu pour fournir une information en profondeur sur une langue. Il est suffisamment grand pour représenter toutes les variétés de cette langue [Sinclair, 1996]



# Les corpus en langue anglaise



## ✿ BNC : British National Corpus

- ◊ 100 millions de mots anglais
- ◊ 10 % d'oral, des textes de fiction depuis 1960, des textes informatifs depuis 1975

## ✿ Cobuild - Bank of English

- ◊ 200 millions de mots, oral et écrit
- ◊ à partir de 1980, à l'Université de Birmingham et Collins (dictionnaires)

## ✿ Santa Barbara

- ◊ oral de l'anglais américain
- ◊ 75 \$, CD-ROM

## ✿ Shakespeare 2000

- ◊ les pièces de Shakespeare en version XML

# Mots?

- ✿ Un mot quelconque dans un corpus est appelé une occurrence (ou une instance, ou très souvent un token)

Ça c' est pour moi , le plus beau et le plus triste paysage du monde . C' est le même paysage que celui de la page précédente , mais je l' ai dessiné une fois encore pour bien vous le montrer .

- ✿ 43 occurrences, en comptant les signes de ponctuation
- ✿ 34 types, en distinguant majuscules/minuscules
- ✿ 75 % de ces types ont une occurrence de 1

# Fréquences (1/2)



- ✿ Combien de mots = combien de word *tokens* ?  
réponse : 71,370 → corpus très réduit
  - ✿ Combien de mots ≠ = combien de types ?  
réponse : 8,018.
- 
- ✿ Taux (tokens/types) =  $8.9 \approx 9$   
→ les mots dans ce corpus sont utilisés en moyenne 9 fois chacun
  - ✿ La distribution des mots utilisés est-elle uniforme?



# Fréquences (2/2)

## ✿ Fréquences des mots?

- ◊ Peu de mots fréquents
- ◊ Beaucoup de mots rares

## ✿ Un vrai problème en TAL :

- pour plusieurs mots , on n'a pas beaucoup d'observations
- Il est difficile de caractériser des mots qui n'existent presque pas dans le corpus

| Word Frequency | Frequency of Frequency |
|----------------|------------------------|
| 1              | 3993                   |
| 2              | 1292                   |
| 3              | 664                    |
| 4              | 410                    |
| 5              | 243                    |
| 6              | 199                    |
| 7              | 172                    |
| 8              | 131                    |
| 9              | 82                     |
| 10             | 91                     |
| 11-50          | 540                    |
| 51-100         | 99                     |
| > 100          | 102                    |

Statistique des mots du roman Tom Sawyer

# Classe fermées (1/2)

✿ Quels sont les mots les plus fréquents dans un corpus?

- ✿ Les mots fréquents forment des classes fermées
- ✿ Classes fermées = classes syntaxiques auxquelles on n'ajoute pas de mots
- ✿ Des classes dans lesquelles les mots ne sont jamais inventés
  - déterminants (un, le, son, etc.)
  - pronoms (je, tu, il, etc.)
  - conjonctions (mais, ou et, donc, etc.)
  - prépositions (pour, de, etc.)

# Classe fermées (2/2)

## Exemples de classes fermées en anglais

| Word | Freq. | Use                                   |
|------|-------|---------------------------------------|
| the  | 3332  | determiner (article)                  |
| and  | 2972  | conjunction                           |
| a    | 1775  | determiner                            |
| to   | 1725  | preposition, verbal infinitive marker |
| of   | 1440  | preposition                           |
| was  | 1161  | auxiliary verb                        |
| it   | 1027  | (personal/expletive) pronoun          |
| in   | 906   | preposition                           |
| that | 877   | complementizer, demonstrative         |
| he   | 877   | (personal) pronoun                    |
| I    | 783   | (personal) pronoun                    |
| his  | 772   | (possessive) pronoun                  |
| you  | 686   | (personal) pronoun                    |
| Tom  | 679   | proper noun                           |
| with | 642   | preposition                           |

# Loi de Zipf (1/4)

| Word  | Freq.<br>( <i>f</i> ) | Rank<br>( <i>r</i> ) | <i>f</i> · <i>r</i> | Word     | Freq.<br>( <i>f</i> ) | Rank<br>( <i>r</i> ) | <i>f</i> · <i>r</i> |
|-------|-----------------------|----------------------|---------------------|----------|-----------------------|----------------------|---------------------|
| the   | 3332                  | 1                    | 3332                | turned   | 51                    | 200                  | 10200               |
| and   | 2972                  | 2                    | 5944                | you'll   | 30                    | 300                  | 9000                |
| a     | 1775                  | 3                    | 5235                | name     | 21                    | 400                  | 8400                |
| he    | 877                   | 10                   | 8770                | comes    | 16                    | 500                  | 8000                |
| but   | 410                   | 20                   | 8400                | group    | 13                    | 600                  | 7800                |
| be    | 294                   | 30                   | 8820                | lead     | 11                    | 700                  | 7700                |
| there | 222                   | 40                   | 8880                | friends  | 10                    | 800                  | 8000                |
| one   | 172                   | 50                   | 8600                | begin    | 9                     | 900                  | 8100                |
| about | 158                   | 60                   | 9480                | family   | 8                     | 1000                 | 8000                |
| more  | 138                   | 70                   | 9660                | brushed  | 4                     | 2000                 | 8000                |
| never | 124                   | 80                   | 9920                | sins     | 2                     | 3000                 | 6000                |
| Oh    | 116                   | 90                   | 10440               | Could    | 2                     | 4000                 | 8000                |
| two   | 104                   | 100                  | 10400               | Applause | 1                     | 8000                 | 8000                |

Statistique des mots du roman Tom Sawyer

# Loi de Zipf (2/4)

✿ Loi de Zipf : la fréquence  $f$  du mot est inversement proportionnelle à son rang  $r$  (**approximativement**)

$$f \propto \frac{1}{r} \quad \text{ou} \quad f \times r = k$$

- ◊ Du rang 1 au rang 2 :  $f$  diminue beaucoup
- ◊ Du rang 5000 au rang 5001 :  $f$  ne diminue presque pas
- ✿ Beaucoup de mots qui ont une fréquence égale mais très basse et très peu de mots ont une fréquence élevée

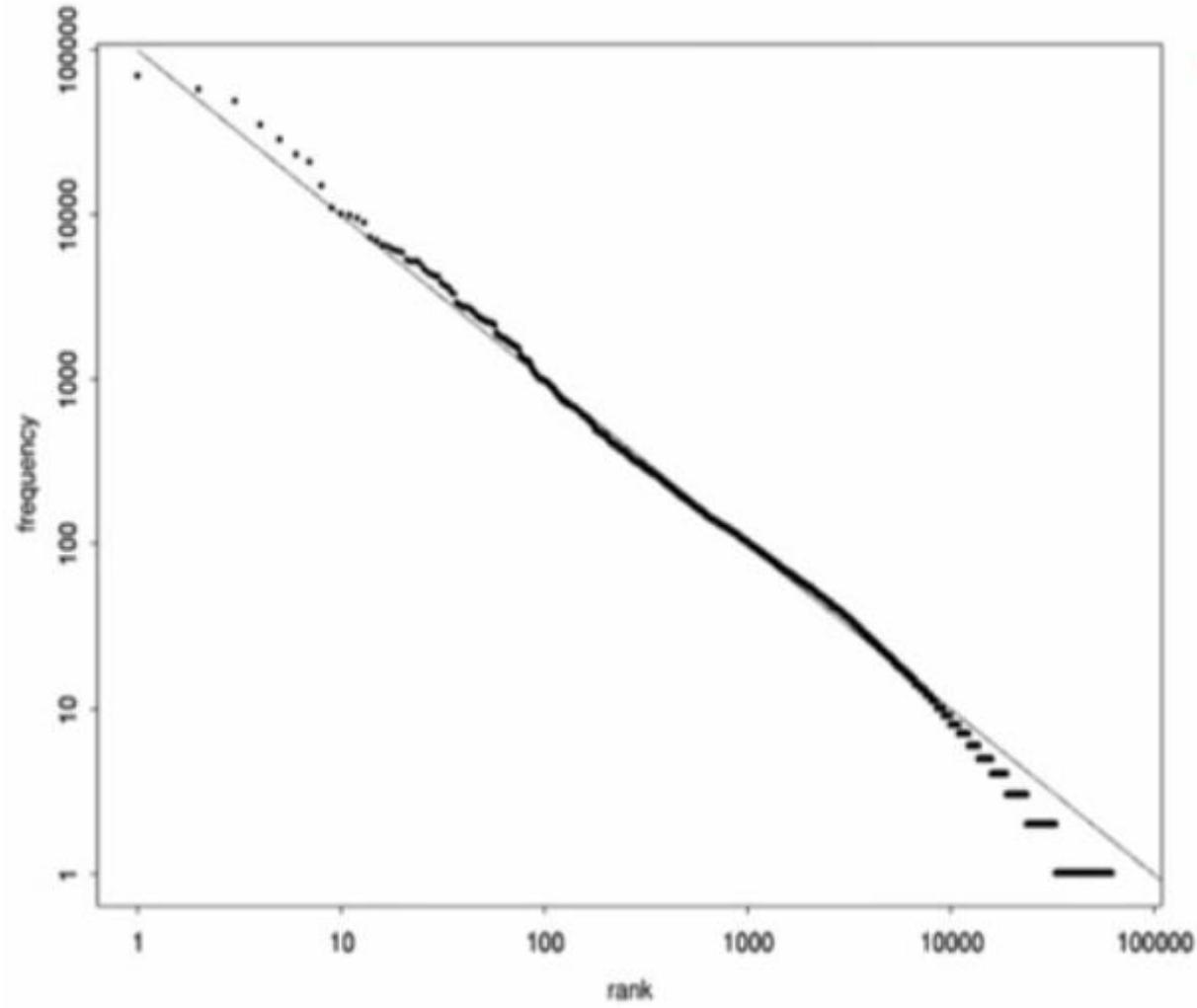
# Loi de Zipf (3/4)



| Word  | Freq.<br>( <i>f</i> ) | Rank<br>( <i>r</i> ) | $f \cdot r$ | Word     | Freq.<br>( <i>f</i> ) | Rank<br>( <i>r</i> ) | $f \cdot r$ |
|-------|-----------------------|----------------------|-------------|----------|-----------------------|----------------------|-------------|
| the   | 3332                  | 1                    | 3332        | turned   | 51                    | 200                  | 10200       |
| and   | 2972                  | 2                    | 5944        | you'll   | 30                    | 300                  | 9000        |
| a     | 1775                  | 3                    | 5235        | name     | 21                    | 400                  | 8400        |
| he    | 877                   | 10                   | 8770        | comes    | 16                    | 500                  | 8000        |
| but   | 410                   | 20                   | 8400        | group    | 13                    | 600                  | 7800        |
| be    | 294                   | 30                   | 8820        | lead     | 11                    | 700                  | 7700        |
| there | 222                   | 40                   | 8880        | friends  | 10                    | 800                  | 8000        |
| one   | 172                   | 50                   | 8600        | begin    | 9                     | 900                  | 8100        |
| about | 158                   | 60                   | 9480        | family   | 8                     | 1000                 | 8000        |
| more  | 138                   | 70                   | 9660        | brushed  | 4                     | 2000                 | 8000        |
| never | 124                   | 80                   | 9920        | sins     | 2                     | 3000                 | 6000        |
| Oh    | 116                   | 90                   | 10440       | Could    | 2                     | 4000                 | 8000        |
| two   | 104                   | 100                  | 10400       | Applause | 1                     | 8000                 | 8000        |

Statistique des mots du roman Tom Sawyer

# Loi de Zipf (4/4)



Montrer que les données textuelles de la courbe sont régis par la loi de Zipf

# Collocations (1/2)

- ✿ Collocation : phrase courte dont le sens va au-delà de la combinaison des sens de ses mots
  - ◆ Elle a un sémantique unique
- ✿ Il faut détecter les collocations dans une phrase pour que les applications TAL soient performantes
- ✿ But : savoir quand ces mots ont un sens différent que lorsqu'ils sont utilisés individuellement

## ✿ Exemples : Make-Up

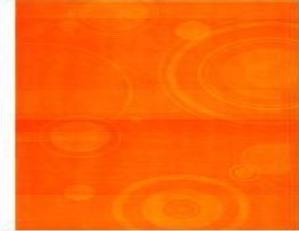
# Collocations (2/2)

✿ New York Times  
corpus :

- ◊ Recherche des pairs de mots consécutifs dans le corpus
  - ◊ Sélection des pairs non-nom ou nom-adj
  - ◊ Garder les plus fréquents
- collocations ou lieux

| Frequency | Word 1    | Word 2    | Part-of-speech pattern |
|-----------|-----------|-----------|------------------------|
| 11487     | New       | York      | A N                    |
| 7261      | United    | States    | A N                    |
| 5412      | Los       | Angeles   | N N                    |
| 3301      | last      | year      | A N                    |
| 3191      | Saudi     | Arabia    | N N                    |
| 2699      | last      | week      | A N                    |
| 2514      | vice      | president | A N                    |
| 2378      | Persian   | Gulf      | A N                    |
| 2161      | San       | Francisco | N N                    |
| 2106      | President | Bush      | N N                    |
| 2001      | Middle    | East      | A N                    |
| 1942      | Saddam    | Hussein   | N N                    |
| 1867      | Soviet    | Union     | A N                    |
| 1850      | White     | House     | A N                    |
| 1633      | United    | Nations   | A N                    |
| 1337      | York      | City      | N N                    |
| 1328      | oil       | prices    | N N                    |
| 1210      | next      | year      | A N                    |
| 1074      | chief     | executive | A N                    |
| 1073      | real      | estate    | A N                    |

# Modèle statistique du langage



- ✿ Transformer des corpus en types et occurrences
- ✿ Transformer les occurrences en distributions de probabilités
  - modèle statistique du langage
- ✿ Un modèle statistique du langage est une distribution de probabilités sur des mots ou suites de mots.
- ✿ Il permet de :
  - ◊ Attribuer une probabilité à une phrase
  - ◊ classer les mots ou les phrases selon leur probabilité d'apparition





# Modèle statistique du langage

Chapitre 3

# Définition

- ✿ Un modèle de langue est une distribution de probabilités sur des séquences de mots (p. ex. des phrases) :

$$P(w_1, w_2, \dots, w_n) = P(W_1^n)$$

Où  $w_i$  sont les mots de la séquence, et  $w_1, \dots, w_n$  = est la séquence complète

# Problématique (1/2)

## ✿ Traduction automatique

$P(\text{high winds tonite}) > P(\text{Large winds tonite})$

## ✿ Correcteur d'orthographe

The office is about fifteen minuets from my house

$P(\text{about fifteen minutes}) > P(\text{about fifteen minuets})$

## ✿ Reconnaissance de la parole

$P(\text{I saw a van}) > P(\text{eyes awe of an})$

# Problématique (2/2)



## ✿ But :

- ◊ Associer une probabilité à toute suite de mots
- ◊ Estimer la probabilité  $P(W)$  d'un mot ou d'une suite de mots  $W = w_1w_2...w_n$

## ✿ Il existe différentes façons de modéliser cette distribution :

- ◊ Souvent on commence par décomposer la probabilité jointe en un produit de probabilités conditionnelles
- ◊ Connaissant la probabilité d'une phrase, quel est le mot suivant le plus probable?

## ✿ Rappel : probabilité conditionnelle :

$$P(A,B) = P(A/B) \times P(B)$$

## ✿ Avec plus de variable !

$$P(A,B,C,D) = P(A) \times P(B/A) \times P(C/A,B) \times P(D/A,B,C)$$



# Règle de la chaîne

$$\begin{aligned}P(\text{"une suite de cinq mots"}) &= \\P(\text{"une"}) \times & \\P(\text{"suite"} \mid \text{"une"}) \times & \\P(\text{"de"} \mid \text{"une suite"}) \times & \\P(\text{"cinq"} \mid \text{"une suite de"}) \times & \\P(\text{"mots"} \mid \text{"une suite de cinq"})\end{aligned}$$

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

✿ En pratique, ces probabilités sont :

- ◊ Difficile à estimer
- ◊ Difficile à stocker

# Approximation par n-grammes

On fait l'hypothèse que chaque mot ne dépend que des  $(n-1)$  mots qui le précèdent : hypothèse de Markov

## ✿ Trigramme

- ◊  $n = 3$
- ◊ un mot dépend des 2 mots qui le précèdent

## ✿ Bigramme

- ◊  $n = 2$
- ◊ un mot dépend du mot qui le précède

## ✿ Unigramme

- ◊  $n = 1$
- ◊ un mot ne dépend d'aucun mot qui le précède



# Décomposition en Trigrammes

$P("une suite de cinq mots") =$

$P("une") \times$

$P("suite" | "une") \times$

$P("de" | "une suite") \times$

$P("cinq" | "suite de") \times$

$P("mots" | "de cinq")$

# Ecriture mathématique



- ✿ Le but d'un modèle de langage est de déterminer la probabilité  $P(w_1 \dots w_n)$  d'une suite de mots  $w_1, \dots, w_n$

$$P(w_1 \dots w_n) = P(w_1) \times P(w_2 / w_1) \times \dots \times P(w_i / w_1 \dots w_{i-1})$$

- ✿ L'hypothèse trigramme rend cette évaluation possible en approximant

$$P(w_i / w_1 \dots w_{i-1}) \approx P(w_i / w_{i-2} w_{i-1})$$

# Apprentissage



## ✿ Comment apprendre la distribution de probabilités ?

- ❖ Combien vaut  $P(\text{"de"} \mid \text{"une suite"})$  ?
- ❖ Combien vaut  $P(\text{"cinq"} \mid \text{"suite de"})$  ?
- ❖ Combien vaut  $P(\text{"mots"} \mid \text{"de cinq"})$  ?

## ✿ Plus généralement, comment estimer $P(w_i / w_{i-2} w_{i-1})$ ?

## ✿ On apprend la distribution de probabilité des n-grams à partir du corpus



# Prétraitement nécessaire avant le modèle de langage



## ✿ Comment convertir du texte en séquences ?

- ❖ dépend de l'application

## ✿ On doit déterminer les facteurs suivants :

- ❖ Comment délimiter les séquences (phrases, paragraphes ou documents) ?
- ❖ Comment délimiter les mots ?
- ❖ Comment normaliser les mots ?
- ❖ Quels mots font partie du vocabulaire du modèle ?



# Normalisation du Corpus

## \* L'écriture des corpus est normalisée

- ◊ codage (UTF-8, ISO-8859-1, ...)
- ◊ segmentation en mots
- ◊ suppression de la ponctuation
- ◊ réduction en minuscules
- ◊ ...

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

## \* On collecte du corpus, on normalise et on compte !

$$P("de" | "une suite") = \frac{N("une suite de")}{N("une suite")}$$

où  $N(.)$  exprime le nombre d'occurrences

Estimation par maximum de  
vraisemblance

# Normalisation : étape 1

Délimiter les séquences

## Comment délimiter les séquences ?

- ❖ dépend du type de séquences qu'on souhaite modéliser
- ❖ il est commun de délimiter nos données en phrases, puisque c'est le type de séquences le plus souvent manipulé (p. ex. traduction automatique)

## On marque souvent cette délimitation avec les token spéciaux < s > et < /s >

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

# Normalisation : étape 2

Délimiter les mots

## ✿ Comment délimiter les mots ?

- ◊ Chercher les tokens par application de la tokenisation

## ✿ En anglais et en français, on peut :

- ◊ séparer en fonction des espaces et des ponctuations
- ◊ traiter les ponctuations comme faisant partie de la séquence

## ✿ Parfois, on va vouloir ignorer les ponctuations

- ◊ la ponctuation est souvent ignorée en reconnaissance de la parole, puisqu'elle n'est pas vraiment prononcée

# Normalisation : étape 3

Normaliser les mots

## ✿ Comment normaliser les mots?

### ◊ garde-t-on les majuscules ?

- non en recherche d'information, oui en traduction automatique

### ◊ est-ce qu'on réduit au lemme ?

- oui en classification de documents, non dans la plupart des cas

### ◊ applique-t-on d'autres conversions ?

- tous les nombres vers <NUMBER>, toutes les dates vers <DATE>, etc.

### ◊ dépend beaucoup de l'application

- ça vaut la peine de ne pas se commettre à un seul choix et de procéder par essais et voir les erreurs

# Exemple 1

## ✿ Texte d'origine

"Rien ne relie physiquement la Terre et la Lune. Pourtant, la Terre tire constamment la Lune vers nous. La Lune suit une ligne tangente à son orbite autour de la Terre. La Lune contrôle les marées et les cycles de vie de bon nombre d'animaux."

## ✿ Texte normalisé

< s > rien ne relie physiquement la terre et la lune < /s >

< s > pourtant la terre tire constamment la lune vers nous < /s >

< s > la lune suit une ligne tangente à son orbite autour de la terre < /s >

< s > la lune contrôle les marées et les cycles de vie de bon nombre d' animaux < /s >

# Combien de unigrammes?

|            |                |               |
|------------|----------------|---------------|
| la 7       | suit 1         | marées 1      |
| lune 4     | son 1          | ligne 1       |
| terre 3    | rien 1         | d' 1          |
| de 3       | relie 1        | cycles 1      |
| les 2      | pourtant 1     | contrôle 1    |
| et 2       | physiquement 1 | constamment 1 |
| vie 1      | orbite 1       | bon 1         |
| vers 1     | nous 1         | autour 1      |
| une 1      | nombre 1       | animaux 1     |
| tire 1     | ne 1           | à 1           |
| tangente 1 |                |               |

Cas spécial : unigramme

$$P(w_i) = \frac{c(w_i)}{N}$$

nb. de mots, incluant les  
fins de phrase </s>, mais  
pas les débuts <s>

Sans compter la balise <s>!

# Probabilités des unigrammes

|                  |                      |                     |
|------------------|----------------------|---------------------|
| la 0,15217       | suit 0,02174         | marées 0,02174      |
| lune 0,08696     | son 0,02174          | ligne 0,02174       |
| terre 0,06522    | rien 0,02174         | d' 0,02174          |
| de 0,06522       | relie 0,02174        | cycles 0,02174      |
| les 0,04348      | pourtant 0,02174     | contrôle 0,02174    |
| et 0,04348       | physiquement 0,02174 | constamment 0,02174 |
| vie 0,02174      | orbite 0,02174       | bon 0,02174         |
| vers 0,02174     | nous 0,02174         | autour 0,02174      |
| une 0,02174      | nombre 0,02174       | animaux 0,02174     |
| tire 0,02174     | ne 0,02174           | à 0,02174           |
| tangente 0,02174 |                      |                     |

## Exemple 2 : Corpus de Berkeley



can you tell me about any good cantonese restaurants close by  
mid priced thai food is what i'm looking for  
tell me about chez panisse  
can you give me a listing of the kinds of food that are available  
i'm looking for a good place to eat breakfast  
when is caffe venezia open during the day



# Combien de bigrammes?

Corpus de Berkeley

## Bigrammes (sous-ensemble)

|                | <b>i</b> | <b>want</b> | <b>to</b> | <b>eat</b> | <b>chinese</b> | <b>food</b> | <b>lunch</b> | <b>spend</b> |
|----------------|----------|-------------|-----------|------------|----------------|-------------|--------------|--------------|
| <b>i</b>       | 5        | 827         | 0         | 9          | 0              | 0           | 0            | 2            |
| <b>want</b>    | 2        | 0           | 608       | 1          | 6              | 6           | 5            | 1            |
| <b>to</b>      | 2        | 0           | 4         | 686        | 2              | 0           | 6            | 211          |
| <b>eat</b>     | 0        | 0           | 2         | 0          | 16             | 2           | 42           | 0            |
| <b>chinese</b> | 1        | 0           | 0         | 0          | 0              | 82          | 1            | 0            |
| <b>food</b>    | 15       | 0           | 15        | 0          | 1              | 4           | 0            | 0            |
| <b>lunch</b>   | 2        | 0           | 0         | 0          | 0              | 1           | 0            | 0            |
| <b>spend</b>   | 1        | 0           | 1         | 0          | 0              | 0           | 0            | 0            |

## Unigrammes (sous-ensemble)

| <b>i</b> | <b>want</b> | <b>to</b> | <b>eat</b> | <b>chinese</b> | <b>food</b> | <b>lunch</b> | <b>spend</b> |
|----------|-------------|-----------|------------|----------------|-------------|--------------|--------------|
| 2533     | 927         | 2417      | 746        | 158            | 1093        | 341          | 278          |

# Probabilités des bigrammes

Corpus de Berkeley

## Modèle Bigramme (sous-ensemble)

| i              | want    | to   | eat    | chinese | food    | lunch  | spend  |
|----------------|---------|------|--------|---------|---------|--------|--------|
| <b>i</b>       | 0.002   | 0.33 | 0      | 0.0036  | 0       | 0      | 0      |
| <b>want</b>    | 0.0022  | 0    | 0.66   | 0.0011  | 0.0065  | 0.0065 | 0.0054 |
| <b>to</b>      | 0.00083 | 0    | 0.0017 | 0.28    | 0.00083 | 0      | 0.0025 |
| <b>eat</b>     | 0       | 0    | 0.0027 | 0       | 0.021   | 0.0027 | 0.056  |
| <b>chinese</b> | 0.0063  | 0    | 0      | 0       | 0       | 0.52   | 0.0063 |
| <b>food</b>    | 0.014   | 0    | 0.014  | 0       | 0.00092 | 0.0037 | 0      |
| <b>lunch</b>   | 0.0059  | 0    | 0      | 0       | 0       | 0.0029 | 0      |
| <b>spend</b>   | 0.0036  | 0    | 0.0036 | 0       | 0       | 0      | 0      |

# Probabilités de la phrase de test avec bigrammes

$$P(< \text{s} > | \text{I want english food } < / \text{s} >) =$$

$$P(\text{I} | < \text{s} >)$$

$$\times P(\text{want} | \text{I})$$

$$\times P(\text{english} | \text{want})$$

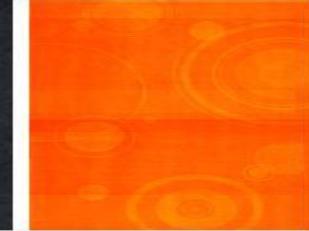
$$\times P(\text{food} | \text{english})$$

$$\times P(< / \text{s} > | \text{food})$$

$$= .000031$$



# Nombre de paramètres d'un modèle n-gram



- ✿ Toute suite de n mots doit avoir une probabilité
- ✿ Impossible en LN, car il existe trop de n-grams possibles :
  - ❖ Vocabulaire trop grand
  - ❖ Corpus pas suffisant
- ✿ En considérant un vocabulaire de 20 000 mots, on obtient :

| Modèle    | Nombre de paramètres                        |
|-----------|---|
| unigramme | 20000                                       |
| bigramme  | $20000 \times 19999 = 400$ millions         |
| trigramme | $20000^2 \times 19999 = 8$ trillions        |
| 4-gramme  | $20000^3 \times 19999 = 1,6 \times 10^{17}$ |



# Toutefois, ...

Corpus de Berkeley

## Modèle Bigramme (sous-ensemble)

|         | i       | want | to     | eat    | chinese | food   | lunch  | spend   |
|---------|---------|------|--------|--------|---------|--------|--------|---------|
| i       | 0.002   | 0.33 | 0      | 0.0036 | 0       | 0      | 0      | 0.00079 |
| want    | 0.0022  | 0    | 0.66   | 0.0011 | 0.0065  | 0.0065 | 0.0054 | 0.0011  |
| to      | 0.00083 | 0    | 0.0017 | 0.28   | 0.00083 | 0      | 0.0025 | 0.087   |
| eat     | 0       | 0    | 0.0027 | 0      | 0.021   | 0.0027 | 0.056  | 0       |
| chinese | 0.0063  | 0    | 0      | 0      | 0       | 0.52   | 0.0063 | 0       |
| food    | 0.014   | 0    | 0.014  | 0      | 0.00092 | 0.0037 | 0      | 0       |
| lunch   | 0.0059  | 0    | 0      | 0      | 0       | 0.0029 | 0      | 0       |
| spend   | 0.0036  | 0    | 0.0036 | 0      | 0       | 0      | 0      | 0       |

# Evènements rares

- ✿ Une suite de mots qui n'a pas été rencontrée dans le corpus, peut tout de même exister dans la langue
- ✿ Comment estimer  $P(\text{"suite"} \mid \text{"une de"})$  ?
  - ◊  $N(\text{"une de suite"})/N(\text{"une de"})$  ?
  - ◊ La probabilité serait 0
- ✿ Problème des probabilités nulles

◊ Le corpus est seulement un échantillon de la langue, pas sa totalité + il n'est pas nécessairement corrélé avec l'application ciblée

→ Engendre des problèmes lors de la phase de reconnaissance car la phrase n'a aucune chance d'être reconnue

# Smoothing : Lissage

- ✿ C'est à partir de la valeur des fréquences d'apparition des n-grams dans les données d'apprentissage que sont estimés les paramètres d'un modèle de langage.
  - ✿ Malheureusement, la quantité de données est en général insuffisante et certains n-grams, voire même certains mots du lexique, n'apparaissent jamais dans le corpus d'apprentissage.
- Les techniques de lissage tentent de compenser cette carence : une sorte de généralisation qui permet d'attribuer une probabilité non nulle à un événement non vu dans le corpus d'apprentissage.

# Smoothing : Lissage

- ✿ Parfois, les mots hors-vocabulaire qui sont rencontrés dans le corpus sont rangés sous l'étiquette <UNK>
- ✿ La probabilité d'un mot jamais vu en présence d'un historique (n-gram) donné est, sans lissage, nulle.
- ✿ Les méthodes de lissage lui attribuent une valeur non nulle calculée à partir d'un historique réduit.
- ✿ **But :** attribuer une faible probabilité aux mots ou n-grams non observés dans le corpus d'apprentissage

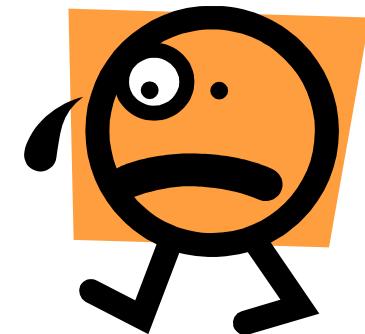
# Techniques “intuitives”

Pretend we saw each word one more time than we did

- ✿ Lissage de Laplace : Ajouter 1 à tous les n-grams :  
(Laplace or Add-one smoothing)

$$P(w_i \mid w_{i-2} w_{i-1}) = \frac{N(w_{i-2} w_{i-1} w_i) + 1}{N(w_{i-2} w_{i-1}) + V}$$

Nombre de mots  
distincts



- ✿ Ajouter  $\delta$  à tous les n-grams :  
(Add-  $\delta$  smoothing)

$$P(w_i \mid w_{i-2} w_{i-1}) = \frac{N(w_{i-2} w_{i-1} w_i) + \delta}{N(w_{i-2} w_{i-1}) + V\delta}$$

Nombre de mots  
distincts

# Simple interpolation

- ✿ Interpolation linéaire entre Tri-gramme, Bi-gramme et Uni-gramme

$$P(z | xy) = \lambda \frac{N(xyz)}{N(xy)} + \mu \frac{N(yz)}{N(y)} + (1 - \lambda - \mu) \frac{N(z)}{N(.)}$$

- ✿ Trouver les coefficients d'interpolation avec un corpus de développement

# Intuition du smoothing

- ✿ Certaines suites de mots apparaissent dans le corpus, mais pas suffisamment souvent pour être représentatives
- ✿ S'il existe 100 millions de trigrammes dans le corpus, un trigramme qui est apparu une seule fois sera ignoré
- ✿ la probabilité serait trop faible et ne représenterait pas ce que ce trigramme porte réellement comme information

# Lissage de Laplace

Corpus de Berkeley

|         | i  | want | to  | eat | chinese | food | lunch | spend |
|---------|----|------|-----|-----|---------|------|-------|-------|
| i       | 6  | 828  | 1   | 10  | 1       | 1    | 1     | 3     |
| want    | 3  | 1    | 609 | 2   | 7       | 7    | 6     | 2     |
| to      | 3  | 1    | 5   | 687 | 3       | 1    | 7     | 212   |
| eat     | 1  | 1    | 3   | 1   | 17      | 3    | 43    | 1     |
| chinese | 2  | 1    | 1   | 1   | 1       | 83   | 2     | 1     |
| food    | 16 | 1    | 16  | 1   | 2       | 5    | 1     | 1     |
| lunch   | 3  | 1    | 1   | 1   | 1       | 2    | 1     | 1     |
| spend   | 2  | 1    | 2   | 1   | 1       | 1    | 1     | 1     |

# Lissage de Laplace

Corpus de Berkeley

$$P(w_n | w_{n-1}) = \frac{P(w_{n-1} w_n) + 1}{P(w_{n-1}) + V}$$

|         | i              | want           | to             | eat            | chinese        | food           | lunch          | spend          |
|---------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| i       | 0.0015         | 0.21           | <b>0.00025</b> | 0.0025         | <b>0.00025</b> | <b>0.00025</b> | <b>0.00025</b> | 0.00075        |
| want    | 0.0013         | <b>0.00042</b> | 0.26           | 0.00084        | 0.0029         | 0.0029         | 0.0025         | 0.00084        |
| to      | 0.00078        | <b>0.00026</b> | 0.0013         | 0.18           | 0.00078        | <b>0.00026</b> | 0.0018         | 0.055          |
| eat     | <b>0.00046</b> | <b>0.00046</b> | 0.0014         | <b>0.00046</b> | 0.0078         | 0.0014         | 0.02           | <b>0.00046</b> |
| chinese | 0.0012         | <b>0.00062</b> | <b>0.00062</b> | <b>0.00062</b> | <b>0.00062</b> | 0.052          | 0.0012         | <b>0.00062</b> |
| food    | 0.0063         | <b>0.00039</b> | 0.0063         | <b>0.00039</b> | 0.00079        | 0.002          | <b>0.00039</b> | <b>0.00039</b> |
| lunch   | 0.0017         | <b>0.00056</b> | <b>0.00056</b> | <b>0.00056</b> | <b>0.00056</b> | 0.0011         | <b>0.00056</b> | <b>0.00056</b> |
| spend   | 0.0012         | <b>0.00058</b> | 0.0012         | <b>0.00058</b> | <b>0.00058</b> | <b>0.00058</b> | <b>0.00058</b> | <b>0.00058</b> |

# Techniques de smoothing

## ✿ Jelinek-Mercer

$$P_{smooth}(z \mid xy) = \lambda(N(xy)) \frac{N(xyz)}{N(xy)} + (1 - \lambda(N(xy))) P_{smooth}(z \mid y)$$

- ✿ Good-Turing
- ✿ Katz
- ✿ Absolute Discounting
- ✿ Interpolated Absolute Discount
- ✿ Interpolated Multiple Absolute Discounts
- ✿ Kneser-Ney
- ✿ Kneser-Ney Modified

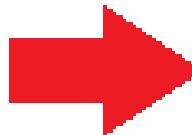
# Fréquence Nc

- ✿ Notation :  $N_c = \text{Nombre de } n\text{-grammes qu'on a vu } c \text{ fois}$ 
  - ◊ Fréquence de la fréquence c

- ✿ Exemple : Sam I am I am sam I do not eat

|     |   |           |
|-----|---|-----------|
| I   | 3 |           |
| sam | 2 | $N_1 = 3$ |
| am  | 2 | $N_2 = 2$ |
| do  | 1 |           |
| not | 1 |           |
| eat | 1 |           |

Unigrammes



# Lissage de Good-Turing

Selon Good-Turing, la probabilité d'observer **n'importe quel** mot jamais vu dans le corpus d'entraînement sera

$$P_{GT}^*(\text{things with frequency zero in training}) = \frac{N_1}{N_0 N}$$

On peut généraliser cette règle à n'importe quel mot de fréquence  $c$

- la fréquence ajustée  $c^*$  est alors

$$c^* = (c + 1) \frac{N_{c+1}}{N_c}$$

$$p_{GT}(x : c) = \frac{c^*}{N}$$



# Evaluation

- ✿ Comment savoir si un modèle de langage est meilleur qu'un autre ?
- ✿ Tester le modèle appris : on peut introduire les modèles dans un système de reconnaissance automatique, et estimer le taux d'erreur → C'est très lent !
- ✿ Perplexité :

◊ Soit un corpus de test  $W = w_1, w_2, \dots, w_n$

$$PP(W) = \sqrt[n]{\prod_{i=1}^n \frac{1}{P(w_i | w_1 \dots i-1)}}$$

◊ Le modèle de langage qui a la plus petite perplexité sur un même corpus de test est le meilleur

# Perplexité : Exemple

**Lower perplexity = better model**

Training 38 million words, test 1.5 million words, WSJ

| N-gram<br>Order | Unigram | Bigram | Trigram |
|-----------------|---------|--------|---------|
| Perplexity      | 962     | 170    | 109     |

# Remarques

## ✿ Données d'apprentissage suffisantes

- ◊ Plus  $n$  est grand ( $n$ -gram), plus la perplexité est réduite

## ✿ Données limitées

- ◊ Si  $n$  est très grand, la perplexité diminue

## ✿ Plusieurs travaux portant sur le TAL utilise 5-grams or 6-grams



# Exercice1

✿ Soit le corpus suivant :

I am Sam. Sam I am. I do not like green eggs and ham.

✿ Calculer les probabilités du Modèle bi-grammes:

# Solution

✿ Soit le corpus suivant :

I am Sam. Sam I am. I do not like green eggs and ham.

✿ Normalisation :

< s > i am sam < /s >

< s > sam i am < /s >

< s > i do not like green eggs and ham < /s >

✿ Modèle bi-grams:

$$\diamond P(i/< s >) = 2/3 = 0.67$$

$$\diamond P(sam/< s >) = 1/3 = 0.33$$

$$\diamond P(< /s >/sam) = 1/2 = 0.5$$

# Exercice 2

\* Soit la phrase : I am Sam

1. Extraire les unigrammes
2. Extraire les bigrammes
3. Extraire les trigrammes

# Exercice 3

✿ Soit le corpus d'apprentissage suivant :

« he is he is he is going abroad is going to study in the field »

✿ Calculer  $P(\text{is going abroad to study})$  avec :

- ◊ Le modèle bi-grams sans lissage ?
- ◊ Le modèle bi-grams avec lissage de Laplace?
- ◊ Le modèle bi-grams avec lissage de Good Turing?



# Solution

✿ Soit le corpus d'apprentissage suivant :

« he is he is he is going abroad is going to study in the field »

✿  $P(\text{is going abroad to study})$  ?

- ◊ Modèle bi-grams sans lissage ? = 0
- ◊ Modèle bi-grams avec lissage de Laplace?  $\approx 0.000828$
- ◊ Le modèle bi-grams avec lissage de Good Turing?  $\approx 1.24 \times 10^{-6}$



**MERCI POUR VOTRE  
ATTENTION**

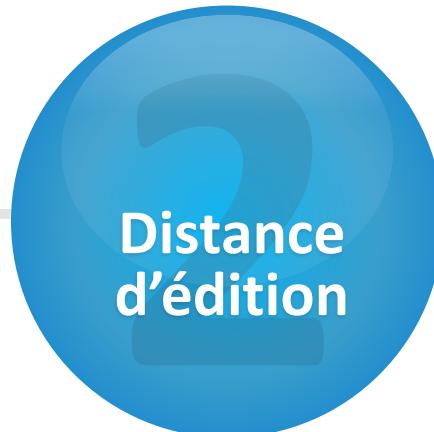
**DES QUESTIONS ? —**



## Partie 5

# Les attributs textuels

# Plan



1

# Expressions régulières

reg[ular]  
expr[essio]n

Chapitre 4

127



# Expressions régulières (1/2)



- ✿ Une expression régulière : langage formel pour spécifier des chaînes de texte
- ✿ Une expression régulière = façon simple de caractériser un ensemble de chaînes de caractère de façon compacte

woodchuck

woodchucks

Woodchuck

Woodchucks



- ✿ Comment peut-on chercher un de ces termes?

# Expressions régulières (2/2)

✿ Définir des motifs pour reconnaître dans les textes des chaînes de caractères ayant une propriété quelconque

→ C'est un formalisme utile pour l'extraction d'informations dans les données textuelles

✿ Exemple : on recherche tous les mots :

- ◊ terminés par le suffixe "able"
- ◊ toutes les phrases contenant le mot "enfant",
- ◊ toutes les phrases commençant par "Je", se terminant par un point d'interrogation, etc.

# Operations sur les ER :

## Disjonctions

### Disjonction : Lettres entre crochets []

- Pour correspondre à plus d'une chaîne, on utilise des disjonctions de caractères à l'aide de []

| Modèle       | Correspondance         |
|--------------|------------------------|
| [wW]oodchuck | woodchuck, Woodchuk    |
| [abc]        | a,b,c                  |
| [1234567890] | N'importe quel chiffre |

- On peut aussi faire des disjonctions d'intervalles

| Modèle | Correspondance                        |
|--------|---------------------------------------|
| [A-Z]  | N'importe quel caractère entre A et Z |
| [a-z]  | N'importe quel caractère entre a et z |
| [0-9]  | N'importe quel chiffre entre 0 et 9   |

# Exemples

## Regular Expression

/[a-z]/g

Javascript ▾



32 matches

## Test String

We looked!  
Then we saw him step in on the mat.

## Regular Expression

/[em]/g

## Test String

We looked!  
Then we saw him step in on the mat.

131

## Regular Expression

/[!]/g

## Test String

We looked!  
Then we saw him step in on the mat.

# Operations sur les ER :

## Disjonction de chaînes

### ✿ Disjonction de chaînes :

- ❖ [] : disjonction de caractère individuel (un seul caractère)
- ❖ Pour faire une disjonction de chaîne de caractères : |

| Modèle                    | Correspondance                                      |
|---------------------------|---|
| groundhog woodchuck       | groundhog ou woodchuck                              |
| a b c                     | [abc]   |
| [gG]roundhog [Ww]oodchuck | groundhog ou Woodchuck ou<br>Groundhog ou woodchuck |

- ❖ Peut être appliquée à une sous-chaîne de l'ER : ()
  - /gupp(y|ies)/ correspond à guppy ou guppies

# Exemples

RegEx Pal

From Dan's Tools

Regular Expression

```
/looked|step/g
```

Test String

We looked!

Then we saw him step in on the mat.

RegEx Pal

From Dan's Tools

Regular Expression

```
/at|ook/g
```

Test String

We looked!

Then we saw him step in on the mat.

# Operations sur les ER :

## Négation

### ✿ Négation

- ◆ Le symbole **^** permet de formuler une négation
- ◆ **^** signifie négation seulement s'il est mis en premier entre [ ]

| Modèle | Correspondance       |
|--------|----------------------|
| [^A-Z] | Lettre non Majuscule |
| [^Ss]  | Ni S ni s            |
| [^e^]  | Ni e ni ^            |
| a^b    | a^b                  |

RegEx Pal  
From Dan's Tools

Regular Expression

/[^!]/g

Test String

We looked!  
Then we saw him step in on the mat.

# Operations sur les ER :

? \* + .

## ✿ Caractère facultatif :

- ◊ Le symbole ? Permet de caractériser un caractère qui est facultativement présent
- ◊ Le caractère optionnel est celui qui précède le ?

## ✿ Répétitions :

- ◊ Les symboles \* et + permettent d'exprimer un nombre arbitraire de répétitions ( peut-être 0 pour \*)

| Modèle    | Correspondance                   |
|-----------|----------------------------------|
| colou?r   | color ou colour                  |
| a*        | a, aa, aaa, aaaa,... ainsi que Ø |
| aa* ou a+ | a, aa, aaa, aaaa,...             |
| Beg.n     | Begin begun beg3n                |

# Operations sur les ER :

^ \$

- ✿ Le caractère ^ désigne le début d'une ligne (sans [])
- ✿ Le caractère \$ désigne la fin d'une ligne

| Modèle | Correspondance              |
|--------|-----------------------------|
| ^[A-Z] | Majuscule au début de ligne |
| [A-Z]& | Majuscule à la fin de ligne |
| \.     | Le point .                  |
| .      | Caractère joker             |

Regular Expression

Test String

```
We looked!
Then we saw him step in on the mat.
```

Regular Expression

Test String

```
We looked!
Then we saw him step in on the mat.
```

Regular Expression

Test String

```
We looked!
Then we saw him step in on the mat.
```

# Exemples

RegEx Pal

From Dan's Tools

Regular Expression

/o+/g

Test String

We looked!

Then we saw him step in on the mat.

RegEx Pal

From Dan's Tools

Regular Expression

/^ [A-Z] /gm

Regular Expression

/^ [A-Z] /gm

Test String

We looked!

Then we saw him step in on the mat.

Test String

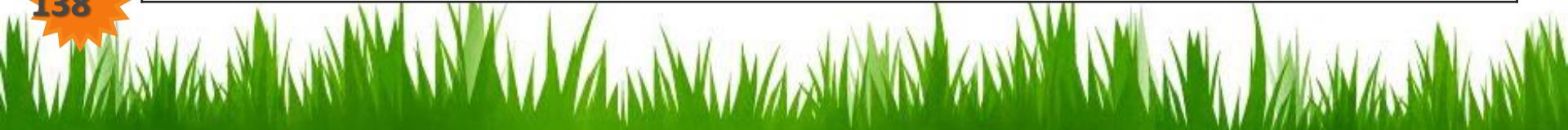
We looked!

Then we saw him step in on the mat

# Caractères spéciaux



| caractère    | Signification   |
|--------------|---|
| $X \{n\}$    | $X$ apparaît exactement $n$ fois.                         |
| $X \{n,\}$   | $X$ apparaît au moins $n$ fois.                           |
| $X \{n, m\}$ | $X$ apparaît au moins $n$ fois et au plus $m$ fois.       |
| \b           | Le début ou la fin d'un mot.                              |
| \B           | Le début ou la fin d'un élément qui n'est pas un mot.     |
| \A           | Le début d'une entrée.                                    |
| \G           | La fin du morceau de texte qui a été trouvé précédemment. |
| \Z           | La fin d'une entrée, sauf s'il s'agit de la fin du texte. |



# Exercice

✿ Soit le texte suivant :

We looked!

Then we saw him step in on the mat.

The cat in the Hat!

The other one there, the blithe one.

[Tt]he

✿ Trouver toute les occurrence du mot the

# Solution

✿ Soit le texte suivant :

We looked!

Then we saw him step in on the mat.

The cat in the Hat!

The other one there, the blithe one.

✿ Trouver toute les occurrence du mot the

[^A-Za-z][Tt]he[^A-Za-z]

# Exercices

## ✿ Ecrire l'expression régulière qui permet de générer :

◊ Une séquence de 1 à 3 lettres, suivie d'une séquence de 1 à deux chiffres, suivie de la même séquence de chiffres suivie de la première séquence de lettres.

- Exemples : ab55ab, abC4141aBc,...

( ?i)(\w{1,3})(\d{1,2})\2\1

◊ Un mot de passe : il doit contenir au moins un chiffre, au moins une lettre et au moins un caractère de ponctuation. De plus il doit avoir au moins 8 caractères.

- Exemple : abce.fr/ty2

( ?i)(? = . \* \d)(? = .\*[a-zA-Z])(? = . \* \p{Punct}).{8, }



# Distance d'édition

Chapitre 4

# Distance d'édition

- ✿ Les corpus avec lesquels on travaille peuvent contenir des erreurs typographiques (coquilles)
  - ❖ Numérisation des livres
  - ❖ Livres écrits par des non-professionnels
- ✿ Pour chaque mot, une façon simple de corriger les erreurs éventuelles est de :
  - ❖ Vérifier si le mot se trouve dans un dictionnaire
  - ❖ Si non, le remplacer par le mot **le plus proche**
- ✿ La distance d'édition : outil permettant de trouver la ressemblance entre des mots

# Distance d'édition

✿ La distance d'édition est une distance générale entre deux séquences de caractères

✿ Opérations autorisées d'édition mot M → mot P:

- ◊ Insertion dans M d'un caractère de P : (i)
- ◊ Substitution d'un caractère de M par un caractère différent de P (s)
- ◊ Suppression d'un caractère de M (d)

✿ Exemple : intension → execusion

- Si le coût de chaque opération vaut 1 : D=5
- Si le coût de « Substitution » vaut 2 :
  - Distance d'édition = distance de Levenshtein
  - D=8

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| I | N | T | E | * | N | T | I | O | N |
|   |   |   |   |   |   |   |   |   |   |
| * | E | X | E | C | U | T | I | O | N |

d s s      i s

# Calcul de la distance d'édition



## ✿ Calculer la matrice des distances dynamique :

- ❖ On calcul  $D(i,j)$  pour les petites valeurs de  $i$  et  $j$  (petites chaînes)
- ❖ Calculer les grandes distances  $D(i,j)$  en fonctions des distances calculées précédemment

Conditions initiales:  $D(i,0) = i, \quad \forall i \quad 0 \leq i \leq m$

$D(0,j) = j, \quad \forall j \quad 0 \leq j \leq n$

Relation de récurrence pour  $i, j > 0$ :

$$D(i,j) = \min \begin{cases} D(i-1,j) & +1 \\ D(i,j-1) & +1 \\ D(i-1,j-1)+\delta(i,j) \end{cases},$$

Où  $\delta(i,j) = 0$  si  $x_i = y_j$  et 1 sinon.



# Exemple : Distance d'édition (de Levenshtein)

- Initialization

$$D(i, 0) = i$$

$$D(0, j) = j$$

- Recurrence Relation:

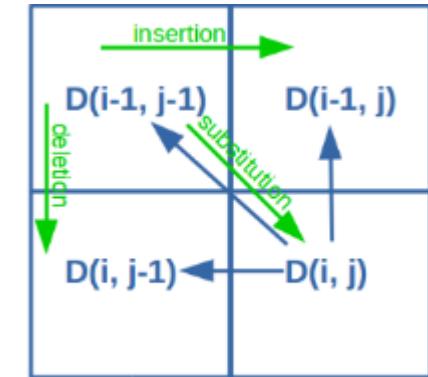
For each  $i = 1 \dots M$

For each  $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) \\ 0; \quad \text{if } X(i) = Y(j) \end{cases} \\ & \text{substitution} \end{cases}$$

- Termination:

$D(N, M)$  is distance



**δ = 2**

# Matrice d'édition : étape 1

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
|   | - | I | N | T | E | N | T | I | O | N |
| - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| E | 1 |   |   |   |   |   |   |   |   |   |
| X | 2 |   |   |   |   |   |   |   |   |   |
| E | 3 |   |   |   |   |   |   |   |   |   |
| C | 4 |   |   |   |   |   |   |   |   |   |
| U | 5 |   |   |   |   |   |   |   |   |   |
| T | 6 |   |   |   |   |   |   |   |   |   |
| I | 7 |   |   |   |   |   |   |   |   |   |
| O | 8 |   |   |   |   |   |   |   |   |   |
| N | 9 |   |   |   |   |   |   |   |   |   |

# Matrice d'édition : étape 2

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
|   | - | I | N | T | E | N | T | I | O | N |
| - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| E | 1 | 2 |   |   |   |   |   |   |   |   |
| X | 2 |   |   |   |   |   |   |   |   |   |
| E | 3 |   |   |   |   |   |   |   |   |   |
| C | 4 |   |   |   |   |   |   |   |   |   |
| U | 5 |   |   |   |   |   |   |   |   |   |
| T | 6 |   |   |   |   |   |   |   |   |   |
| I | 7 |   |   |   |   |   |   |   |   |   |
| O | 8 |   |   |   |   |   |   |   |   |   |
| N | 9 |   |   |   |   |   |   |   |   |   |

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases}$$

# Matrice d'édition : étape 2



|   | - | I | N | T | E  | N  | T  | I  | O  | N  |
|---|---|---|---|---|----|----|----|----|----|----|
| - | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
| E | 1 | 2 | 3 | 4 | 3  | 4  | 5  | 6  | 8  | 9  |
| X | 2 | 3 | 4 | 5 | 4  | 5  | 6  | 7  | 7  | 8  |
| E | 3 | 4 | 5 | 6 | 5  | 6  | 7  | 8  | 9  | 10 |
| C | 4 | 5 | 6 | 7 | 6  | 7  | 8  | 9  | 10 | 11 |
| U | 5 | 6 | 7 | 8 | 7  | 8  | 9  | 10 | 11 | 12 |
| T | 6 | 7 | 8 | 7 | 8  | 9  | 8  | 9  | 10 | 11 |
| I | 7 | 6 | 7 | 8 | 9  | 10 | 9  | 8  | 9  | 10 |
| O | 8 | 7 | 8 | 9 | 10 | 11 | 10 | 9  | 8  | 9  |
| N | 9 | 8 | 7 | 8 | 9  | 10 | 11 | 10 | 9  | 8  |



# pseudocode

**function** MIN-EDIT-DISTANCE(*target, source*) **returns** *min-distance*

*n*  $\leftarrow$  LENGTH(*target*)

*m*  $\leftarrow$  LENGTH(*source*)

Create a distance matrix *distance*[*n+1,m+1*]

Initialize the zeroth row and column to be the distance from the empty string  
*distance*[0,0] = 0

**for** each column *i* **from** 1 **to** *n* **do**

*distance*[*i*,0]  $\leftarrow$  *distance*[*i*-1,0] + ins-cost(*target*[*i*])

**for** each row *j* **from** 1 **to** *m* **do**

*distance*[0,*j*]  $\leftarrow$  *distance*[0,*j*-1] + del-cost(*source*[*j*])

**for** each column *i* **from** 1 **to** *n* **do**

**for** each row *j* **from** 1 **to** *m* **do**

*distance*[*i*,*j*]  $\leftarrow$  MIN( *distance*[*i*-1,*j*] + ins-cost( *target*[*i*] ),

*distance*[*i*-1,*j*-1] + sub-cost( *source*[*j*], *target*[*i*] ),

*distance*[*i*,*j*-1] + del-cost( *source*[*j*] ) )

**return** *distance*[*n,m*]

# Matrice d'édition avec Alignement

- ✿ Au cours de la remplissage de la matrice d'édition, on garde des pointeurs :
  - ❖ Vers la case « minimisante »
  - ❖ Une fois le calcul des distances terminé, on suit les pointeurs en reculant de la case  $D(n,m)$  vers  $D(0,0) \rightarrow$  alignement optimal
- ✿ Une case peut contenir plusieurs pointeurs → plusieurs alignements optimaux possibles

# Matrice d'édition de Levenshtein avec Alignment

- Base conditions:

$$D(i, 0) = i$$

$$D(0, j) = j$$

Termination:

$D(N, M)$  is distance

- Recurrence Relation:

For each  $i = 1 \dots M$

For each  $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) & \text{substitution} \\ 0; & \text{if } X(i) = Y(j) \end{cases} \end{cases}$$
$$\text{ptr}(i, j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

# Matrice d'édition de Levenshtein avec Alignment

|          |          |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>n</b> | 9        | ↓ 8      | ↙ ↖ ↓ 9  | ↙ ↖ ↓ 10 | ↙ ↖ ↓ 11 | ↙ ↖ ↓ 12 | ↓ 11     | ↓ 10     | ↓ 9      | ↙ 8      |
| <b>o</b> | 8        | ↓ 7      | ↙ ↖ ↓ 8  | ↙ ↖ ↓ 9  | ↙ ↖ ↓ 10 | ↙ ↖ ↓ 11 | ↓ 10     | ↓ 9      | ↙ 8      | ← 9      |
| <b>i</b> | 7        | ↓ 6      | ↙ ↖ ↓ 7  | ↙ ↖ ↓ 8  | ↙ ↖ ↓ 9  | ↙ ↖ ↓ 10 | ↓ 9      | ↙ 8      | ← 9      | ← 10     |
| <b>t</b> | 6        | ↓ 5      | ↙ ↖ ↓ 6  | ↙ ↖ ↓ 7  | ↙ ↖ ↓ 8  | ↙ ↖ ↓ 9  | ↙ 8      | ← 9      | ← 10     | ← 11     |
| <b>n</b> | 5        | ↓ 4      | ↙ ↖ ↓ 5  | ↙ ↖ ↓ 6  | ↙ ↖ ↓ 7  | ↙ ↖ ↓ 8  | ↙ ↖ ↓ 9  | ↙ ↖ ↓ 10 | ↙ ↖ ↓ 11 | ↙ ↖ ↓ 10 |
| <b>e</b> | 4        | ↙ 3      | ← 4      | ↙ ← 5    | ← 6      | ← 7      | ← 8      | ↙ ← 9    | ↙ ← 10   | ↓ 9      |
| <b>t</b> | 3        | ↙ ↖ ↓ 4  | ↙ ↖ ↓ 5  | ↙ ↖ ↓ 6  | ↙ ↖ ↓ 7  | ↙ ↖ ↓ 8  | ↙ 7      | ← 8      | ↙ ↖ ↓ 9  | ↓ 8      |
| <b>n</b> | 2        | ↙ ↖ ↓ 3  | ↙ ↖ ↓ 4  | ↙ ↖ ↓ 5  | ↙ ↖ ↓ 6  | ↙ ↖ ↓ 7  | ↙ ↖ ↓ 8  | ↓ 7      | ↙ ↖ ↓ 8  | ↙ 7      |
| <b>i</b> | 1        | ↙ ↖ ↓ 2  | ↙ ↖ ↓ 3  | ↙ ↖ ↓ 4  | ↙ ↖ ↓ 5  | ↙ ↖ ↓ 6  | ↙ ↖ ↓ 7  | ↙ 6      | ← 7      | ← 8      |
| #        | <b>0</b> | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        |
|          | #        | <b>e</b> | <b>x</b> | <b>e</b> | <b>c</b> | <b>u</b> | <b>t</b> | <b>i</b> | <b>o</b> | <b>n</b> |

# Exercices

\* Donner les matrices d'édition qui permettent de calculer les distances entre :

- ◊ Elephant et relevant
- ◊ park et spake
- ◊ Google et look at



**MERCI POUR VOTRE  
ATTENTION**

**DES QUESTIONS ? —**



## Partie 6

# Indexation : Techniques de pondération d'un texte (des tokens aux attributs)

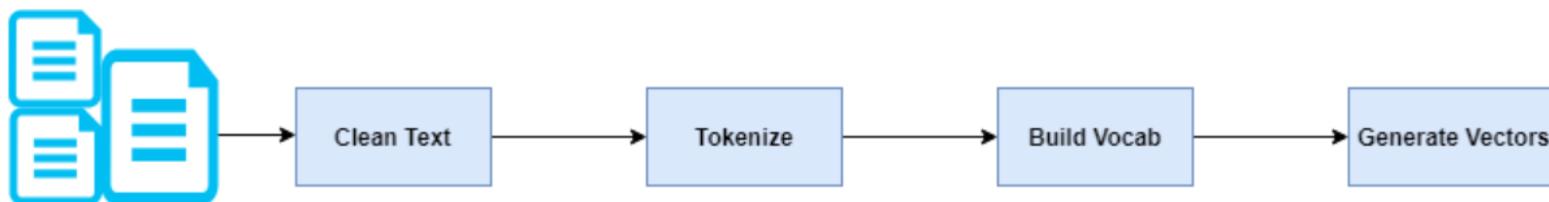
# Bag of words (BOW)

✿ BOW est une approche largement utilisée en :

- ◊ Récupération d'informations
- ◊ Classifications de documents



✿ Le processus comporte les étapes suivantes:



# Bag of words (BOW)

Motivation : chercher des mots-marqueurs comme « excellent » ou « décevant » pour l'étude des sentiments dans les tweets



- ✿ comptage des occurrences d'un segment (token) particulier dans le texte
- ✿ Pour chaque token, nous aurons un attribut numérique → vectorisation du texte

|                  |
|------------------|
| good movie       |
| not a good movie |
| did not like     |

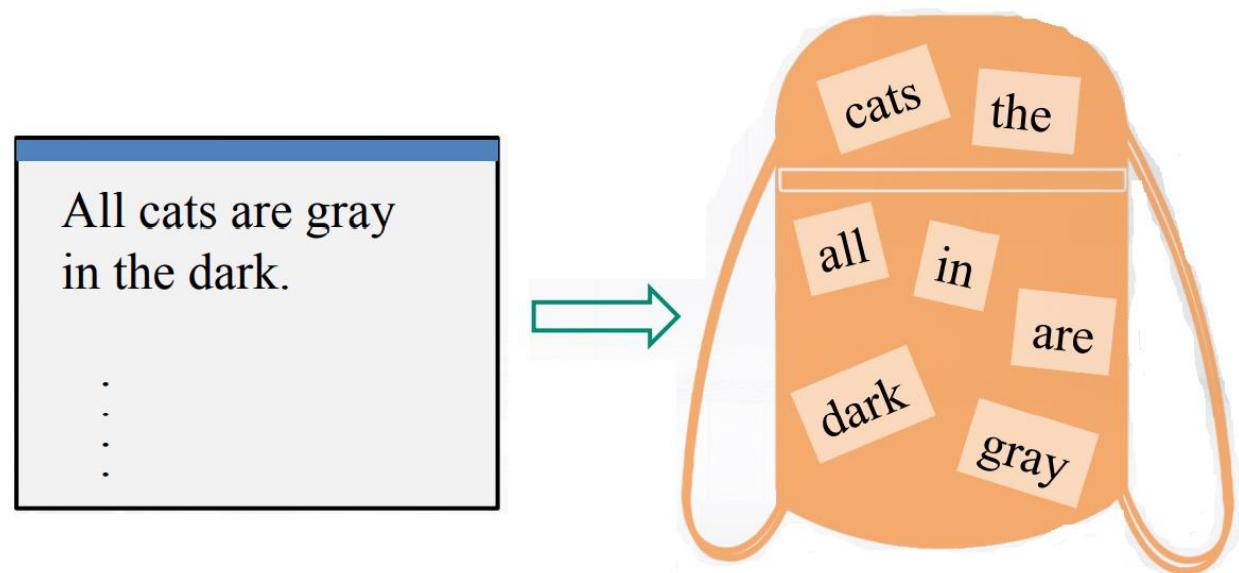


| good | movie | not | a | did | like |
|------|-------|-----|---|-----|------|
| 1    | 1     | 0   | 0 | 0   | 0    |
| 1    | 1     | 1   | 1 | 0   | 0    |
| 0    | 0     | 1   | 0 | 1   | 1    |

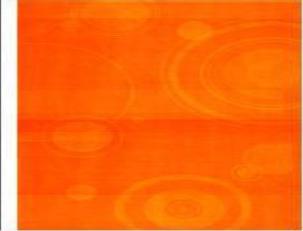
# Bag of words (BOW)

## Problèmes:

- ◊ nous perdons l'ordre de mot,
- ◊ les compteurs ne sont pas normalisés
- ◊ Problème de sparsity : beaucoup de zéros



# Un modèle plus adapté



✿ Solution : compter les paires de tokens pairs, ou les triplets de tokens, etc.. : n-grammes

- ◊ 1-gramme pour les tokens
- ◊ 2-grammes pour les paires de tokens
- ◊ 3-grammes pour les triplets de tokens

The diagram illustrates the creation of a feature vector from text data. On the left, a list of movie reviews is shown in a table:

|                  |
|------------------|
| good movie       |
| not a good movie |
| did not like     |

An arrow points from this list to a binary matrix on the right, which represents the presence or absence of specific n-grams (tokens) in the reviews. The matrix has columns for "good", "movie", "did not", "a", and "...".

| good | movie | did not | a | ... |
|------|-------|---------|---|-----|
| 1    | 1     | 0       | 0 | ... |
| 1    | 1     | 0       | 1 | ... |
| 0    | 0     | 1       | 0 | ... |

✿ Problèmes : Nombre énorme d'attributs !!

# Suppression de qq n-grammes



✿ Solution: supprimer certains n-grammes en se basant sur leurs fréquences d'apparition dans les documents de notre corpus

✿ N-grammes de haute fréquence:

- ◊ Articles, prépositions, etc. (exemple : et, a, le)
- ◊ Ils sont appelés **stop-words**, ils n'aideront pas à discriminer les textes → les supprimer

✿ N-grammes de basse fréquence :

- ◊ Typos, n-grammes rares
- ◊ Nous n'en avons pas besoin non plus, sinon nous allons probablement avoir un sur-apprentissage

✿ N-grammes de Fréquence moyenne :

- ◊ Ce sont de bons n-grammes



# Pondération par TF-IDF

- ✿ Il est utile d'examiner la fréquence n-gramme dans notre corpus pour filtrer les mauvais n-grammes
- ✿ Idée : les n-grammes avec des fréquences plus petites peuvent être plus discriminants parce qu'ils concernent des termes spécifiques dans le corpus

## ✿ TF-IDF

$$\diamond \text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

◆ Un poids élevé du TF-IDF est atteint par un terme à fréquence élevée (dans le document donné) et une faible fréquence de document dans l'ensemble corpus

# Pondération par TF-IDF

- ✿ TF/IDF : Term Frequency/Inverse Document Frequency
- ✿ TF : plus un terme t est fréquent dans un document d
- ✿ plus il est important dans la description de ce document

– Exemple de  $tf$ :

$$tf = \begin{cases} freq(t,d) & \\ 1 + \log(freq(t,d)) & \\ \frac{freq(t,d)}{\max_{\forall t' \in d}(t',d)} & \\ \frac{freq(t,d)}{\sum_{\forall t' \in d} freq(t',d)} & \end{cases}$$

# Pondération par TF-IDF

\* IDF : (Inverse Document Frequency) la fréquence du terme dans la collection

$$idf(t) = \begin{cases} \log\left(\frac{N}{n_t}\right) \\ \log\left(\frac{N - n_t}{n_t}\right) \end{cases}$$

avec

N : le nombre de documents de la collection,

$n_t$  : le nombre de documents contenant le terme t

# Exercice 1

✿ Soit un corpus contenant 3 documents :

Corpus (tiré d'œuvres de Friedrich Gottlieb Klopstock)

| Document 1  | Document 2  | Document 3   |
|---|---|--|
| Son nom est célébré par le bocage qui frémît, et par le ruisseau qui murmure, les vents l'emportent jusqu'à l'arc céleste, l'arc de grâce et de consolation que sa main tendit dans les nuages. | À peine distinguait-on deux buts à l'extrémité de la carrière : des chênes ombrageaient l'un, autour de l'autre des palmiers se dessinaient dans l'éclat du soir. | Ah ! le beau temps de mes travaux poétiques ! les beaux jours que j'ai passés près de toi ! Les premiers, inépuisables de joie, de paix et de liberté ; les derniers, empreints d'une mélancolie qui eut bien aussi ses charmes. |

✿ L'exemple porte sur le document 1 et le terme analysé est « qui ». La ponctuation et l'apostrophe sont ignorées.

# Solution de l'exercice 1



## ✿ Calcul de TF :

$$tf_{1,1} = \frac{n_{1,1}}{\sum_k n_{k,1}} = \frac{2}{38}$$

Pour les autres documents :

## ✿ Calcul de IDF :

$$idf_1 = \log \frac{|D|}{|\{d_j : t_1 \in d_j\}|} = \log \frac{3}{2}$$

## ✿ Poids final

Le premier document apparaît ainsi comme « le plus pertinent »

$$tfidf_{1,1} = \frac{2}{38} \cdot \log \frac{3}{2} \approx 0,0092$$



# Exercice 2

✿ Soient 5 documents différents dans le corpus :

- ❖ D1 = "If it walks like a duck and quacks like a duck, it must be a duck."
- ❖ D2 = "Beijing Duck is mostly prized for the thin, crispy duck skin with authentic versions of the dish serving mostly the skin."
- ❖ D3 = "Bugs' ascension to stardom also prompted the Warner animators to recast Daffy Duck as the rabbit's rival, intensely jealous and determined to steal back the spotlight while Bugs remained indifferent to the duck's jealousy or used it to his advantage. This turned out to be the recipe for the success of the duo."
- ❖ D4 ="6:25 PM 1/7/2007 blog entry: I found this great recipe for Rabbit Braised in Wine on [cookingforengineers.com](http://cookingforengineers.com)."
- ❖ D5 = "Last week Li has shown you how to make the Sechuan duck. Today we'll be making Chinese dumplings (Jiaozi), a popular dish that I had a chance to try last summer in Beijing. There are many recipies for Jiaozi

Pour la requête Q = "Beijing duck recipe", Trouver les deux documents les plus pondérés selon le TF/IDF score.

# Améliorer BOW

- ❶ Remplacer les compteurs par TF-IDF
- ❷ Normaliser le résultat par ligne

| good movie       |
|------------------|
| good movie       |
| not a good movie |
| did not like     |



| good movie | movie | did not | ... |
|------------|-------|---------|-----|
| 0.17       | 0.17  | 0       | ... |
| 0.17       | 0.17  | 0       | ... |
| 0          | 0     | 0.47    | ... |

# TF-IDF en python



```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
texts = [
    "good movie", "not a good movie", "did not like",
    "i like it", "good one"
]
tfidf = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2))
features = tfidf.fit_transform(texts)
pd.DataFrame(
    features.todense(),
    columns=tfidf.get_feature_names()
)
```

|   | good movie | like     | movie    | not      |
|---|------------|----------|----------|----------|
| 0 | 0.707107   | 0.000000 | 0.707107 | 0.000000 |
| 1 | 0.577350   | 0.000000 | 0.577350 | 0.577350 |
| 2 | 0.000000   | 0.707107 | 0.000000 | 0.707107 |
| 3 | 0.000000   | 1.000000 | 0.000000 | 0.000000 |
| 4 | 0.000000   | 0.000000 | 0.000000 | 0.000000 |



**MERCI POUR VOTRE  
ATTENTION**

**DES QUESTIONS ? —**