RITAL

Information retrieval and natural language processing
Recherche d'information et traitement automatique de la langue

Master 1 DAC, semestre 2

Nicolas Thome

SCIENCES SORBONNE UNIVERSITÉ

ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

MLIA Machine Learning & Deep Learning for Information Access

Outline

Organisation de l'UE

Les données textuelles:

- correspondent à de nombreuses applications et **débouchés professionnels**
- nécessitent des **outils spécifiques**...
- ... qui ont beaucoup évolué ces dernières années
- en plus des modèles, le **savoir faire** et les **campagnes d'expériences** sont très importants
- une **évaluation** souvent délicate

$\Rightarrow$ RITAL = vous donner les clés pour attaquer ces problèmes

UE séparée en deux parties:

1. TAL: Traitement Automatique de la Langues
   - Analyse du texte, classement de documents, compréhension des phrases, etc...
     - Séances 1-4 : Nicolas Thome
     - Séance 5 : Xavier Tannier
2. RI: Recherche d'Information
   - Stockage, indexation, accès
     - Séances 6-9 : Benjamin Piwowarski
     - Séance 10 : Christophe Servan (Qwant)

Mots clés

Modèles, campagnes d'expériences, savoir-faire opérationnel

30% **Projet TAL**

- Notebooks sur la classification supervisée, non supervisée, introduction à l'apprentissage de représentations
- Performances en classification de document (x2)
- Rapport sur les campagnes d'expériences

30% **Projet RI**

- Ré-implémentation d'un article scientifique
- Expérimentations, rendu oral

40% Examen final

- Mises en situation
- Formulations légères

NLP/TALN = vaste domaine applicatif

Traitement Automatique de la Langue Naturelle = Natural Language Processing

Divisé en de multiples applications à différents niveaux d'analyse

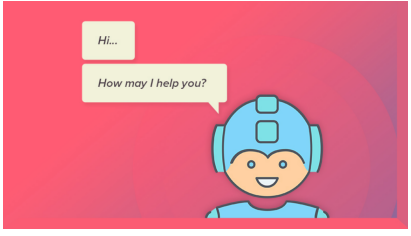
Au moins une source à étudier: C. Manning, Stanford:
`https://nlp.stanford.edu/cmanning/`
`http://web.stanford.edu/~jurafsky/NLPCourseraSlides.html`
`http://web.stanford.edu/class/cs224n/`

A guided tour on NLP applications

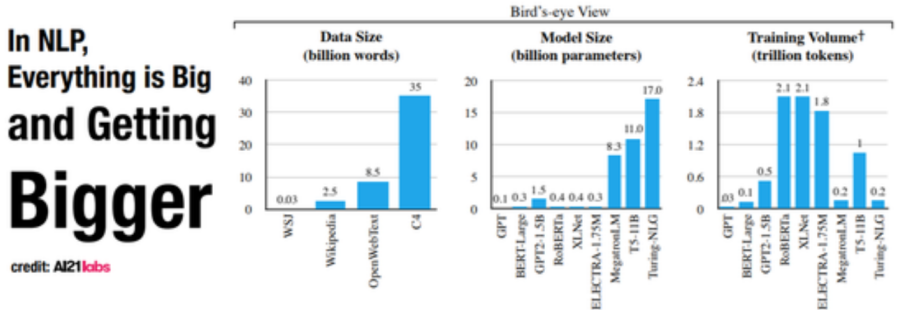
Outline

Why Natural Language Processing?

- Access Knowledge (search engine, recommender system...)
- Communicate (e.g. Translation)
- Linguistics and Cognitive Sciences (Analyse Languages themselves)



- 70 billion web-pages online (1.9 billion websites)
- 55 million Wikipedia articles
- 9000 tweets/second
- 3 million mail / second (60% spam)



Potential Users of NLP

- 7.9 billion people use some sort of language (January 2022)
- 4.7 billion internet users (January 2021) (59%)
- 4.2 billion social media users (January 2021) (54%)

Products

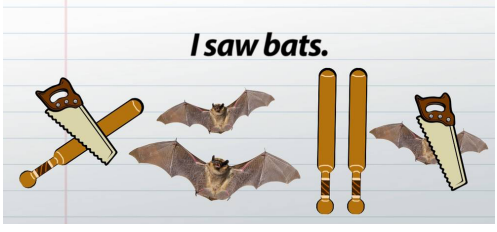
- Search: +2 billion Google users, 700 millions Baidu users
- Social Media: +3 billion users of Social media (Facebook, Instagram, WeChat, Twitter...)
- Voice assistant: +100 million users (Alexa, Siri, Google Assistant)
- Machine Translation: 500M users for google translate

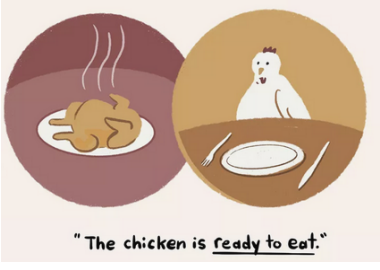
1. Productivity
   - New words, senses, structure
     - Ex: staycation, social distance added to the Oxford Dictionary in '21
2. Ambiguity
3. Variability
4. Diversity
5. Sparsity

- Having more than one meaning
- Disambiguation ⇒ context, external knowledge

Type of ambiguities

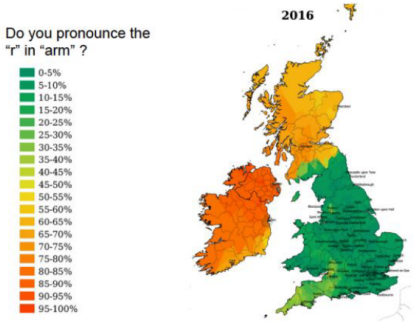
- Semantic / lexical ambiguities: several possible meanings within a single word
  - Polysemy, *e.g.* set , arm, head: Head of New-Zealand is a woman
  - Name Entity: *e.g.* Michael Jordan (professor at Berkeley, basketball player)
  - Object/Color *e.g.* cherry (your cherry coat)
- Syntactic ambiguities: structural/grammatical ambiguity with a sentence



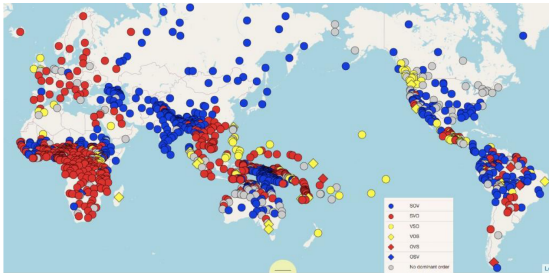


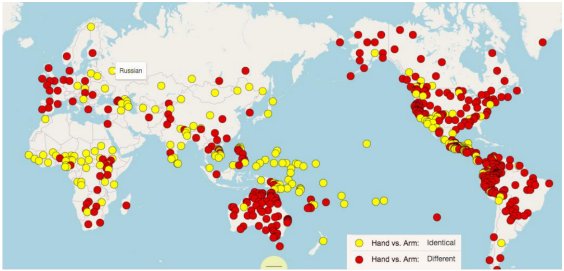
Variations at several levels: phonetic, syntactic, semantic

- Semantic: language variability wrt social context, geography, sociology, date, topic

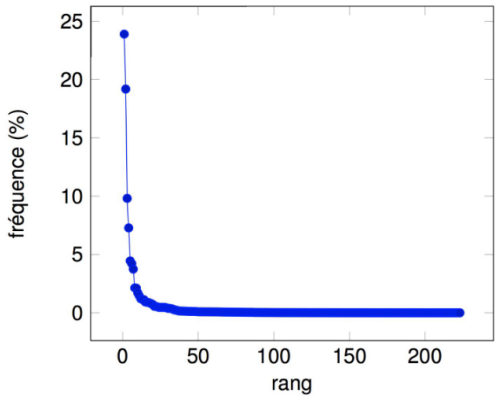


- Syntactic, *e.g.* Subject (S) Verb (V) Object (O) (VOS) order
- Semantic, *e.g.* Words partition: hands vs head





- Word's frequency $\sim$ Zipf law, *i.e.* k^{st} most frequent term: $f_w(k) \propto \frac{1}{k^\theta}$



- <u>Issue:</u> **informative words/info in this distribution?** $\Rightarrow$ more details soon

Why else is natural language understanding difficult?

non-standard English

Great job @justinbieber! Were SOO PROUD of what youve accomplished! U taught us 2 #neversaynever & you yourself should never give up either♥

segmentation issues

the New York-New Haven Railroad

the New York-New Haven Railroad

idioms

dark horse
get cold feet
lose face
throw in the towel

neologisms

unfriend
Retweet
bromance

world knowledge

Mary and Sue are sisters.
Mary and Sue are mothers.

tricky entity names

Where is *A Bug's Life* playing …
Let It Be was recorded …
… a mutation on the *for* gene …

Outline

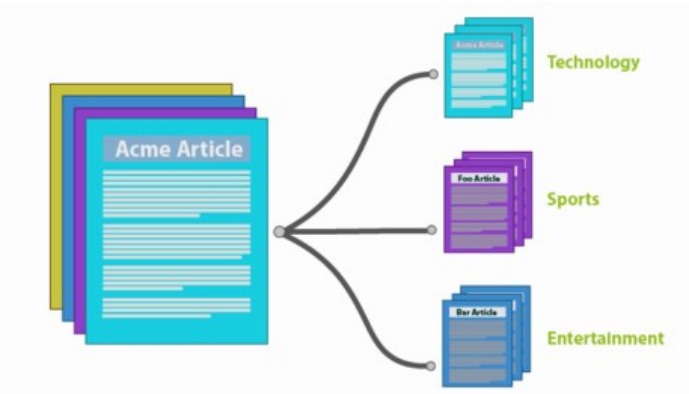
NLP tasks: different levels of analysis

1. At the document level
   - Topic/sentiment classification
   - etc...
2. At the paragraph / sentence level
   - Co-reference resolution
3. At the word level
   - Synonymy
4. At the stream level
   - Topic detection & tracking

$\Rightarrow$ Different levels often refer to different format of data (tabular, sequence, stream)

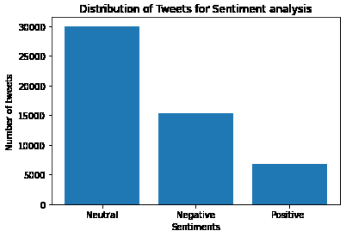
- **Indexing**
  - cf Information Retrieval
- **Classification / filtering**
  - topic
- **Counting / survey**
  - Sentiment classification
- **Clustering** (topic analysis)
  - Unsupervised formulation
  - Large corpus primary exploration
  - Lexical field extraction

- **Indexing**
  - cf Information Retrieval
- **Classification / filtering**
  - topic
- **Counting / survey**
  - Sentiment classification
- **Clustering** (topic analysis)
  - Unsupervised formulation
  - Large corpus primary exploration
  - Lexical field extraction

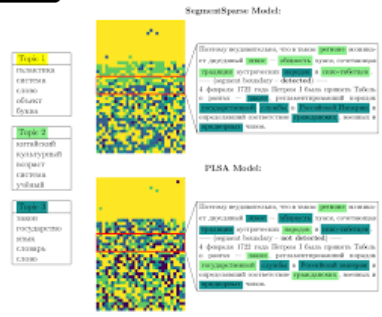


- **Indexing**
  - cf Information Retrieval
- **Classification / filtering**
  - topic
- **Counting / survey**
  - Sentiment classification
- **Clustering** (topic analysis)
  - Unsupervised formulation
  - Large corpus primary exploration
  - Lexical field extraction

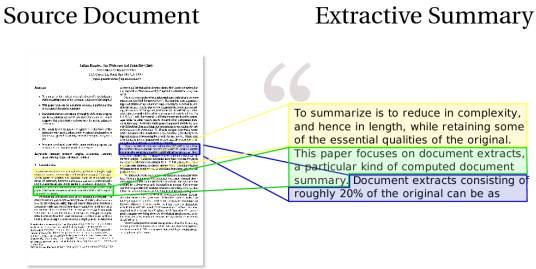
- **Indexing**
  - cf Information Retrieval
- **Classification / filtering**
  - topic
- **Counting / survey**
  - Sentiment classification
- **Clustering** (topic analysis)
  - Unsupervised formulation
  - Large corpus primary exploration
  - Lexical field extraction



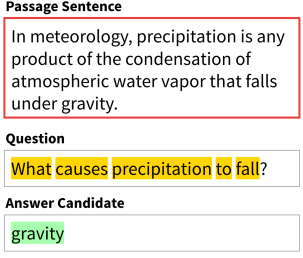
- **Document Segmentation**
  - Sentence classification
  - Link with stream
  - Break detection
- **Automated summary**
  - Sentence extraction
  - Generative architecture
- **Translation**
  - Mostly at the sentence level



- **Document Segmentation**
  - Sentence classification
  - Link with stream
  - Break detection
- **Automated summary**
  - Sentence extraction
  - Generative architecture
- **Translation**
  - Mostly at the sentence level



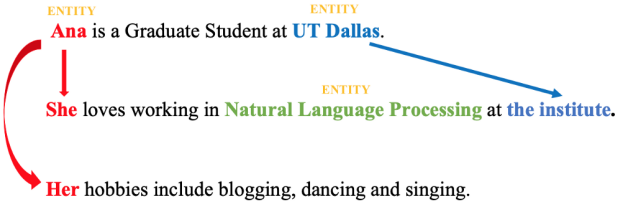
- **Question Answering (QA)**
  - Classification formulation
  - Extraction approach (Siri/Google assistant)
  - An emerging task... For several other tasks!
- **Coreference resolution**
- **Information extraction**
  - Mainly a sentence level task...
  - Sometimes at the paragraph level (with coreference / QA)
- **Dialog State Tracking**
  - Chatbot...



- Between question and answer

cause---gravity
precipitation---gravity
fall---gravity
what---gravity

- **Question Answering (QA)**
  - Classification formulation
  - Extraction approach (Siri/Google assistant)
  - An emerging task... For several other tasks!
- **Coreference resolution**
- **Information extraction**
  - Mainly a sentence level task...
  - Sometimes at the paragraph level (with coreference / QA)
- **Dialog State Tracking**
  - Chatbot...

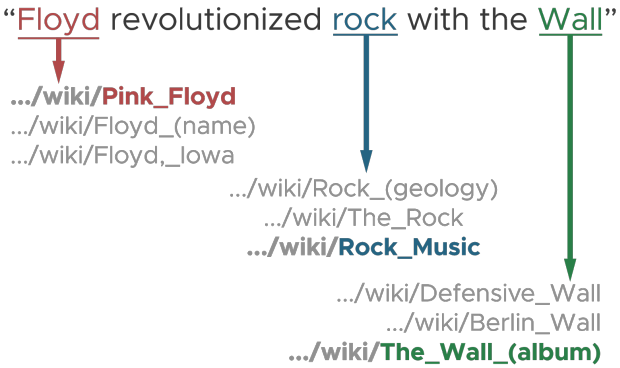


- **Question Answering (QA)**
  - Classification formulation
  - Extraction approach (Siri/Google assistant)
  - An emerging task... For several other tasks!
- **Coreference resolution**
- **Information extraction**
  - Mainly a sentence level task...
  - Sometimes at the paragraph level (with coreference / QA)
- **Dialog State Tracking**
  - Chatbot...

```
Sam walks into the kitchen.
Sam picks up an apple.
Sam walks into the bedroom.
Sam drops the apple.
Q: Where is the apple?
A. Bedroom
```

Note that for each question, a

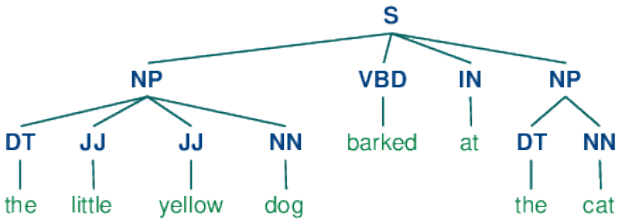
- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**



Word/entity disambiguation

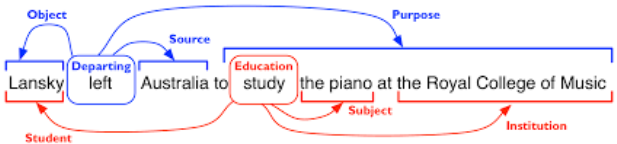
- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**



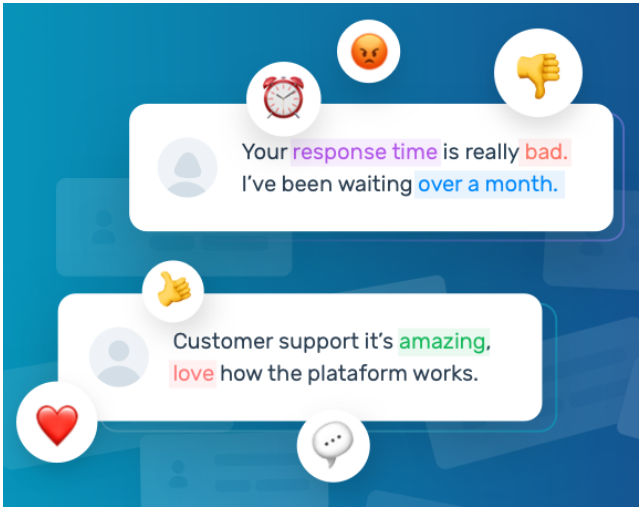
Chunking: Syntactic Parsing

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**



SRL

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**

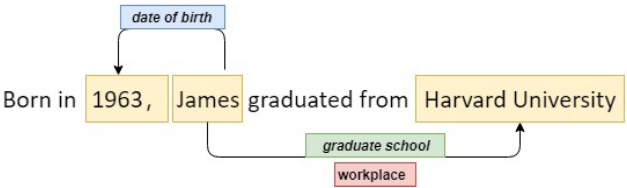


Fine-grained (word-level) sentiment analysis

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**

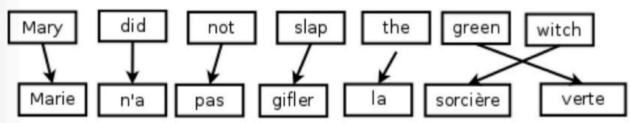
NER

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**



Relation Extraction

- **Word/entity disambiguation**
- **Part-Of-Speech** tagging (POS)
- **Chunking** (sentence segmentation)
- **Semantic Role Labeling** (SRL)
- **Sentiment Analysis** (fine grained)
- **Named Entity Recognition** (NER)
- **Information Extraction**
- **Machine Translation**



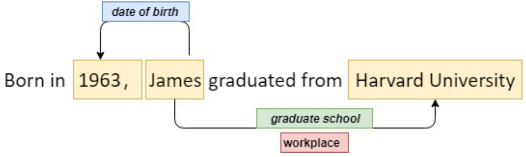
Translation

Text = lot of resources ⇒ Universal knowledge?

- Noisy (both at the syntactic & semantic levels)
- Hard to query & exploit (search, knowledge inference,...)

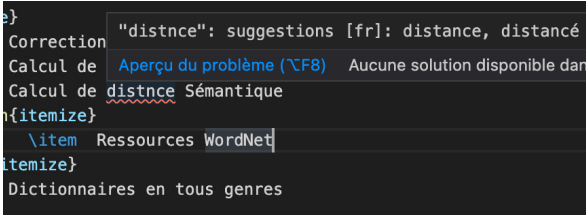
(RDF) knowledge

- Def: Entity 1 (subject) Relation (predicate) Entity 2 (target)
- Simpler version: Key / value
- Easy to handle



At the word level

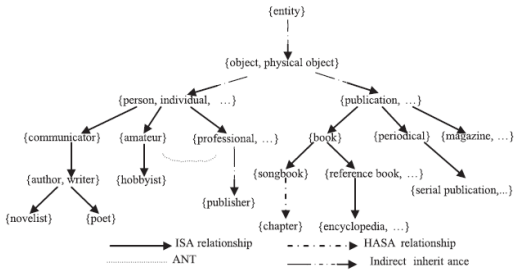
- Syntactic similarities
  - spelling correction
  - Levenshtein = DTW
- Semantic distance (synonymy)
  - WordNet (& other resources)
  - Representation learning



- Syntactic similarities
  - spelling correction
  - Levenshtein = DTW
- Semantic distance (synonymy)
  - WordNet (& other resources)
  - Representation learning

		H	Y	U	N	D	A	I
	0	1	2	3	4	5	6	7
H	1	0	1	2	3	4	5	6
O	2	1	1	2	3	4	5	6
N	3	2	2	2	2	3	4	5
D	4	3	3	3	3	2	3	4
A	5	4	4	4	4	3	2	3

- Syntactic similarities
  - spelling correction
  - Levenshtein = DTW
- Semantic distance (synonymy)
  - WordNet (& other resources)
  - Representation learning



- Historical tasks: **Topic Detection and Tracking (TDT)**
- = adding a temporal axis to the topic detection
  - topic quatification
  - topic appearance
  - topic vanishing

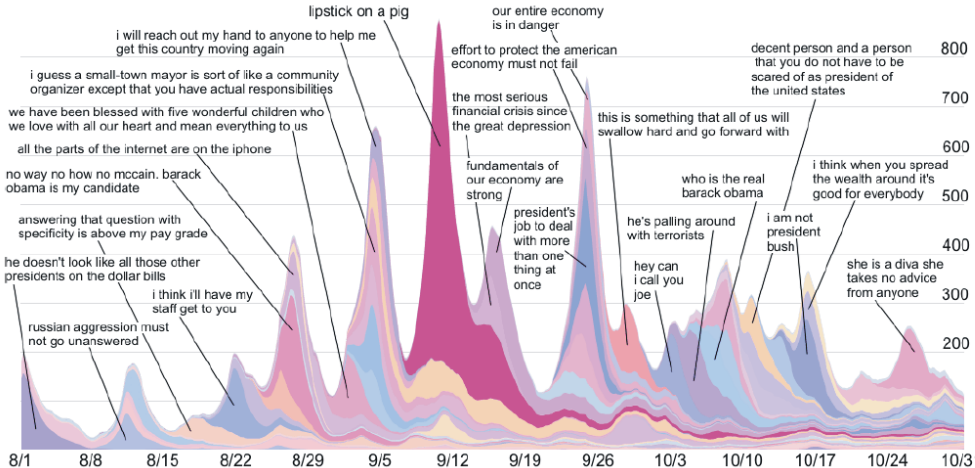


Figure 4: Top 50 threads in the news cycle with highest volume for the period Aug. 1 – Oct. 31, 2008. Each thread consists of all news articles and blog posts containing a textual variant of a particular quoted phrases. (Phrase variants for the two largest threads in each week are shown as labels pointing to the corresponding thread.) The data is drawn as a stacked plot in which the thickness of the strand corresponding to each thread indicates its volume over time. Interactive visualization is available at http://memetracker.org.

Profiling = recommender system

- modern User Interface (UI)
- Information Access
  - Personalization
  - Active suggestion (user = query)

Recent proposal: mixing user interaction & textual review

- user interaction = best item similarity
- review = in depth textual description

McAuley & Leskovec 2013

Language Technology

mostly solved

Spam detection

Let's go to Agra!

Buy V1AGRA ...

✓

✗

Part-of-speech (POS) tagging

ADJ	ADJ	NOUN	VERB	ADV
Colorless	green	ideas	sleep	furiously.

Named entity recognition (NER)

PERSON		ORG		LOC
Einstein	met with	UN	officials in	Princeton

making good progress

Sentiment analysis

Best roast chicken in San Francisco!

The waiter ignored us for 20 minutes.

Coreference resolution

Carter told Mubarak he shouldn't run again.

Word sense disambiguation (WSD)

I need new batteries for my ***mouse***.

Parsing

I can see Alcatraz from the window!

Machine translation (MT)

第13届上海国际电影节开幕...

The 13th Shanghai International Film Festival...

Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30

Party
May 27
add

still really hard

Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

Summarization

The Dow Jones is up

The S&P500 jumped

Housing prices rose

→ Economy is good

Dialog

Where is Citizen Kane playing in SF?

Castro Theatre at 7:30. Do you want a ticket?

Bag of Words (BOW) for document classification

Outline

Think as a data-scientist:

- No magic !
- For each application :
  1. Identify the problem class (regression, classification, ...)
  2. Find global Input / output
  3. Think about the data format (I/O)
  4. Find a model to deal with this data
  5. **Optimize all parameters & hyper-parameters**
     - **Conduct an experiment campaign**

$$\{(x_i, y_i)\}_{i=1,\dots,N} \quad \Rightarrow \text{ learn: } \quad f_{\theta,\phi}(x) \approx y$$

Let's have a tour on NLP applications !

1. Preprocessing

- encoding (latin, utf8, ...)
- ponctuation
- stemming
- lemmatization
- tokenization
- capitals/lower case
- regex
- ...

2. Formatting

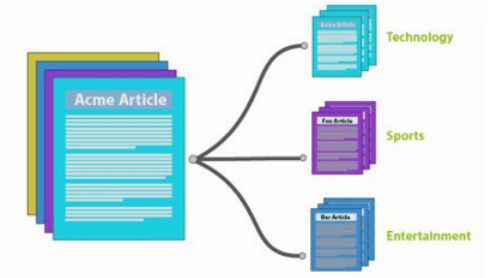
- Dictionary
- + reversed Index
- Vectorial format
- Sequence format

3. Learning

- Doc / sentence / word classification
- Linear/non-linear classifiers?
- HMM, CRF?
- Deep, transformers?

4. Hyper-parameter optimization

- Assigning a document (or paragraph) to a set of predefined categories: sport, science, medicine, religion, etc
- Classification task



2 tasks in project (practicals):

- Sentiment classification on movie reviews
- Speaker identification (Chirac/Miterrand)

Handling textual data: the classification case

1. Big corpus $\Leftrightarrow$ Huge vocabulary
2. Sentence structure is hard to model
3. Words are polymorphous: singular/plural, masculine/feminine
4. Synonyms: how to deal with?
5. Machine learning + large dimensionality = problems

Handling textual data: the classification case

1. Big corpus ⇔ Huge vocabulary
   Logistic Regression, SVM, Naive Bayes... Boosting, Bagging...
   Distributed & efficient algorithms
2. Sentence structure is hard to model
   Removing the structure...
3. Words are polymorphous: singular/plural, masculine/feminine
   Several approaches... (see below)
4. Synonyms: how to deal with?
   wait for the next course !
5. Machine learning + large dimensionality = problems
   Removing useless words

⇒ **Pre-processing + Bag of Word (BoW) Model + Machine Learning**

1. Preprocessing

- encoding (latin, utf8, ...)
- ponctuation
- stemming
- lemmatization
- tokenization
- capitals/lower case
- regex
- ...

More details next course

2. BoW Model

- Dictionary
- Vectorial format
- TF-IDF
- Binary/non-binary
- N-grams

3. Learning

- Doc / sentence / paragraph classification
- Linear/non-linear classifiers?
- Naive Bayes, logistic regression, SVM

4. Hyper-parameter optimization

What is textual data?

- A **series** of letters

```
t h e _ c a t _ i s ...
```

- A **series** of words

```
the cat is ...
```

- A **set** of words

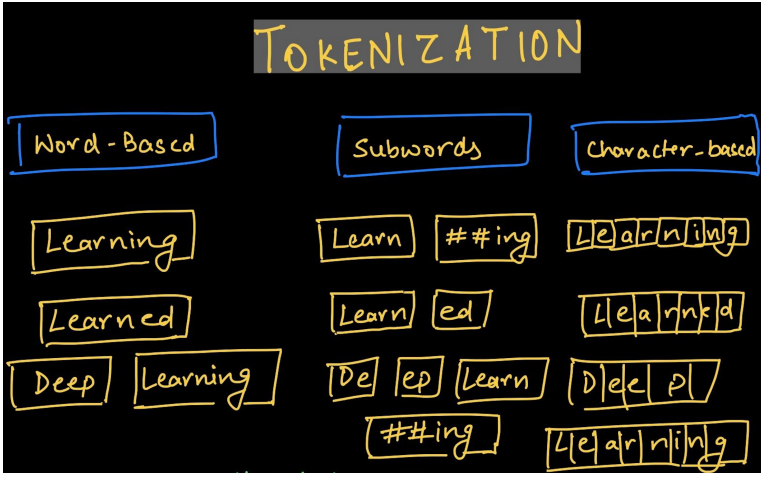
in alphabetical order

```
cat
is
the
...
```

- N-gram dictionary:

```
ex bi-grams: BEG_the the_cat cat_is is_...
```

- Text: extracts "tokens", *i.e.* raw inputs
- Which tokens, *e.g.* characters, words, N-grams, sentences?



- Simplest encoding of tokens: **one-hot representation**
- Binary vector of vocabulary size $|V|$, with 1 corresponding to term index (0 otherwise)
- $|V|$ small for chars (~ 10), large for words ($\sim 10^4$), huge for N-grams / sentences
- Basis for constructing Bag of Word (BoW) Models

the dog is on the table

Sentence structure = costly handling ⇒ Elimination !

Bag of words representation

1. Extraction of vocabulary V
2. Each document becomes a counting vector : $d \in \mathbb{N}^{|V|}$

this new iPhone, what a marvel

An iPhone? What a scam!

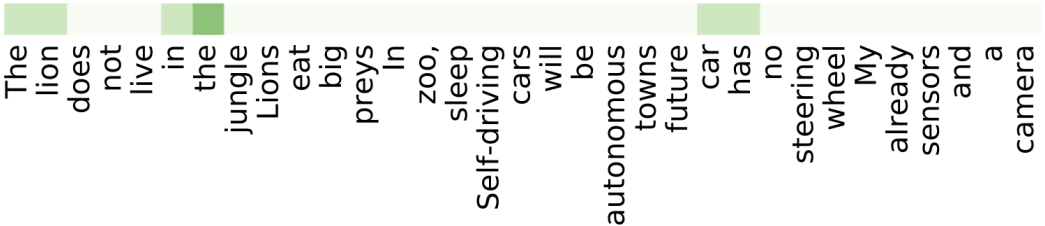
marvel	this	new	iPhone	what	an	scam
1	1	1	1	1	0	0
0	0	0	1	1	1	1

Note: d is always a sparse vectors, mainly composed of 0

Set of toy documents:

```
documents = ['The lion does not live in the jungle',\
'Lions eat big preys',\
'In the zoo, the lion sleep',\
'Self-driving cars will be autonomous in towns',\
'The future car has no steering wheel',\
'My car already has sensors and a camera']
```

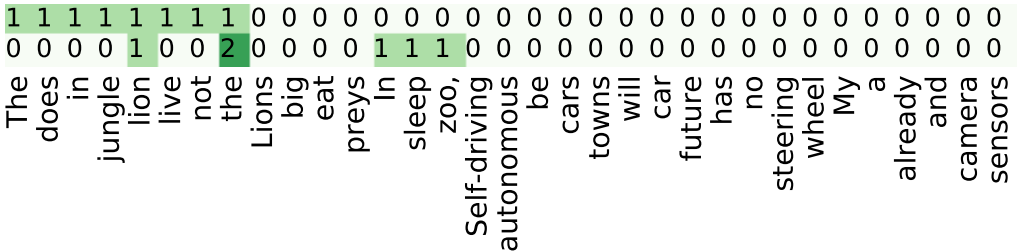
Dictionary



Green level $\propto$ nb occurences

Counting words appearing in 2 documents:

The lion does not live in the jungle
In the zoo, the lion sleep



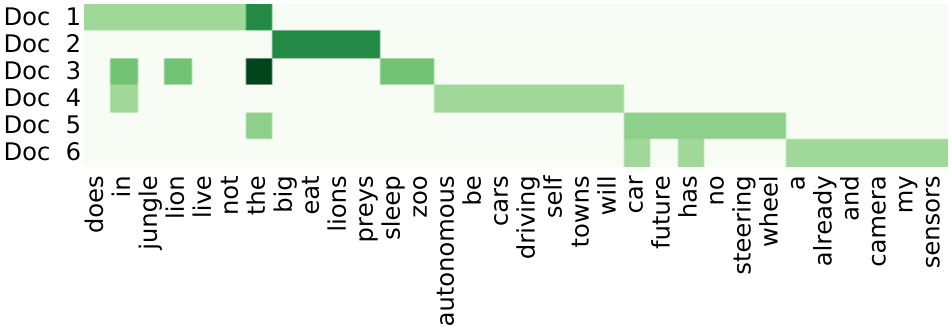
\+ We are able to vectorize textual information

− Dictionary requires preprocessing

Classical issues in text modeling

Every document in the corpus has the **same nearest neighbor**...

A strong magnet with many words (=long document)



Normalization $\Rightarrow$ descriptors = **term frequency** in the document

$$\forall i, \qquad \sum_k d_{ik}^{(tf)} = 1$$

Frequent word are overweighted...

if word k is in most documents, it is probably useless

Introduction of the **document frequency**:

$$\mathrm{df_k} = \frac{|\{\mathrm{d} : t_k \in \mathrm{d}\}|}{|C|}, \qquad \text{Corpus : } C = \{\mathrm{d}_1, \ldots, \mathrm{d}_{|C|}\}$$

Tf-idf coding = term frequency, inverse document frequency:

$$d_{ik}^{(tfidf)} = d_{ik}^{(tf)} \log \frac{|C|}{|\{\mathrm{d} : t_k \in \mathrm{d}\}|}$$

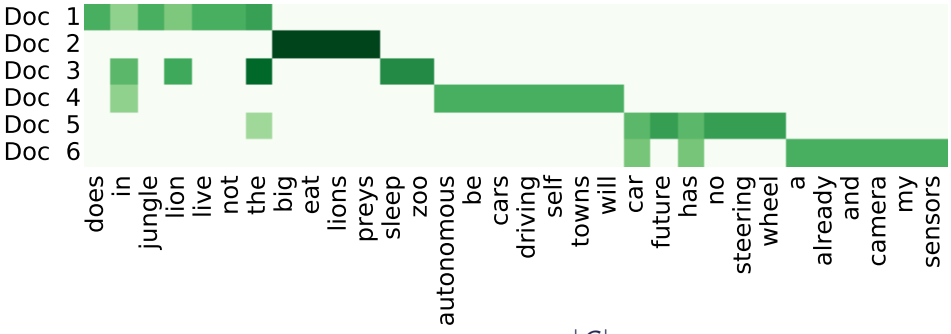
$\Rightarrow$ A strong idea to focus on **keywords**...

... But not always strong enough to get rid of all stop words

$\Rightarrow$ TFIDF could be combined with blacklists (see next course)

Frequent word are overweighted...

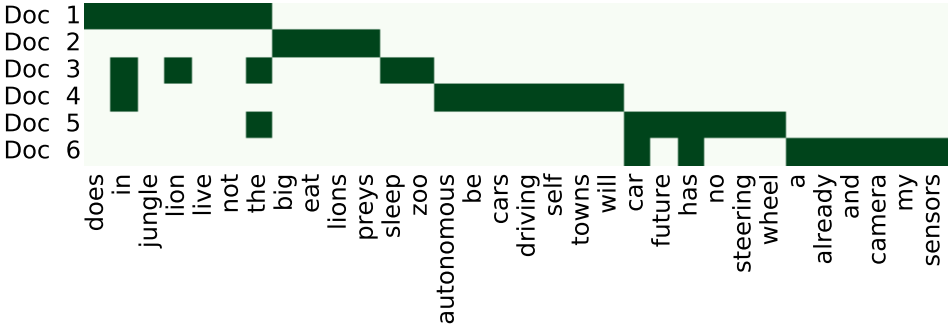
if word k is in most documents, it is not discriminant



$$d_{ik}^{(tfidf)} = d_{ik}^{(tf)} \log \frac{|C|}{|\{\mathrm{d} : t_k \in \mathrm{d}\}|}$$

In some specific cases (e.g. sentiment classification), it is more robust to remove all information about frequency...

$\Rightarrow$ Presence coding



$$d_{ik}^{(pres)} = \begin{cases} 1 & \text{if word k is in d}_i \\ 0 & \text{else} \end{cases}$$

+ Strengths
  - Easy, light, fast (Real-time systems, IR (indexing)...)
  - Opportunity to enrich (POS, context encoding,..)
  - Efficient implementations `nltk, sklearn`
  - Still very effective on document classification see project

− Drawbacks
  - Loose document/sentence structure
    - Can be mitigated with N-grams
  - ⇒ **BUT: several tasks almost impossible to tackle**
    - NER, POS tagging, SRL
    - Text generation

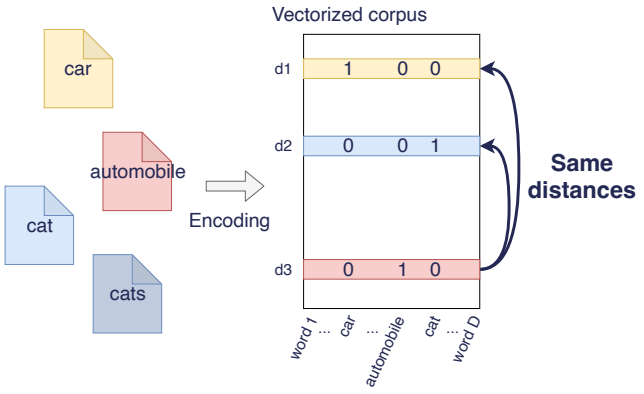
All words are orthogonal:

Considering virtual 2 documents made of a single word :

$$\begin{bmatrix} 0 & \dots & 0 & d_{ik} > 0 & \dots & 0 \\ 0 & \dots & d_{jk'} > 0 & 0 & \dots & 0 \end{bmatrix}$$

Then: $k \neq k' \Rightarrow \mathrm{d}_i \cdot \mathrm{d}_j = 0$

...even if $w_k = \textit{lion}$ and $w_{k'} = \textit{lions}$



⇒ Definition of the **semantic gap**

No metrics between words

BoW extension: beyond unigrams

- Syntactic difference ⇒ orthogonality of the representation vectors
- Word groups : more intrinsic semantics... ... but fewer match with other document
  - N-grams ⇒ dictionary size ↗
  - N-grams = great potential... but require careful preprocessings

This film was not interesting

- Unigrams: `this`, `film`, `was`, `not`, `interesting`
- bigrams: `this_film`, `film_was`, `was_not`, `not_interesting`
- N-grams... + combination: e.g. 1-3 grams

⇒ **See project**

Outline

Exemples de tâches

- Classification thématique
  - classer les news sur un portail d'information,
  - trier des documents pour la veille sur internet,
  - présenter les résultats d'une requête
- Classification d'auteurs
  - review spam,
  - détection d'auteurs
- Information pertinente/non pertinente
  - filtrage personnalisé (à partir d'exemple), classifieur actif (évoluant au fil du temps)
  - spam/non spam
- Classification de sentiments
  - documents positifs/négatifs, sondages en ligne

- Très rapide, interprétable:
  le classifieur *historique* pour les sacs de mots

- Très rapide, interprétable:
  le classifieur *historique* pour les sacs de mots
- Modèle génératif:
  - ensemble des documents $\{d_i\}_{i=1,\dots,N}$,
  - documents = une suite de mots w_j: $d_i = (w_1, \dots, w_{|d_i|})$.
  - modèle Θ_c pour chaque classe de documents.
  - max de vraisemblance pour l'affectation

- Très rapide, interprétable:
  le classifieur *historique* pour les sacs de mots
- Modèle génératif:
  - ensemble des documents $\{d_i\}_{i=1,\dots,N}$,
  - documents = une suite de mots w_j: $d_i = (w_1, \dots, w_{|d_i|})$.
  - modèle Θ_c pour chaque classe de documents.
  - max de vraisemblance pour l'affectation
- Modélisation naive: $P(d_i|\Theta_c) = \prod_{j=1}^{|d_i|} P(w_j|\Theta_c) = \prod_{j=1}^{|D|} P(w_j|\Theta_c)^{x_i^j}$
  x_i^j décrit le nombre d'apparitions du mot j dans le document i

- Très rapide, interprétable:
  le classifieur *historique* pour les sacs de mots
- Modèle génératif:
  - ensemble des documents $\{d_i\}_{i=1,\ldots,N}$,
  - documents = une suite de mots w_j: $d_i = (w_1, \ldots, w_{|d_i|})$.
  - modèle Θ_c pour chaque classe de documents.
  - max de vraisemblance pour l'affectation
- Modélisation naive: $P(d_i|\Theta_c) = \prod_{j=1}^{|d_i|} P(w_j|\Theta_c) = \prod_{j=1}^{|D|} P(w_j|\Theta_c)^{x_i^j}$
  x_i^j décrit le nombre d'apparitions du mot j dans le document i
- Notation: $P(w_j|\Theta_c) \Rightarrow \Theta_c^j$,
  Résolution de: $\Theta_c = \arg\max_\Theta \sum_{i=1}^{|C|} \sum_{j=1}^{|D|} x_i^j \log \Theta_c^j$
  Solution: $\Theta_c^j = \frac{\sum_{d_i \in C} x_i^j}{\sum_{d_i \in C} \sum_{j \in D} x_i^j}$

- Très simple à calculer (possibilité de travailler directement en base de données)
- Naturellement multi-classes,

  inférence: $\arg\max_c \sum_{j=1}^{|D|} x_i^j \log(\Theta_c^j)$

  Performance intéressante... Mais améliorable

- Très simple à calculer (possibilité de travailler directement en base de données)
- Naturellement multi-classes,

  inférence: $\arg\max_c \sum_{j=1}^{|D|} x_i^j \log(\Theta_c^j)$

  Performance intéressante... Mais améliorable
- Extensions:
  - Robustesse : $\Theta_c^j = \frac{\sum_{d_i \in C_m} x_i^j + \alpha}{\sum_{d_i \in C_m} \sum_{j \in D} x_i^j + \alpha |D|}$
  - Mots fréquents (*stopwords)* ... Bcp d'importance dans la décision
  - pas d'aspect discriminant

- Données en sacs de mots, différents codages possibles:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}$$

Un document $x_i \in \mathbb{R}^d$

- Données en sacs de mots, différents codages possibles:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}$$

  Un document $\mathrm{x}_i \in \mathbb{R}^d$
- Décision linéaire:
  - Décision linéaire simple:

$$f(\mathrm{x}_i) = \mathrm{x}_i \mathrm{w} = \sum_j x_{ij} w_j$$

  - Régression logistique:

$$f(\mathrm{x}_i) = \frac{1}{1 + \exp(-(\mathrm{x}_i \mathrm{w} + b))}$$

- Données en sacs de mots, différents codages possibles:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}$$

  Un document $\mathrm{x}_i \in \mathbb{R}^d$
- Décision linéaire:
  - Décision linéaire simple:

$$f(\mathrm{x}_i) = \mathrm{x}_i \mathrm{w} = \sum_j x_{ij} w_j$$

  - Régression logistique:

$$f(\mathrm{x}_i) = \frac{1}{1 + \exp(-(\mathrm{x}_i \mathrm{w} + b))}$$

- Mode de fonctionnement bi-classe (extension par un-contre-tous)

- Formulation
  - Maximisation de la vraisemblance ($y_i \in \{0, 1\}$):

$$L = \Pi_{i=1}^{N} P(y_i = 1|\mathrm{x}_i)^{y_i} \times [1 - P(y_i = 1|\mathrm{x}_i)]^{1-y_i}$$

$$L_{\log} = \sum_{i=1}^{N} y_i \log(f(\mathrm{x}_i)) + (1 - y_i) \log(1 - f(\mathrm{x}_i))$$

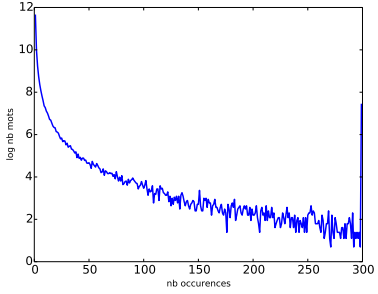
  - Minimisation d'un coût ($y_i \in \{-1, 1\}$):

$$C = \sum_{i=1}^{N} (f(\mathrm{x}_i) - y_i)^2 \qquad \text{ou} \qquad C = \sum_{i=1}^{N} (-y_i f(\mathrm{x}_i))_+$$

- Passage à l'échelle: technique d'optimisation, gradient stochastique, calcul distribué
- Fléau de la dimensionnalité...

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ \vdots & & & \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{bmatrix}$$

- $d = 10^5, N = 10^4 ...$
- Distribution des mots en fonction de leurs fréquences:



- Construire un système pour bien classer tous les documents proposés

Fléau de la dimensionnalité (suite)

- Il est souvent (toujours) possible de trouver des mots qui n'apparaissent que dans l'une des classes de documents...
- Il suffit de se baser dessus pour prendre une décision parfaite...
- Mais ces mots apparaissent ils dans les documents non vus jusqu'ici???

Idée:

Ajouter un terme sur la fonction coût (ou vraisemblance) pour pénaliser le nombre (ou le poids) des coefficients utilisés pour la décision

$$L_{\log} = \sum_{i=1}^{N} y_i \log(f(\mathrm{x}_i)) + (1 - y_i) \log(1 - f(\mathrm{x}_i)) - \lambda \|w\|_\alpha$$

$$C = \sum_{i=1}^{N} (f(\mathrm{x}_i) - y_i)^2 + \lambda \|w\|_\alpha, \quad C = \sum_{i=1}^{N} (-y_i f(\mathrm{x}_i))_+ + \lambda \|w\|_\alpha$$

Avec: $\|w\|_2 = \sum_j w_j^2$ ou $\|w\|_1 = \sum_j |w_j|$

- Etude de la mise à jour dans un algorithme de gradient
- On se focalise sur les coefficients *vraiment* important

Problème

Déséquilibre important entre les populations des classes dans l'ensemble d'entraînement

- Courant en pratique $\Rightarrow$ prédiction privilégiée de la classe dominante
- Ex: classification binaire avec 99% de $\ominus$: classifieur qui prédit toujours la classe dominante $\ominus \Rightarrow$ 99% accuracy
  - Utiliser d'autres métriques, ROC/AUC
- Comment améliorer l'entraînement ?
  - Ré-équilibrer le jeu de données: supprimer des données dans la classe majoritaire et/ou sur-échantilloner la classe minoritaire.
  - Changer la formulation de la fonction de coût pour pénaliser plus les erreurs dans la classe minoritaire. Avec :$\Delta(f(x_i), y_i)$ la fonction de coût:

$$C = \sum_i \alpha_i \Delta(f(x_i), y_i), \qquad \alpha_i = \begin{cases} 1 & \text{si } y_i \in \text{classe majoritaire} \\ B > 1 & \text{si } y_i \in \text{classe minoritaire} \end{cases}$$

Project

Données d'apprentissage:

```
<100:1:C> Quand je dis chers amis, ...
<100:2:C> D'abord merci de cet ...
...
<100:14:M> Et ce sentiment  ...
```

Le format est le suivant: <ID-Discours:ID-phrase:Etiquette>, C $\rightarrow$ Chirac, M $\rightarrow$ Mitterrand

Données de test, sans les étiquettes:

```
<100:1> Quand je dis chers amis, ...
<100:2> D'abord merci de cet ...
...
```

The Task

Building a model for movies revisions in English for classifying it into positive or negative.

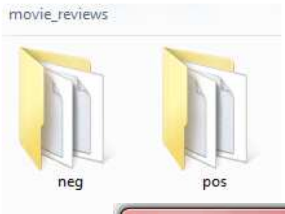
Sentiment Polarity Dataset Version 2.0

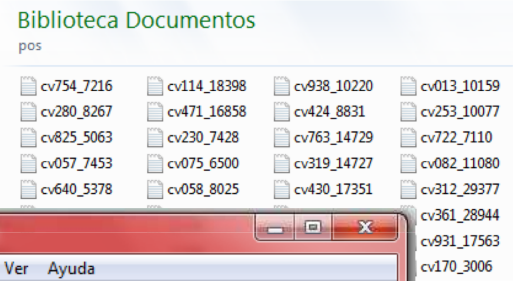
1000 positive movie review and 1000 negative review texts from:

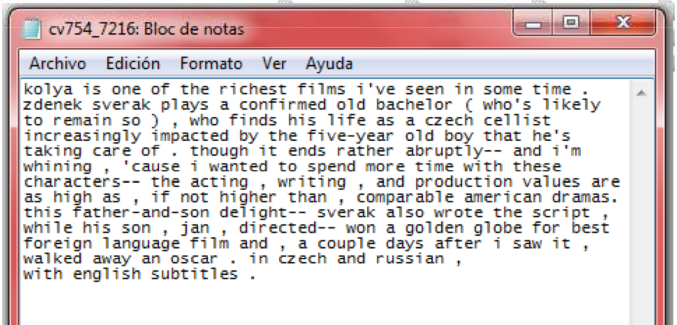
Thumbs up? Sentiment Classification using Machine Learning Techniques. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Proceedings of EMNLP, pp. 79--86, 2002.

"Our data **source** was the **Internet Movie Database** (IMDb) archive of the rec.arts.movies.reviews newsgroup.3 We selected only reviews where the **author rating** was **expressed** either with stars or some **numerical value** (other conventions varied too widely to allow for automatic processing). Ratings were automatically extracted and converted into one of three categories: positive, negative, or neutral. For the work described in this paper, we concentrated **only** on discriminating between **positive** and **negative** sentiment."

The Data (1/2)


movie_reviews
neg
pos


Biblioteca Documentos
pos
cv754_7216
cv280_8267
cv825_5063
cv057_7453
cv640_5378
cv114_18398
cv471_16858
cv230_7428
cv075_6500
cv058_8025
cv938_10220
cv424_8831
cv763_14729
cv319_14727
cv430_17351
cv013_10159
cv253_10077
cv722_7110
cv082_11080
cv312_29377
cv361_28944
cv931_17563
cv170_3006


cv754_7216: Bloc de notas
Archivo Edición Formato Ver Ayuda
kolya is one of the richest films i've seen in some time .
zdenek sverak plays a confirmed old bachelor (who's likely
to remain so) , who finds his life as a czech cellist
increasingly impacted by the five-year old boy that he's
taking care of . though it ends rather abruptly-- and i'm
whining , 'cause i wanted to spend more time with these
characters-- the acting , writing , and production values are
as high as , if not higher than , comparable american dramas.
this father-and-son delight-- sverak also wrote the script ,
while his son , jan , directed-- won a golden globe for best
foreign language film and , a couple days after i saw it ,
walked away an oscar . in czech and russian ,
with english subtitles .

UE TAL :

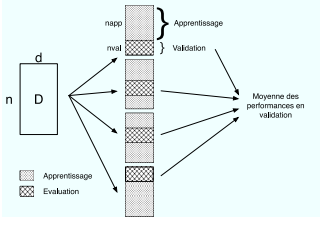
Obligation de participer à une mini-compétition sur les 2 jeux de données

$\Rightarrow$ Buts:

- Traiter des données textuelles (!)
- Travail minimum d'optimisation des classifieurs
- Post-traitements & interactions (minimales) avec un système externe

Outline

- Métriques d'évaluation
  - Taux de reconnaissance $\frac{N_{correct}}{N_{tot}}$
  - Précision (dans la classe c) $\frac{N^c_{correct}}{N^c_{predits}}$
  - Rappel (dans la classe c) (=couverture) $\frac{N^c_{correct}}{N^c_{tot}}$
  - F1 $\frac{(1+\beta^2)precision \cdot rappel}{\beta^2 precision + rappel}$
  - ROC (faux pos *VS* vrai pos) / AUC
- Procédures
  - Apprentissage/test
  - Validation croisée
  - Leave-one-out



Regarder les poids des mots du classifieur:

annoying	37.2593
another	-8.458
any	3.391
anyone	-1.4651
anything	-15.5326
anyway	29.2124
apparently	12.5416
...	
attention	-1.2901
audience	1.7331
audiences	-3.7323
away	-14.9303
awful	30.8509

- nltk
  - Corpus, ressources, listes de *stopwords*
  - quelques classifieurs (mais moins intéressant que sklearn)
- gensim
  - Très bonne implémentation (rapide)
  - Outils pour la sémantique statistique (cours suivants)
- sklearn
  - Boite à outils de machine learning (SVM, Naive Bayes, regression logistique ...)
  - Evaluations diverses
  - Quelques outils pour le texte (simples mais pas très optimisés)

- Lancer des expériences à distance:
  - `nohup` (simple, mais perte du terminal)
    - Redirection, usage des logs
  - `screen`, `tmux`
- Connexion à distance = usage d'une passerelle
  - tunnel ssh
- Gestion des quotas
  - Travail sur le `/tmp`

$\Rightarrow$ Un rythme à trouver: fiabiliser le code en local, lancer les calculs lourds la nuit ou le week-end

Aspects industriels

1. Récupération/importation d'un corpus
   - Lecture de format XML
   - Template NLTK...
2. Optimisation d'un modèle.
   - Campagne d'expérience (d'abord grossière - codage, choix modèle...-, puis fine - régularisation...)
   - Assez long... Mais essentielle
   - **Le savoir-faire est ici**
3. Evaluation des performances (souvent en même temps que la phase d'optimisation)
   - Usage de la validation croisée
4. Apprentissage + packaging du modèle final
   - Définition des formats IO
   - Mode de fonctionnement : API, service web...
   - Documentation

- Montrer que vous êtes capables de réaliser une campagne d'expériences:
  - Courbes de performances
  - Analyse de ces courbes
- Montrer que vous êtes capable de valoriser un modèle
- Concrètement:
  - Mise au propre de votre code
  - Intégration des expériences dans une boucle (ou plusieurs)
  - Analyse qualitative du modèle final (tri des poids)
  - OPT: construction de nuages de mots...