BAG OF WORDS (BOW)

Agro–IODAA, semestre 1

Vincent Guigue

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 $\Leftrightarrow$ Huge vocabulary
   Perceptron, 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

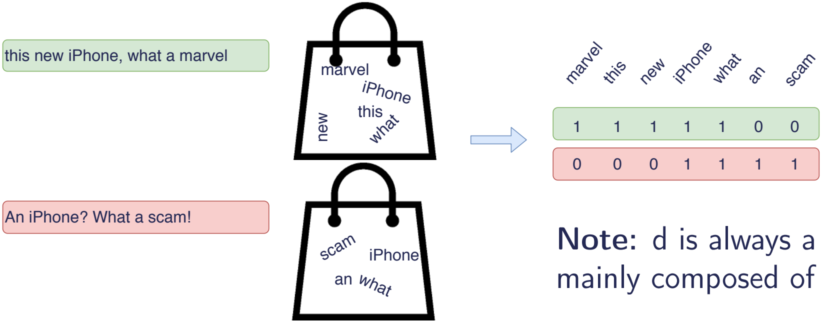
Bag of Words

Bag of words

Sentence structure = costly handling $\Rightarrow$ Elimination !

Bag of words representation

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



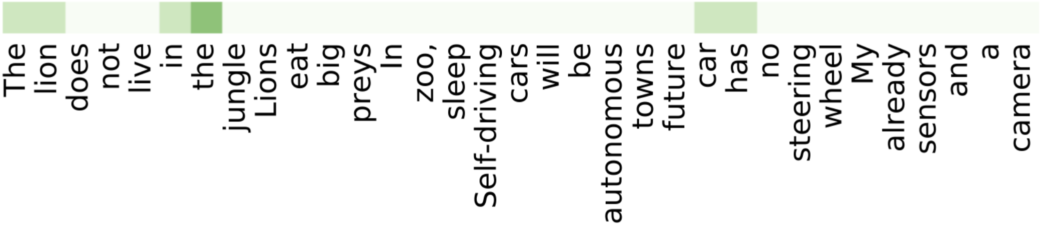
Note: d is always a sparse vectors, mainly composed of 0

Example

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

Information coding

Counting words appearing in 2 documents:

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

The	does	in	jungle	lion	live	not	the	Lions	big	eat	preys	In	sleep	zoo,	Self-driving	autonomous	be	cars	towns	will	car	future	has	no	steering	wheel	My	a	already	and	camera	sensors
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	2	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

+ We are able to vectorize textual information
– Dictionary requires preprocessing

Word representation & semantic gap

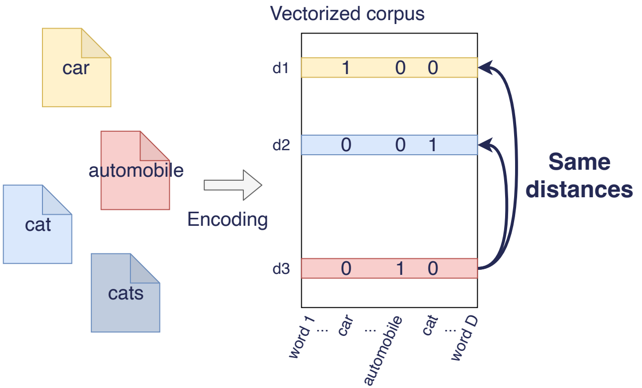
All words are orthogonal:

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 $w_k = lion$ and $w_{k'} = lions$



⇒ Definition of the **semantic gap**

No metrics between words

Semantic issue

Understanding documents = matching relevant descriptors

- Syntactic difference $\Rightarrow$ orthogonality of the representation vectors
- Word groups : more intrinsic semantics...
  ... but fewer match with other document
  - N-grams $\Rightarrow$ dictionary size $\nearrow$
  - 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

Text format

- Bag-of-Words, BoW

\+ Advantages

  - Easy, light
  - fast (Real-time systems, IR (indexing)...)
  - Opportunity to enrich (POS, context encoding, N-gram...)
  - Efficient on document classification
  - Efficient implementations `nltk`, `sklearn`

– Drawbacks

  - Loose document/sentence structure

  ⇒ **Several tasks almost impossible to tackle**

    - NER, POS tagging, SRL
    - Text generation

Dimensionality of the bag of words

Vocabulary size & word occurences

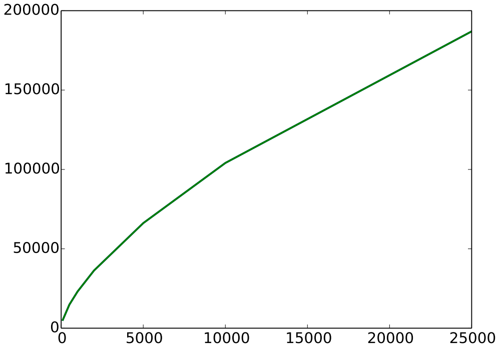
Vocabulary size:

$$|V| \propto \log(N), \qquad N = \text{number of documents}$$

On movie reviews :

$|V|$ with respect to # reviews
Let's have a closer look on the axes !!!

25k docs $\Leftrightarrow$ 200k words !!



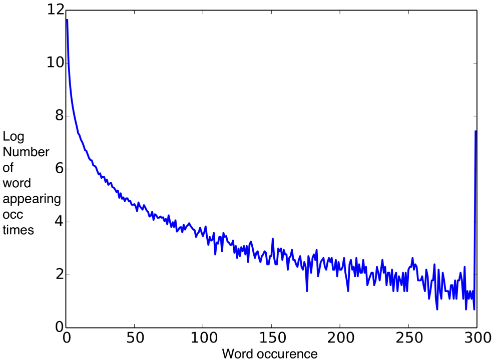
Vocabulary size & word occurences

Vocabulary size:

$$|V| \propto \log(N), \qquad N = \text{number of documents}$$

Word occurence distribution:

$$Occ_i = \{w | \; occurence(w) = i\}$$
$$\text{Plot} = \log(|Occ_i|) \text{ wrt } i$$



Vocabulary size & word occurences

Vocabulary size:

$$|V| \propto \log(N), \qquad N = \text{number of documents}$$

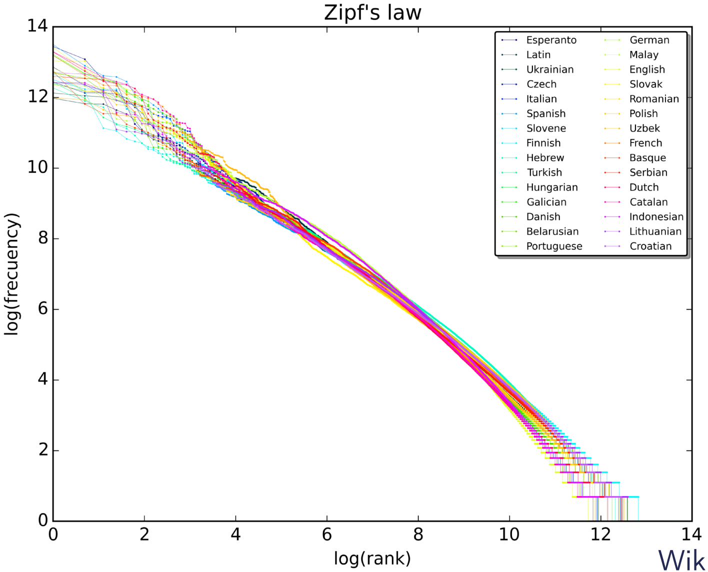
Corpus size:

Which weight for a $25k \times 200k$ matrix of double?

$$= 5 \cdot 10^9 \times 4\text{bytes}$$

$\Rightarrow$ Need a specific data structure: Sparse Matrix

Zipf law in several languages

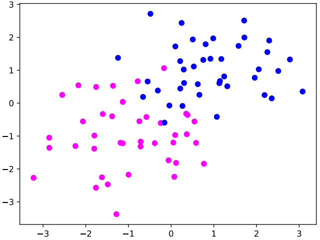


Wikipedia statistics

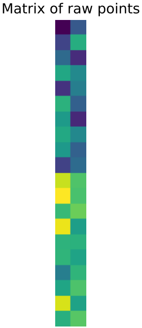
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality:

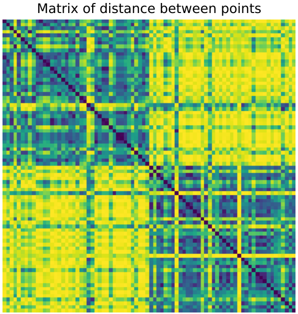
Original dataset:



Matrix view



Distance matrix

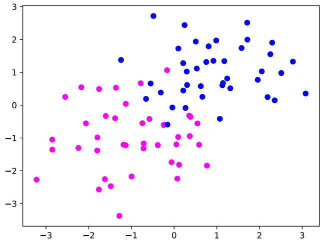


Easy problem / classes are clearly separated

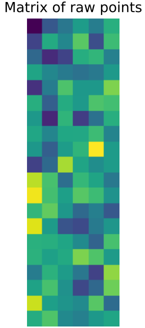
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality:

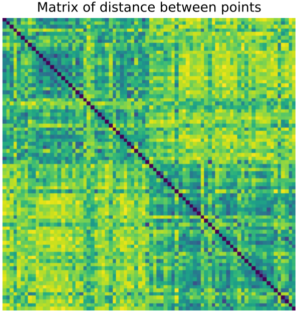
Original dataset:



Matrix view



Distance matrix

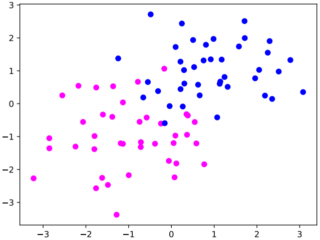


Adding some noisy dimensions in the dataset

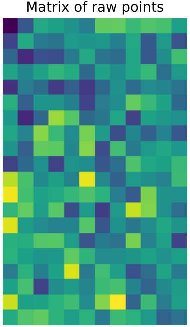
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality:

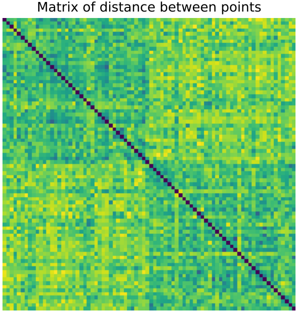
Original dataset:



Matrix view



Distance matrix



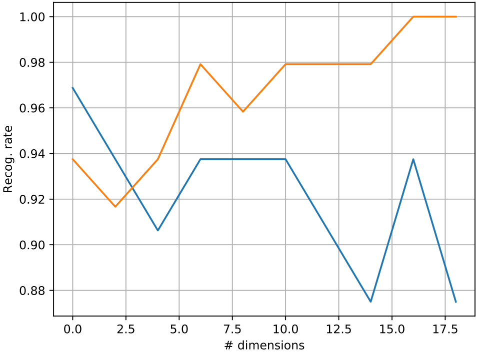
Adding **more** noisy dimensions in the dataset

$\Rightarrow$ Euclidian distance is very sensitive to the dimensionality issue

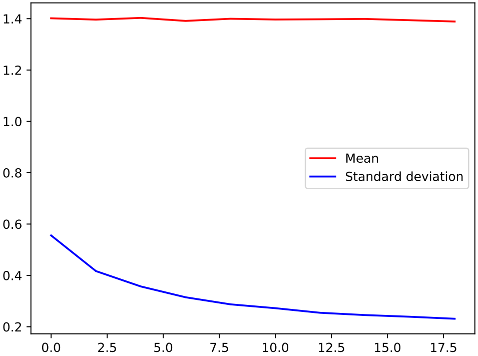
Curse of dimensionality

A classical toy example to illustrate the curse of dimensionality:

Basic classifier on those datasets



Distances between points in the dataset



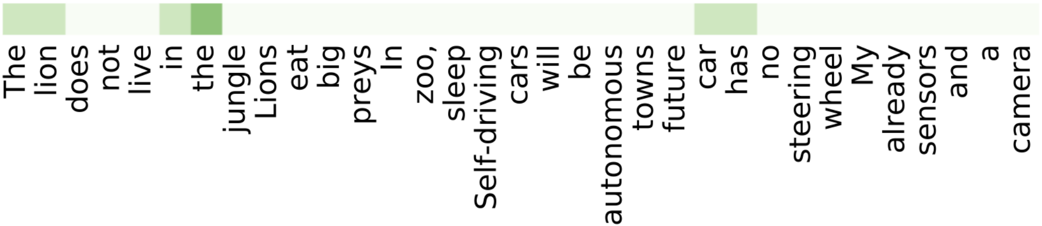
$\Rightarrow$ Learn accuracy $\nearrow$, test accuracy $\searrow$ = **overfitting**

$\Rightarrow$ All points tend to lay on an hypersphere (they become equidistant)

Dimensionality (and noise) reduction

```
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']
```

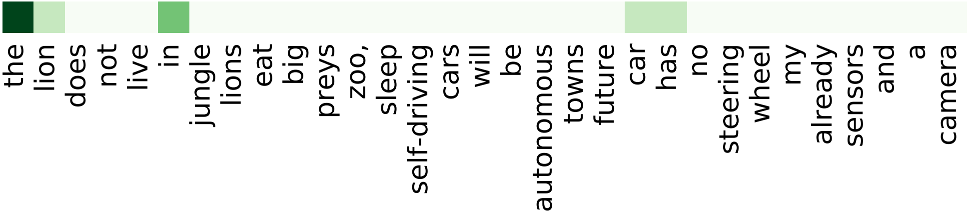
Original dictionary:



Dimensionality (and noise) reduction

```
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']
```

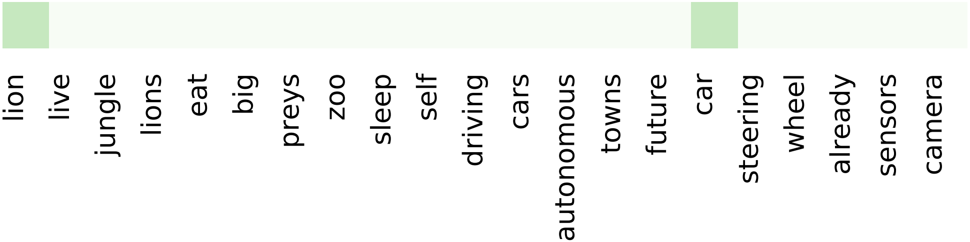
Removing capitals:



Dimensionality (and noise) reduction

```
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']
```

Removing stop words:



Implementation: black list (nltk) or upper frequency bound

Dimensionality (and noise) reduction

```
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']
```

Removing rare words occurring less than a threshold :

Dictionary = {lion, car}

⇒ too extreme in this toy example...

But a good idea in real situations.

Remainder: rare words represent the a large part of the dictionary

⇒ Tricky setting of the thresholds (upper & lower bounds)

Dimensionality (and noise) reduction

```
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']
```

Lemmatization:

in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form

$\Rightarrow$ Requires advanced linguistic ressources including words and inflected forms.
E.g. : lions $\Rightarrow$ lion; are $\Rightarrow$ be; ...

Dimensionality (and noise) reduction

```
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']
```

Stemming:

A statistical approximated process of the lemmatization

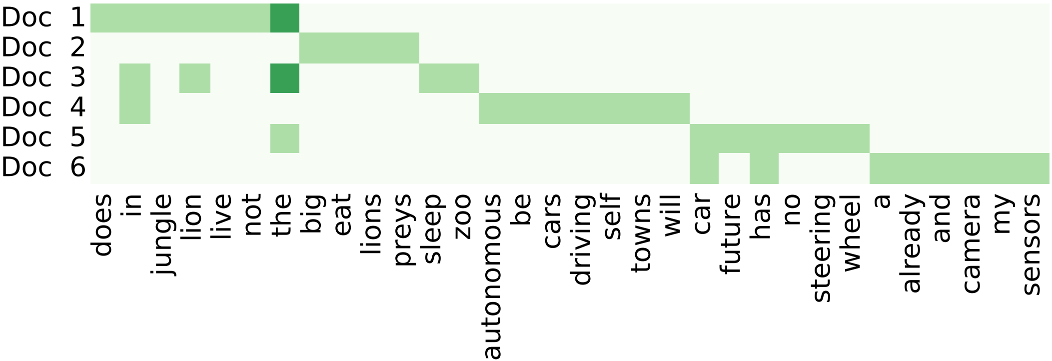
E.g. : removing **s** or **ly** at the end of the words

Note: it is not a problem to create invalid words... If they are stable!

Corpus representation

```python
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']
```

Corpus mapping on the basic dictionary:



Metrics between documents

Given documents $\mathrm{d}_i \in \mathbb{R}^{|D|}$ and $d_j \in \mathbb{R}^{|D|}$ with the dictionary D.

First idea : **Euclidian metrics**

$$d(\mathrm{d}_i, \mathrm{d}_j) = \|\mathrm{d}_i - \mathrm{d}_j\| = \sqrt{\sum_k (d_{ik} - d_{jk})^2}$$

But:

$$d(\mathrm{d}_i, \mathrm{d}_j) = \sqrt{\|\mathrm{d}_i\|^2 + \|\mathrm{d}_j\|^2 - 2\mathrm{d}_i \cdot \mathrm{d}_j}$$

$\Rightarrow$ Sensitive to the norm of d or to the ratio $\mathrm{d}_i \cdot \mathrm{d}_j$ vs $\|\mathrm{d}\|$

Metrics between documents

- **Euclidian distance**
  $\Rightarrow$ not robust enough
- **Inner product**

$$sim(\mathrm{d}_i, \mathrm{d}_j) = \frac{\mathrm{d}_i \cdot \mathrm{d}_j}{\|\mathrm{d}_i\| \|\mathrm{d}_j\|} = \cos(\widehat{\vec{\mathrm{d}}_i, \vec{\mathrm{d}}_j}) \propto \sum_k d_{ik} d_{jk}$$

  $\Rightarrow$ focusing on common non-zeros dimensions
- **Kullback Leibler**
  Assuming that each document can be seen as a distribution over words (ie, $\forall i, \ \sum_k d_{ik} = 1$)

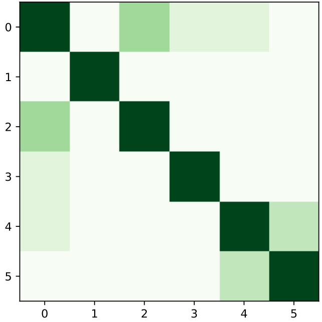
$$D_{\mathrm{KL}}(\mathrm{d}_i \| \mathrm{d}_j) = \sum_k d_{ik} \log \frac{d_{ik}}{d_{jk}}$$

  $\Rightarrow$ not very stable, take care of $d_{ik} = 0$ or $d_{jk} = 0$

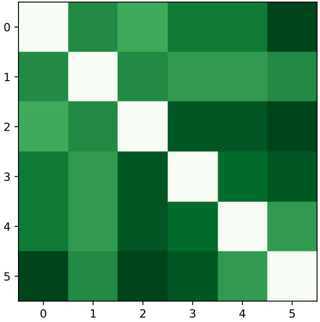
Metrics on our example

```
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']
```

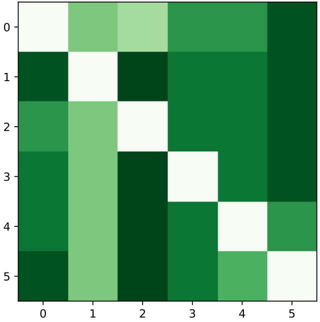
Cos



Euclidian



KL

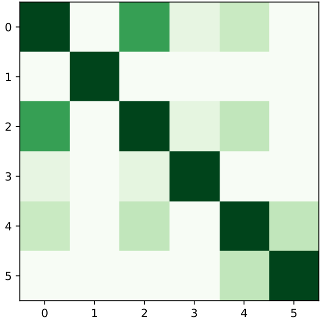


Basic representation of texts... **Too much noise !**

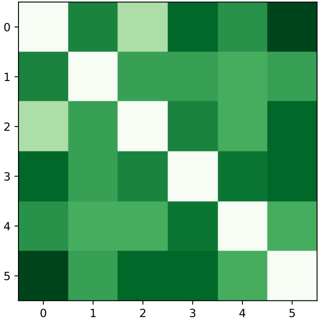
Metrics on our example

```
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']
```

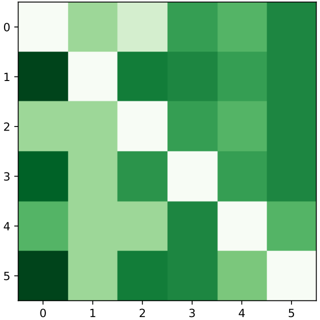
Cos



Euclidian



KL

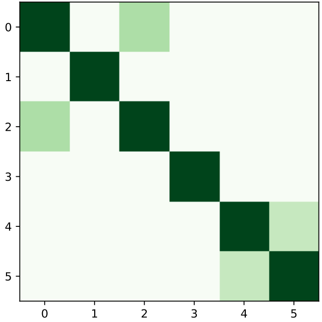


Preprocessing (removing capitals/puntuation) ⇒ situation still confuse

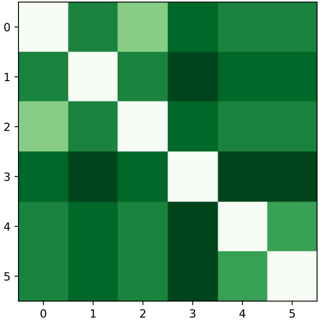
Metrics on our example

```
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']
```

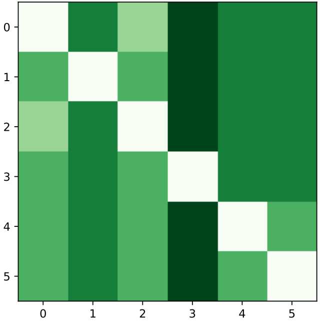
Cos



Euclidian



KL

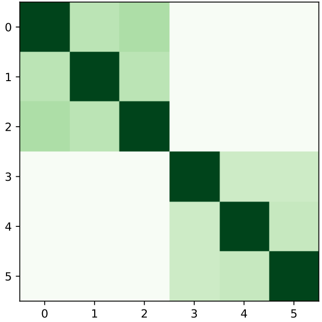


Preprocessing + removing stop words $\Rightarrow$ slight improvment

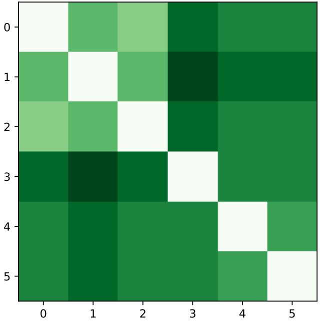
Metrics on our example

```python
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']
```

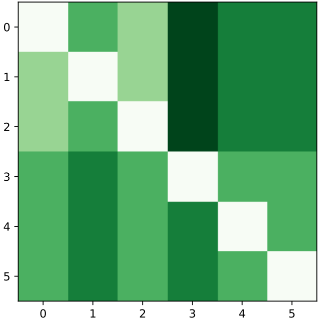
Cos



Euclidian



KL

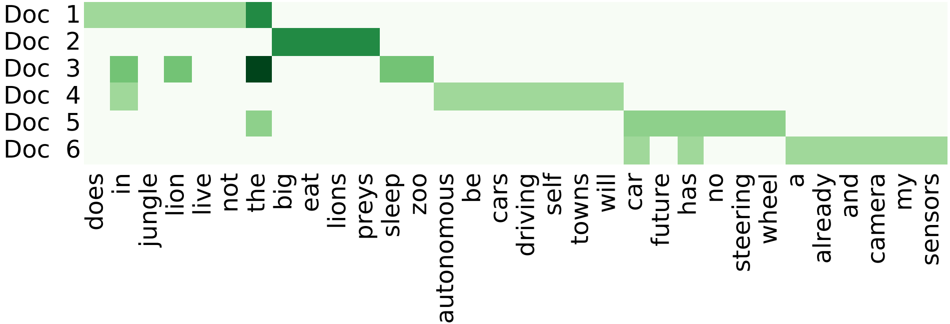


Preprocessing + removing stop words + stemming $\Rightarrow$ distinction between classes

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

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

Classical issues in text modeling

Frequent word are overweighted...

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

Introduction of the **document frequency**:

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

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

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

⇒ A strong idea to focus on **keywords**...

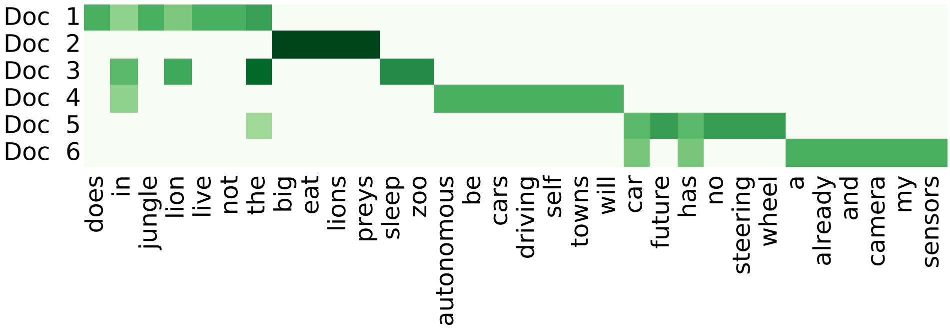
... But not strong enough to get rid of all stop words

⇒ TFIDF should be combined with blacklists

Classical issues in text modeling

Frequent word are overweighted...

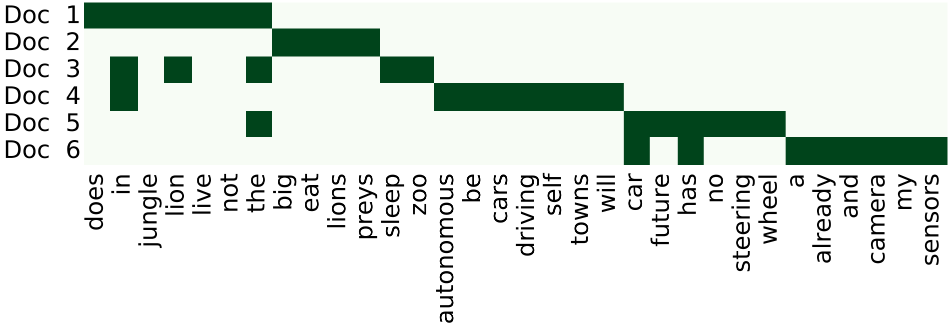
if word k is in most documents, it is not discriminant



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

Classical issues in text modeling

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



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

A small digression onto Information Retrieval

IR main task :

Answering a query q $= \{q_1, \ldots, q_n\}$ by selecting documents d according to metrics : $dist(\mathrm{q}, \mathrm{d})$

Most common metrics: BM25

$$\mathrm{score}(\mathrm{q},\mathrm{d}) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{freq(q_i,\mathrm{d}) \cdot (k_1+1)}{freq(q_i,\mathrm{d}) + k_1 \cdot \left(1 - b + b \cdot \frac{|\mathrm{d}|}{\mathrm{avgdl}}\right)}$$

$$\mathrm{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}, \qquad b = 0.75, k_1 \in [1.2,\ 2.0]$$

N : Corpus size, $n(q_i)$: Number of documents with q_i

IR subtasks:

- Enforcing senrendipity
- Modeling source authority (PageRank)

Dimensionality reduction as a learning problem

- Eliminating word according to a criterion (still preprocessing)
  - Saliency : $S_{tf-idf}(i) = \frac{\sum_j \mathrm{tf-idf}(i,j)}{|\{\mathrm{tf-idf}(i,j) \neq 0\}|}$ (word i, word j)
  - Odds ratio: $S_{odds}(i) = \frac{p_i/(1-p_i)}{q_i/(1-q_i)} = \frac{p_i(1-q_i)}{q_i(1-p_i)}$. (often in log). Where p_i is the frequency of t_i in class 1 and q_i is the frequency of t_i in class 2.
  - Other criteria :Fisher, Mallows... Based on **separability**
- Regularization (improving robustness in learning)
  - cf after

Document classification

Avant les traitements, l'encodage...

http://sametmax.com/lencoding-en-python-une-bonne-fois-pour-toute/

- Sur le disque, les fichiers sont encodés de manière spécifique...
- En python 2, les strings sont encodées de manière spécifique...

Avant les traitements, l'encodage...

`http://sametmax.com/lencoding-en-python-une-bonne-fois-pour-toute/`

- Sur le disque, les fichiers sont encodés de manière spécifique...
- En python 2, les strings sont encodées de manière spécifique...
- L'ouverture des fichiers est souvent associée à un encodage !!!

$\Rightarrow$ Comment gérer cela?

Avant les traitements, l'encodage...

`http://sametmax.com/lencoding-en-python-une-bonne-fois-pour-toute/`

- Sur le disque, les fichiers sont encodés de manière spécifique...
- En python 2, les strings sont encodées de manière spécifique...
- L'ouverture des fichiers est souvent associée à un encodage !!!

⇒ Comment gérer cela?

Solution 1

1. Ouverture en binaire des fichiers (e.g. en python)
2. Conversion des strings depuis un encodage connu
   `str.decode('utf8')`
   `unicodedata`, `unidecode`

Solution 2

1. Vérifier le type d'encodage + convertir avant

Chaine de traitements standard

1. Preprocessing

- encodage (latin, utf8, ...)
- ponctuation
- stemming
- lemmatisation
- tokenization
- minuscule/maj
- regex
- ...

2. Mise en forme

- Construction d'un dictionnaire (index)
- + Index inversé (pour l'explication des traitements
- Mise en forme vectorielle
- Conservation des séquences

3. Traitements

- Classification des docs, des mots, des phrases
- Sémantique

...
Perceptron ou HMM?

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

Naive Bayes

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

- Solution: $\Theta_c^j = \frac{\sum_{d_i \in C} x_i^j}{\sum_{d_i \in C} \sum_{j \in D} x_i^j}$

Naive Bayes

- 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

- Solution: $\Theta_c^j = \frac{\sum_{d_i \in C} x_i^j}{\sum_{d_i \in C} \sum_{j \in D} x_i^j}$

Naive Bayes

- 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

- Solution: $\Theta_c^j = \frac{\sum_{d_i \in C} x_i^j}{\sum_{d_i \in C} \sum_{j \in D} x_i^j}$

Naive Bayes

- Très rapide, interprétable:
  le classifieur *historique* pour les sacs de mots
- Modèle génératif:
  - ensemble des documents $\{d_i\}_{i=1,...,N}$,
  - 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
- 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: $\Theta_c^j = \frac{\sum_{d_i \in C} x_i^j}{\sum_{d_i \in C} \sum_{j \in D} x_i^j}$

Naive Bayes (suite)

- 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

  Quid des mots inconnus dans une classe?

Naive Bayes (suite)

- 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

  Quid des mots inconnus dans une classe?
- 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

Classifieur linéaire, mode de décision

- 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 $\mathrm{x}_i \in \mathbb{R}^d$

Classifieur linéaire, mode de décision

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

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

  - Régression logistique:

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

Classifieur linéaire, mode de décision

- 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)

Formulations et contraintes

- 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 ($y_i \in \{-1, 1\}$):

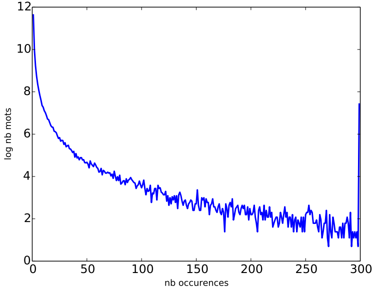
$$C = \sum_{i=1}^{N} (f(\mathrm{x}_i) - y_i)^2 \qquad \text{ou} \qquad C = \sum_{i=1}^{N} (-y_i f(\mathrm{x}_i))_+$$

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

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???

Régularisation

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

Données (exemples)

Analyse de twitter (& réseaux sociaux en général)

- Sur un mot clés:
  - Quantification des documents positifs/négatifs
- Groupe politiques
- Emergence de thématiques...

Locuteurs, Chirac/Mitterrand

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 ...
...
```

Revues de films

The Task

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

Revues de films

Sentiment Polarity Dataset Version 2.0

1000 positive movie review and 1000 negative review texts from:

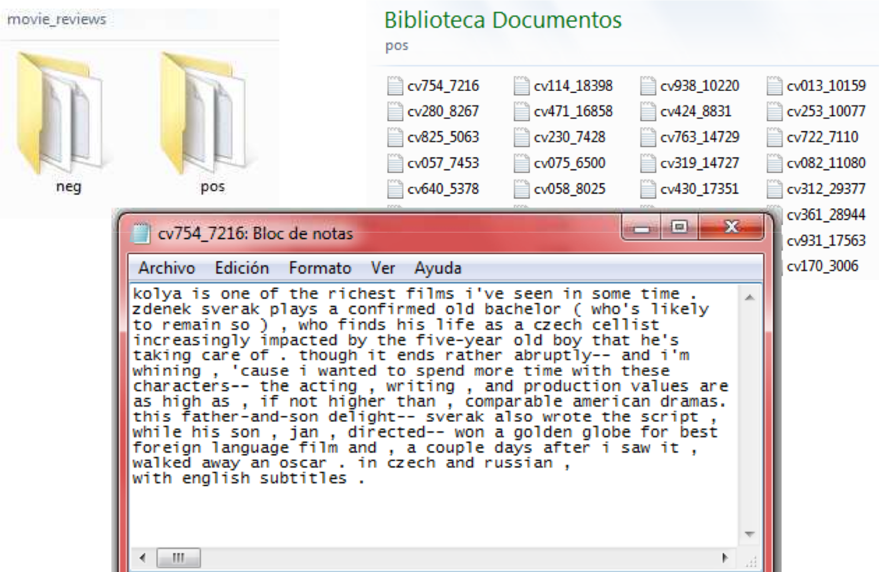
Thumbs up? Sentiment Classification using Machine Learning Techniques. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Proceedings of EMNLP, pp. 79--86, 2002.

"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."

http://www.cs.cornell.edu/people/pabo/movie-review-data/

Revues de films

The Data (1/2)



Compétition

UE TAL :

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

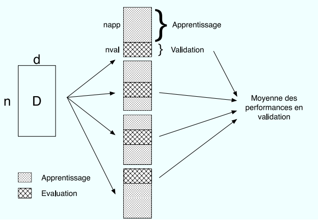
⇒ Buts:

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

Evaluation/outils

Comment évaluer les performances?

- 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

Analyse qualitative

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

Ressources (python)

- 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)

Ressources systèmes

- 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`

⇒ 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