UNIVERSITÉ
CAEN
NORMANDIE

Newspaper classification by date of publication

OUTIN Louis

louis.outin@gmail.com

November 14th, 2018

Summary

Introduction

- Goal : find the publication date of a newspaper article.
- Use of machine learning methods with a training and testing database.
- Evaluate the accuracy of the maximal repeated strings algorithm in this context.
- Evaluate the different classifications algorithms for this problem.
- Result comparing.

Corpus

Newspaper Corpus

- 3600 articles for learning base
- 2450 articles for testing
- 25 documents per year for learning
- 17 documents per year for testing
- classed by exact year (from 1800 to 1950) -> over a 150 year period
- 7 differents newspaper sources

- Database construction for supervised classification
- In newspaper article case, OCR (Optical Character Recognition).

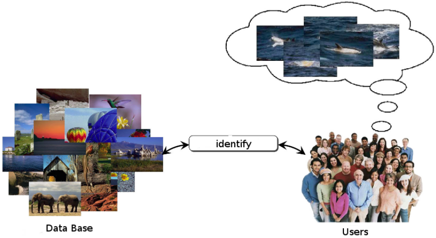


FIGURE – Creating a learning database

Texts format


```xml
<portion id="1">
 <meta>
  <journal>Le Journal des Débats politiques et littéraires</journal>
  <date annee="1848" mois="01" jour="06" />
  <page>2</page>
 </meta>
 <texte>
  a déclaré depuis, dans un document ofHclei, que, sans l'appui que t Europe lui avait
  prêté, elle n'aurait jamais pu surmonter tes obstacles qu'elle rencontrait dans la
  division des esprits et l'opposition des intérêts. Plusieurs cantons, et notamment ceux
  de Schwytz et d'Unterwalden, inquiets sur le maintien de leur souveraineté cantonale et
  sur la protection de leur foi religieuse, se refusaient à entrer dans la Confédération
  c'est sur la parole des grandes puissances et à leur invitation pressante que ces
  cantons ont cédé. Il y a plus. Pour donner à la Suisse une véritable frontière
  défensive, pour établir entre les cantons une contiguïté qui n'existait pas, les grandes
  puissances lui ont concédé gratuitement des territoires considérables. C'est ainsi que
  le district de Versoix a été détaché de la France pour établir la contiguïté entre le
  canton de Genève et celui de Vaud, et que, par le traité de Turin, les communes de
  Savoie qui bordent le lac Léman. entre le Valais et le territoire de Genève, ont été
  réunies à cette dernière république. D'autres concessions du même genre ont encore eu
  lieu. 1 Enfin, les grandes puissances ont garanti à la Confédération helvétique un état
  de neutralité perpétueite, et placé ainsi a l'abri de toute agression son indépendance
  et son intégrité territoriale. EUes ont été déterminées à ces actes de bienveillance par
  l'espérance d'assurer la tranquillité de l'Europe, en plaçant entre plusieurs monarchies
  du continent un Etat pacifique par destination. C'est ce qui se trouve positivement
  exprimé dans le rapport fait au Congrès de Vienne, le 16 janvier 1815, et inséré au
  dixième protocole des actes de ce Congrès. En présence de pareils précéden:, ces
  puissances ont le droit évident d'examiner si la Confédération dont elles ont entendu
  favoriser ia formation et la durée par tant et de telies concessions,
 </texte>
</portion>
```


FIGURE – Texts XML format

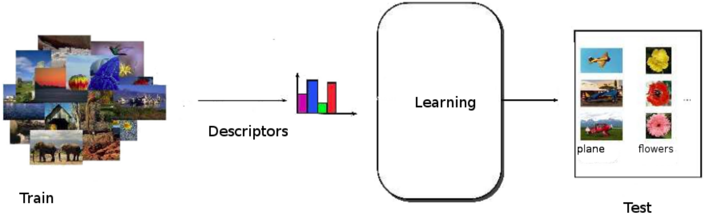
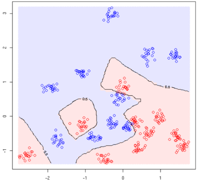
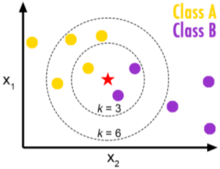


FIGURE – Learning and predicting





Avoid overfittinge

- We train the algorithm with the training datas
- We evaluate the efficiency of the algorithm's parameters over the validation datas.
- Once we found the best classifier and the best parameters, we train it over the training and validation datas and we test it on the test datas.
- Optionnal : K-fold crossvalidation.

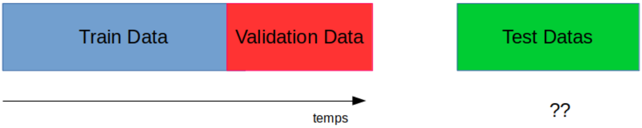


FIGURE – Separating datas

Summary

The maximal repeated strings algorithm :

- **Input** :
  `List of texts` → All the texts from the training
- **Options** :
  `minsup` → Minimum texts occurences
  `maxsup` → Maximum texts occurences

  `minlen` → Minimum length
  `maxlen` → Maximum length

- In texts algorithmic, the **maximal repeated strings** corresponds in data mining to the **frequent** (minimal occurence = 2) **closed sequences** (-> maximal).
- Be careful : maximality here is not to confound with Longest Repeated Substring !
- Used a lot in data mining in general but not that much in natural language processing.
- *Idea* : Can we detect relations between end of one word and the beginning of an other one.

- Thesis Helsinki-Caen, maximal frequent subsequences, *Antoine Doucet 2005*
- Was only word sequences and not characters sequences
- After came the idea to extend it to characters sequences

Algorithm improved by 2 text algorithmicians from Helsinkin, *Juha Karkkainen and Esko Ukonnen*

They improved data structures until having a linear complexity regarding to the input size.

Python implementation by *Romain Brixtel* (Université de Caen)

Publications using this algorithm

- Lejeune and Cartier "Character Based Pattern Mining for Neology Detection" 2017
- Buscaldi and al. "Tweets classification according to the emotion DEFT" 2017
- Lejeune and al. "Highlighting Psychological Features for Predicting Child Interventions During Story Telling" 2016
- Brixtel "Maximal Repeats Enhance Substring-based Authorship Attribution" 2015
- Brixtel and al. "Any Language Early Detection of Epidemic Diseases from Web News Streams" 2013
- Lejeune and al. "Deft 2011 : Matching abstracts and scientific articles based on string distributions"
- Brixtel and al. "Language-Independent Clone Detection Applied to Plagiarism Detection." 2010

Extraction
The algorithm

- Maximal repeated strings algorithm extraction :
  - Take in input : **a list of strings**
  - Return : **a list of lists**
- The idea was to modify the output to have :
  - Every **sub-list** containing as first element **a pattern** which is linked to **a hashmap/dict** as second element.
  - In this hashmap, every key is **the index of a text** which contains this pattern ; and every associated value is the **occurency number** of this pattern in the text.

- With the input :

  "HATTIVATTAATTI" , "ATII ATTA" , "AT"

- minimum repeat : 1 and minimum length : 1

Output of the maximal repeated strings algorithm

```
[
    ['ATT', {0: 3, 1: 1}],
    ['TI', {0: 2, 1: 1}],
    ['I', {0: 2, 1: 2}],
    ['AT', {0: 3, 1: 2, 2: 1}],
    ['A', {0: 4, 1: 3, 2: 1}],
    ['T', {0: 6, 1: 3, 2: 1}],
    ['ATTA', {0: 1, 1: 1}],
    ['ATTI', {0: 2}]
]
```

Extraction
Exemple

- With the input :

  "HATTIVATTAATTI" , "ATII ATTA" , "AT"

- minimum repeat : 2 and minimum length : 2

Output of the maximal repeated strings algorithm

```
[
    ['ATT', {0: 3, 1: 1}],
    ['TI', {0: 2, 1: 1}],
    ['AT', {0: 3, 1: 2, 2: 1}],
    ['ATTA', {0: 1, 1: 1}]
]
```

Summary

Vectorization algorithm

```
For every Text T:
  For every pattern P in the hashmap:
   -if the Text T is in the values of the pattern P:
          -append the occurence number
   -if not:
          -append 0
```

Vectorization algorithm

```
For every Text T:
  For every pattern P in the hashmap:
   -if the Text T is in the values of the pattern P:
         -append the occurence number
   -if not:
         -append 0
```

- Double **for** loop $\rightarrow$ too "complex" over big datas !
- Sparse matrix !

- Use of **pypy** interpreter (mutiply code execution speed by 10) for extracting the data only, (not compatible with scikit-learn).
- We adapt the algorithm's extraction output in order to make it vectorizable by the "Bag of Words" method of **scikit-learn**
- For an input (the same than the previous section) : "HATTIVATTAATTI" , "ATII ATTA" , "AT"

Algorithm result

[' ATT ATT ATT TI TI I I AT AT AT A A A A T T T
T T T ATTA ', ' ATT TI I I AT AT A A A T T T
ATTA ', ' AT A T ']

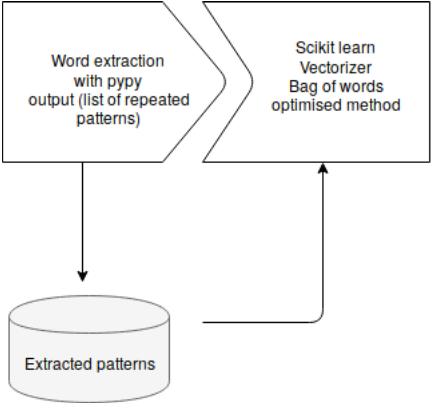


FIGURE – Word extraction and scikit-learn connexion

For vectorization, we use the "bag of words" method of scikit-learn. It allows to extract words occurency of a text in 3 steps :

- **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
- **counting** the occurrences of tokens in each document.
- **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.

Note

- The new patterns for predicting new samples will be ignored. We just use the patterns known by the algorithm during the training.

Summary

Classifiers efficiency evaluation

Separation by decades

- Working over 15 decades → Allows to divide the number of classes by 10
- Score metric : f1-mesure (parameter beta equal to 1), harmonic mean of precision and recall

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

FIGURE – Confusion matrix

- The **precision :** The proportion of well classed documents to a class *i* over all the documents classed into this class *i*.

$$precision_i = \frac{nb\ of\ true\ positive}{nb\ of\ true\ positive + nb\ of\ false\ positive}$$

- The **recall :** The proportion of well classed documents into a class *i* over all the documents belonging to this class *i*.

$$recall_i = \frac{nb\ of\ true\ positive}{nb\ of\ true\ positive + nb\ of\ false\ negative}$$

Note : In multi-class case, the global means of precision and recall over the whole set of classes *i* can be evaluated by the mean of precision and recall over N classes.

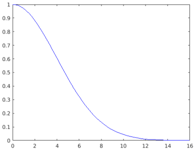
- Boundaries problems
  Exemple : "Year 1919" $\rightarrow$ classed into "Decade 1910"
- Working over years whereas decades multiply by 10 the class numbers
  $\rightarrow$ So we use an area of 15 years around the reference year.
- Scoring metric : the scoring function defined during the DEFT 2011. The system receives for this task a bigger score if the predicted year is close to the reference year (between 0 and 1).

$$S = \frac{1}{\mathrm{N}} \sum_{i=1}^{N} s(d_p(a_i), d_r(a_i))$$

Classifiers efficiency evaluation

Separation by years

The fonction used for computing the similarity between predicted date and reference date is the Gaussian function :



$$s(d_p, d_r) = e^{\frac{-\pi}{10^2}(d_p - d_r)^2}$$

$\|d_p - d_r\|$	0	1	2	3	4	5	6	7	8
$s_g(d_p, d_r)$	1,000	0,969	0,882	0,754	0,605	0,456	0,323	0,215	0,134
$\|d_p - d_r\|$	9	10	11	12	13	14	15	> 15	
$s_g(d_p, d_r)$	0,078	0,043	0,022	0,011	0,005	0,002	0,001	0,000	

TAB. 1 – Valeur du score de similarité s_g selon la distance entre deux années. On peut vérifier que la somme de ces valeurs pour $d_p - d_r$ variant entre -15 et $+15$ est 10.

Summary

Classification algorithms used
SVM

- Support vector machine is a set of supervised learning's methods.
- Their goal is to find the hyperplanes separating the best classes with a maximal marge
- hyperplan : $h(x) = w^T x + w_0$
- Goal : maximize $max(\frac{2}{||w||})$

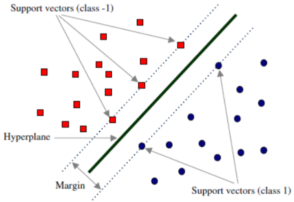


FIGURE – SVM optimal hyperplan and his marge

- Use of a kernel, or mapping function to translate the data into a higher dimensional space.
- The polynomial and RBF are especially useful when the data-points are not linearly separable

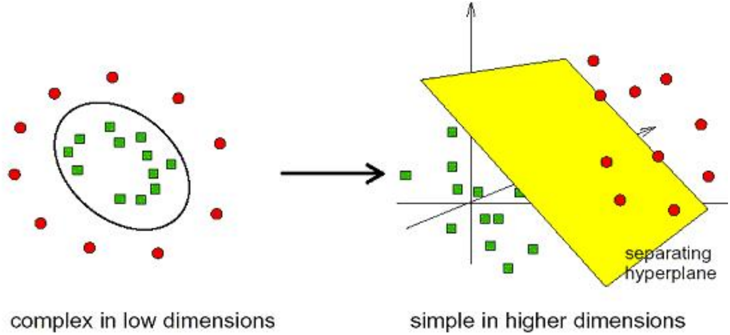


FIGURE – Separation may be easier in higher dimensions

Classification algorithms used

Bayesian Networks

- If we know the probability of each word to belong to a text, knowing that this last one is from a certain year, we can use the Bayesian formula to deduce a probability of a text to belong to a year knowing that a group of words is contained in this text.

$$\underbrace{P(c \mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x \mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

FIGURE – Bayesian formula for classification

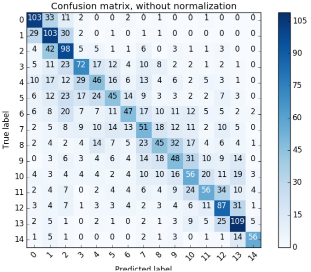
- **ShuffleSplit** from scikit-learn for cross validation
- We split the training set in two sub-set :
  - 70% for trainning
  - 30% for testing
- We iterate over 3 differents splits
- GridMultipleClasifiers for comparing different classifiers with differents parameters sets :
  - Linear SVC, parameters :
    - **C**, (boundaries rigidity) values : [0.1, 0.5, 1, 1.5, 2, 5]
  - Multinomial Naive Bayesien
  - Bernouilli Naive Byesien, parameters :
    - **alpha**, (Additive (Laplace/Lidstone) smoothing parameter) **values** : (0.1, 0.5, 1.0, 2.5)
    - **fit_prior**, (Whether to learn class prior probabilities or not) **values** : True or False
  - Moreover, we repeat it with different length for extracted patterns : 1-3 / 1-7 / 3-7 / 1-1000

- For maximal repeated strings extraction, **Multinomial Naive Bayesien** looks to be the best classifier with the following parameters :
  - **alpha : 0.5**
  - **fit_prior : False**
  - **patterns length : 3-7**
- For "Bag of words" extraction, the **Linear SVC** looks to be the best classifier with the following parameters :
  - **C : 1.5**
  - **patterns length : 3-7**

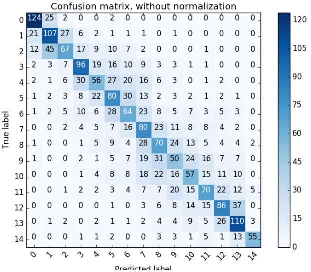
Summary

500 words texts, DEFT 2011 :

- **Maximal repeated strings extraction**
- f-mesure : **0.409**
- pourcentage of decades well predicted : **41.8%**



- **'Bag of Words' extraction**
- f-mesure : **0.477**
- pourcentage of decades well predicted : **47.9%**



500 words texts, DEFT 2011 :

- **Maximal repeated strings extraction**
- mean : **0.327**
- median : **0.011**
- std : **0.415**
- variance : **0.172**
- pourcentage of decades well predicted : **46.9%**

- **'Bag of Words' extraction**
- mean : **0.402**
- median : **0.215**
- std : **0.426**
- variance : **0.181**
- pourcentage of decades well predicted : **57.1%**

Classification in two steps

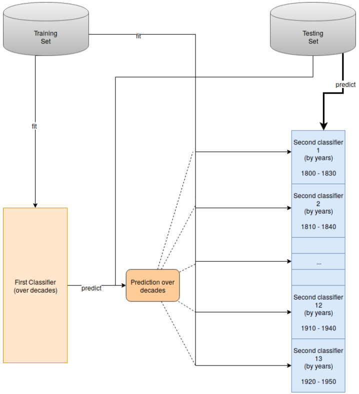


FIGURE – Classification in two steps

- First classification by decades
- Second classification aimed on the chosen decade from first step plus the two adjacent decades
- Allows to eliminate interferences with faraway classes
- Inconvenient : longer to train

500 words texts, DEFT 2011 :

- Maximal repeated strings extraction

- mean : **0.392**
- median : **0.134**
- STD : **0.422**
- variance : **0.178**
- pourcentage of decades well predicted : **57.4%**

- 'Bag of words' extraction

- mean : **0.435**
- median : **0.323**
- STD : **0.422**
- variance : **0.178**
- pourcentage of decades well predicted : **63.1%**

Summary

Results comparing

TABLE – Comparing with DEFT 2011 others results

	Classification in two steps (by maximal repeated strings extraction)	Classification in two steps (by 'bag of words' extraction)	Mean score of participants to DEFT 2011
Mean	0.392	0.435	0.247
Median	0.134	0.323	0.358
STD	0.422	0.422	0.183
Variance	0.178	0.178	0.033
% good decades	57.4	63.1	

Conclusion

- Most efficient algorithm for this task :
  - with maximal repeated strings : **Multinomial Naive Bayesien**
  - with Bag of words : **Linear SVM**
- Better score on years division .
- Better strategy : Classify in two step (fisrt by decades then by years).
- The maximal repeated strings extraction is finally less efficient for this task than the 'bag of words' extraction.
- 63% of texts predicted into the good decade -> complex task.

- Github link :

https://github.com/louisoutin/textClassification

THANK YOU FOR YOUR ATTENTION !
QUESTIONS ?