Newspaper classification by date of publication

OUTIN Louis

louis.outin@gmail.com

November 14th, 2018
Summary

Introduction
- Create database
- Learn and Predict
- Separate datas

Classification algorithms used and comparison

Results

Conclusion

Extraction

Vectorization

Classifiers efficiency evaluation
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.
## 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 different newspaper sources
Database construction for supervised classification
In newspaper article case, OCR (Optical Character Recognition).

**FIGURE** – Creating a learning database
a déclaré depuis, dans un document de Chéron, que, sans l'appui que l'Europe lui avait prêté, elle n'aurait jamais pu surmonter ses 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. Enfin, les grandes puissances ont garanti à la Confédération helvétique un état de neutralité perpétueite, et placé ainsi à 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 précédents, ces puissances ont le droit évident d'examiner si la Confédération dont elles ont entendu favoriser la formation et la durée par tant et de telles concessions,
Machine learning and classification
2: Learn and predict

**FIGURE** – Learning and predicting
Avoid overfitting

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

**FIGURE – Separating data**
Summary

Introduction

Extraction

Vectorization

Classifiers efficiency evaluation

Classification algorithms used and comparison

Results

Conclusion
The maximal repeated strings algorithm:

- **Input**: List of texts → All the texts from the training

- **Options**:
  - minsup → Minimum texts occurrences
  - maxsup → Maximum texts occurrences
  - minlen → Minimum length
  - maxlen → Maximum length
In texts algorithmic, the **maximal repeated strings** corresponds in data mining to the **frequent** (minimal occurrence = 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 another 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
- Lejeune and al. "Deft 2011 : Matching abstracts and scientific articles based on string distributions"
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 occurrence number of this pattern in the text.
With the input:

"HATTIVATTATTI", "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}]
]
```
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

Introduction

Extraction

Vectorization
  Optimization

Classifiers efficiency evaluation

Classification algorithms used and comparison

Results

Conclusion
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 occurrence number
      – if not:
         – append 0

▶ Double for loop → too “complex” over big datas!
▶ Sparse matrix!
Vectorization
First idea

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 occurrence number
      − if not:
          − append 0

▷ Double for loop → too "complex" over big datas!
▷ Sparse matrix!
Use of **pypy** interpreter (multiply 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

```plaintext
[ ' ATT ATT ATT TI TI I I AT AT AT A A A A T T 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 occurrence 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

Introduction

Extraction

Vectorization

Classifiers efficiency evaluation
  Separation by decades
  Separation by years

Classification algorithms used and comparison

Results

Conclusion
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

\[ F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]

**Figure** – Confusion matrix
Classifiers efficiency evaluation
Separation by decades

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

$$\text{precision}_i = \frac{\text{nb of true positive}}{\text{nb of true positive} + \text{nb of false positive}}$$

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

$$\text{recall}_i = \frac{\text{nb of true positive}}{\text{nb of true positive} + \text{nb of false negative}}$$

**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.
Classifiers efficiency evaluation
Separation by years

- Boundaries problems
  **Exemple**: "Year 1919" → classed into "Decade 1910"

- Working over years whereas decades multiply by 10 the class numbers → 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).

\[
S = \frac{1}{N} \sum_{i=1}^{N} s(d_p(a_i), d_r(a_i))
\]
The function used for computing the similarity between predicted date and reference date is the Gaussian function:

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

\[ \begin{array}{cccccccccc}
|d_p - d_r| & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
s_g(d_p, d_r) & 1,000 & 0,969 & 0,882 & 0,754 & 0,605 & 0,456 & 0,323 & 0,215 & 0,134 \\
|d_p - d_r| & 9 & 10 & 11 & 12 & 13 & 14 & 15 & > 15 \\
\hline
s_g(d_p, d_r) & 0,078 & 0,043 & 0,022 & 0,011 & 0,005 & 0,002 & 0,001 & 0,000 \\
\end{array} \]

Table 1: Value of the similarity score \( s_g \) according to the distance between two years. It can be verified that the sum of these values for \( d_p - d_r \) varying between \(-15\) and \(+15\) is 10.
Summary

Introduction

Extraction

Vectorization

Classifiers efficiency evaluation

Classification algorithms used and comparison
  Algorithms
  Classification algorithms used
  Classifiers comparing

Results

Conclusion
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 \( \max \left( \frac{2}{||w||} \right) \)

Figure – SVM optimal hyperplan and his marge
Classification algorithms used
SVM

- 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
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.

\[
P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}
\]

\[
P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)
\]

**Figure** – Bayesian formula for classification
Classifiers comparing Cross Validation and MultipleGridSearch

- **ShuffleSplit** from scikit-learn for cross validation
- We split the training set in two sub-set:
  - 70% for training
  - 30% for testing
- We iterate over 3 different splits

- GridMultipleClassifiers for comparing different classifiers with different parameters sets:
  - Linear SVC, parameters:
    - \( C \), (boundaries rigidity) values: [0.1, 0.5, 1, 1.5, 2, 5]
  - Multinomial Naive Bayesien
  - Bernouilli Naive Bayesien, 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

Introduction

Extraction

Vectorization

Classifiers efficiency evaluation

Classification algorithms used and comparison

Results

Classification in two steps

Conclusion
Results
Decades

500 words texts, DEFT 2011:

- Maximal repeated strings extraction
  - f-measure: 0.409
  - Percentage of decades well predicted: 41.8%

- 'Bag of Words' extraction
  - f-measure: 0.477
  - Percentage of decades well predicted: 47.9%

Confusion matrix, without normalization
## Results

### Years

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Median</th>
<th>Std</th>
<th>Variance</th>
<th>Decades Well Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximal repeated strings</td>
<td>0.327</td>
<td>0.011</td>
<td>0.415</td>
<td>0.172</td>
<td>46.9%</td>
</tr>
<tr>
<td>‘Bag of Words’ extraction</td>
<td>0.402</td>
<td>0.215</td>
<td>0.426</td>
<td>0.181</td>
<td>57.1%</td>
</tr>
</tbody>
</table>

500 words texts, DEFT 2011:
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

**Figure** – Classification in two steps
Classification in two steps
Results

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

Introduction
Extraction
Vectorization
Classifiers efficiency evaluation

Classification algorithms used and comparison

Results

Conclusion
Results comparing

**TABLE – Comparing with DEFT 2011 others results**

<table>
<thead>
<tr>
<th></th>
<th>Classification in two steps (by maximal repeated strings extraction)</th>
<th>Classification in two steps (by ‘bag of words’ extraction)</th>
<th>Mean score of participants to DEFT 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.392</td>
<td>0.435</td>
<td>0.247</td>
</tr>
<tr>
<td>Median</td>
<td>0.134</td>
<td>0.323</td>
<td>0.358</td>
</tr>
<tr>
<td>STD</td>
<td>0.422</td>
<td>0.422</td>
<td>0.183</td>
</tr>
<tr>
<td>Variance</td>
<td>0.178</td>
<td>0.178</td>
<td>0.033</td>
</tr>
<tr>
<td>% good decades</td>
<td>57.4</td>
<td>63.1</td>
<td></td>
</tr>
</tbody>
</table>
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 (first 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?