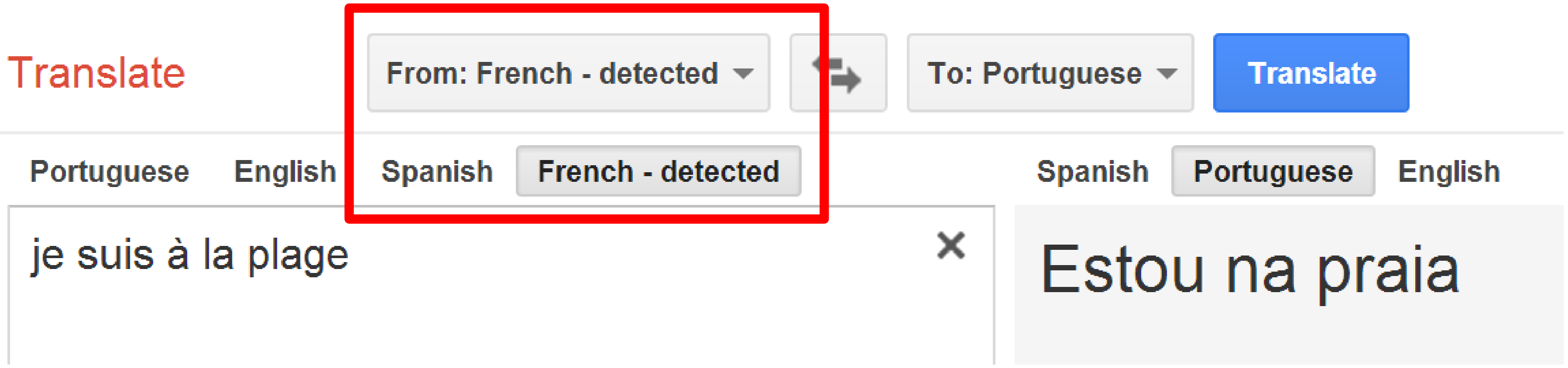
Natural Language Processing using Machine Learning

Miguel Almeida, André Martins, Afonso Mendes

Priberam Labs
http://labs.priberam.com
mba@priberam.com

December 18, 2012

Automatic language detection



From: French - detected

To: Portuguese

je suis à la plage

Estou na praia

Automatic language detection

- One of the easiest NLP problems

- One of the simplest classifiers: Naïve Bayes
  - Also used for spam detection

- Relies on two simple concepts:
  - Bayes Rule
  - Conditional independence

(Bayes rule)

- For any random variables *A* and *B*:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

(conditional independence)

- Independence between variables *A* and *B*:
  - Knowing *A* does not give information about *B* and vice-versa

$$P(A, B) = P(A)P(B)$$

- Conditional independence of *A* and *B*, given *C*:
  - If we know *C*, knowing *A* does not give information about *B* and vice-versa

$$P(A, B|C) = P(A|C)P(B|C)$$

Automatic language detection

- Training data: Wikipedia
  - 3.3 GB of Portuguese text (PT)
  - 5.6 GB of Spanish text (ES)
  - 8.4 GB of French text (FR)

WIKIPEDIA

- Some preprocessing involved
  - Remove XML markup to keep only the text
  - Remove uninformative sections (e.g. references)
  - Transform everything to lowercase

Automatic language detection

- *x* = input (string)
  - Example: x = “eu fui”
- *y* = output (language)
  - y belongs to {PT, ES, FR}
  - Easy to add more languages (use more Wikipedias...)

- **Our goal:** given the string *x*, find the language *y* which is most likely ➔ maximize $P(y|x)$
  - Known as **Maximum A Posteriori (MAP)** estimator

Automatic language detection

- x = string
- y = language

- Goal: maximize $P(y|x)$

$$y^* = \underset{y}{\operatorname{argmax}}\, P(y|x) \overset{\text{Bayes Rule}}{=} \underset{y}{\operatorname{argmax}} \frac{P(x|y)P(y)}{P(x)}$$

$P(x)$ does not depend on y

$$= \underset{y}{\operatorname{argmax}}\; P(x|y)P(y)$$

Automatic language detection

- x = string
- y = language

- Goal: find $y^* = \underset{y}{\operatorname{argmax}}\ P(x|y)P(y)$

- How do we compute $P(y)$?
- How do we compute $P(x|y)$?

Automatic language detection

- How do we find *P*(*y*)? (called **prior**)

- In this case, essentially two choices:
  - All languages have the same prior (uniform prior)
    - *P*(*y* = PT) = *P*(*y* = ES) = *P*(*y* = FR) = 1/3
  - Estimate prior from the data
    - *P*(*y*) α (size of data of language *y*)

  - In our case, we use the uniform prior
  - Since we want the argmax, we can forget about the prior

$$\operatorname*{argmax}_y P(x|y)P(y) = \operatorname*{argmax}_y P(x|y)\frac{1}{3} = \operatorname*{argmax}_y P(x|y)$$

(MAP with uniform prior = ML)

Maximum *A Posteriori* Estimator

$$y^* = \operatorname*{argmax}_y P(y|x)$$

Bayes Rule

$$= \operatorname*{argmax}_y \frac{P(x|y)P(y)}{P(x)}$$

Drop P(x)

$$= \operatorname*{argmax}_y P(x|y)P(y)$$

Uniform Prior

Maximum Likelihood Estimator

$$= \operatorname*{argmax}_y P(x|y)$$

Automatic language detection

- How do we find *P*(*x*|*y*)? (called **class conditional**)

- For example, what’s *P*(“eu fui“ | PT)?
  - Maybe count how often “eu fui” appears in the PT data...

- What about this one?

P(“eu fui à praia com os meus amigos, mas começou a chover por isso fomos ao cinema ver o ‘Shrek’, que é um filme de animação”| PT)

  - Probably never appears in the training set for any language!
  - Most non-small sentences would get *P*(*x*|*y*) = 0 for every *y* ☹
  - What would be the best *y* ???

Automatic language detection

- Slight change of notation:

$$P(\text{"eu fui"}|\text{PT}) = P(\text{"eu\_"}, \text{"u\_f"}, \text{"\_fu"}, \text{"fui"} \mid \text{PT})$$

  - i.e. we represent the sentence with all its triplets
  - this is completely equivalent to the original formulation

- Naïve Bayes: assume conditional independence

$$\begin{gathered}P(\text{"eu fui"} \mid \text{PT}) = P(\text{"eu\_"}, \text{"u\_f"}, \text{"\_fu"}, \text{"fui"} \mid \text{PT}) \\ = P(\text{"eu\_"} \mid \text{PT}) \; P(\text{"u\_f"} \mid \text{PT}) \; P(\text{"\_fu"} \mid \text{PT}) \; P(\text{"fui"} \mid \text{PT})\end{gathered}$$

Automatic language detection

- We just need to estimate probabilities of the form *P*("abc" | y), where "abc" are any three characters
  - Can be estimated from train data <u>just by counting</u>:

$$P(\text{"abc"} \mid \text{PT}) = \frac{\#\text{"abc" in PT train data}}{\#\text{triplets in PT train data}}$$

- Example:
  - "fui" appears 10^2 times in PT train data
  - there are 10^6 triplets in PT train data
  - then, $P(\text{"fui"} \mid \text{PT}) = 10^{-4}$

Automatic language detection

- No problem with long sentences!

P(“eu fui à praia com os meus amigos, mas começou a chover por isso fomos ao cinema ver o ‘Shrek’, que é um filme de animação”| PT) =

= *P*(“eu_”|PT) *P*(“u_f”|PT) *P*(“_fu”|PT) ... *P*(“açã”|PT) *P*(“ção”|PT)

  - “eu_” probably appears in all languages
  - same for “u_f”, “_fu”, “fui”, and so on

  - if a few triplets do not appear in a language, that can be solved with **smoothing**

(log trick)

- Each *P*(“abc” | *y*) probability of the order of 10^{-4} to 10^{-7}
- Sentence with *N* characters has (*N*-2) triplets
- Sentence with 60 characters (10-12 words) has probability of order $(10^{-4}$ to $10^{-7})^{58} = 10^{-232}$ to 10^{-406}
- Very easy to get underflow errors!

- Solution: use log-probabilities, $\log(10^{-406}) = -406*\log(10) =$ $= -934.85$, no risk of underflow, and same argmax:

$$\arg\max_y P(x \mid y) = \arg\max_y \log\left[P(x \mid y)\right]$$

- Products of probabilities become sums of log-probabilities

$$\log\left[P(\text{"eu\_"} \mid \text{PT}) P(\text{"u\_f"} \mid \text{PT}) P(\text{"\_fu"} \mid \text{PT}) P(\text{"fui"} \mid \text{PT})\right] =$$

$$= \log\left[P(\text{"eu\_"} \mid \text{PT})\right] + \log\left[P(\text{"u\_f"} \mid \text{PT})\right] + \log\left[P(\text{"\_fu"} \mid \text{PT})\right] + \log\left[P(\text{"fui"} \mid \text{PT})\right]$$

Demo time!

- Feel free to suggest a few sentences to test...

Automatic language detection

- Why is “não sei” Portuguese?

log[P(x|y)]	PT	ES	FR
“não“	-7,561	-14,777	-15,513
“ão_”	-5,655	-10,812	-11,252
“o_s”	-6,779	-7,234	-9,674
“_se”	-6,000	-5,997	-6,571
“sei”	-9,464	-10,188	-8,589
“não sei”	-35,459	-49,008	-51,599

- Best: PT, second best: ES
  - large log-ratio ➔high confidence in result

$$\text{log}-\text{ratio} \overset{\text{def}}{=} \log\left(\frac{P(x|y=\text{PT})}{P(x|y=\text{ES})}\right) = \log(P(x|y=\text{PT}) - \log(x|y=\text{ES}) = 13.549$$

Automatic language detection

- Why is “eu fui”

French?

log[P(x|y)]	PT	ES	FR
“eu_“	-7,417	-11,610	-8,198
“u_f”	-10,024	-10,196	-9,014
“_fu”	-7,960	-7,067	-8,366
“fui”	-12,456	-13,531	-11,640
“eu fui”	-37,857	-42,404	-37,218

- Best: FR, second best: PT
  - small log-ratio ➔ low confidence in result

$$\text{log−ratio} = \log(P(x|y = FR) - \log(x|y = PT) = 0{,}639$$

Naïve Bayes (summary)

- Goal: maximize *P*(*y*|*x*)
- Bayes Rule, drop *P*(*x*) from argmax, uniform prior → maximize *P*(*x*|*y*)
- Assume features conditionally independent:

$$P(x_1, x_2, \dots, x_N | y) = P(x_1|y)P(x_2|y) \dots P(x_N|y)$$

- Advantage: number of parameters to estimate
- *P*("fui"|*y*): easy to estimate from train data (just count)
- *P*("eu fui à praia com ..."|*y*): hard (usually impossible) to estimate directly

- Usually NOT a good model of the data!
  - Is ("_fu"|PT) really independent of ("fui"|PT)?
- Sometimes, the best model which can be used in reasonable time...
- In this case, it works well even though it is not a perfect model