Laboratory 6

In this laboratory we will focus on generative models for classification of text.

Multinomial models

Dante's "Divina Commedia" is a well known example of a poem in which the author uses different styles through the different parts of the work. The poem is divided in three *Cantiche*, respectively "Inferno" (Hell), "Purgatorio" (Purgatory) and "Paradiso" (Heaven). Each part is written using different linguistic styles, moving from less to more aulic as we progress from Hell towards Heaven. Each Cantica is divided in Canti, 34 for Inferno, and 33 for Purgatorio and Paradiso. Each Canto consists of a variable number of verses (115 to 160), organized in tercets.

USA METODI STATISTICI PER ANALIZZARE COME LE DIFFERENZE STILISTICHE SI POSSANO USARE PER COMPRENDERE LA CANTICA DI UNA DATA TERZINA.

In this laboratory we will use statistical methods to analyze how the stylistic differences can be exploited to understand the Cantica of a given tercet. In particular, we will build a multinomial word model for the three Canticas and classify tercets excerpts. To avoid biased results, the tercets used to train the model will be different from those we evaluate on. → DIVERSO DATASET TRAINING E VALIDATION

Dataset

The files `data/inferno.txt`, `data/purgatorio.txt`, `data/paradiso.txt` contain tercets for the three different parts. To simplifying parsing, the files are already organized so that each line corresponds to a tercet. In the following examples, we use 25% of the tercets as validation, and the remaining ones as training data. Loading function examples can be found in `data/load.py`. You can load the data using the function `load_data`. This returns a list with all tercets for the three canticas. For each list, you can split the tercets in training and validation lists using function `split_data` (again in file `load.py`)

A multinomial model for text

OGNI PAROLA È UN EVENTO. ABBIAMO N VAR INDIPENDENTI CHE DESCRIVONO CIASCUNO DEGLI N TOKEN. D È TUTTO L'INSIEME DELLE PAROLE POSSIBILI CON DIMENSIONE M

We have seen two, equivalent, models for text data:

- We can assume that each word is an event. We have N independent categorical R.V.s $X_i \in \mathcal{D}$ that describe each of the N tokens of the document. $\mathcal{D}$ is the set of all possible words (dictionary), with size M. The whole document is a realization of $(X_1 \dots X_N)$. The model parameters are the word probabilities $\pi_i = P(X_j = \mathcal{D}_i)$.

- We can represent the document in terms of occurrences of each word. In this case, we have M R.V.s Y_j that describe the number of occurrences of each word, $Y_j \in \mathbb{N}$. Each Y_j is related to the R,V.s X_i of the first model as $Y_j = \sum_{i=1}^{N} \mathbb{I}\left[X_i = j\right]$, where $\mathbb{I}$ is the indicator function.

SI PUÒ RAPPRESENTARE IL DOCUMENTO IN TERMINI DI OCCORRENZE DI CIASCUNA PAROLA. ABBIAMO M VAR CHE DESCRIVONO IL NUMERO DI OCCORRENZE DI OGNI PAROLA.

The likelihood, in the first case, is given by

$$\mathcal{L}_X(\mathbf{\Pi}) = \prod_{i=1}^{N} \pi_{x_i} = \prod_{i=1}^{N} \prod_{j=1}^{M} \pi_j^{\mathbb{I}[x_i=j]}$$

and the log-likelihood is thus

$$\ell_X(\mathbf{\Pi}) = \log \mathcal{L}_X(\mathbf{\Pi}) = \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{I}\left[x_i = j\right] \log \pi_j = \sum_{j=1}^{M} N_j \log \pi_j$$

where N_j is the number of occurrences of word j.

In the second case the likelihood is

$$\mathcal{L}_Y(\mathbf{\Pi}) = \frac{N!}{y_1! \dots y_M!} \prod_{j=1}^{m} \pi_j^{y_j} = \frac{N!}{N_1! \dots N_M!} \prod_{j=1}^{m} \pi_j^{N_j}$$

since y_j is the number of occurrences of word j, i.e. $y_j = N_j$. The log-likelihood is thus

$$\ell_Y(\mathbf{\Pi}) = \log \mathcal{L}_Y(\mathbf{\Pi}) = \xi + \sum_{j=1}^{m} y_j \log \pi_j = \xi + \sum_{j=1}^{m} N_j \log \pi_j$$

where ξ is the logarithm of the multinomial coefficient, which **does not depend on the parameters**. Thus, we have that

$$\mathcal{L}_X(\mathbf{\Pi}) \propto \mathcal{L}_Y(\mathbf{\Pi})$$

Since the likelihoods are proportional, inference based on $\mathcal{L}_X$ and $\mathcal{L}_Y$ will be the same:

- ML estimates will be the same for both models (the terms ξ is irrelevant for maximization)
- Likelihood ratios and class posterior probabilities will also be the same, since the proportionality term $\alpha = e^{\xi}$ simplifies

We can therefore use both models to represent the text.

Training the model

ML estimates of the model parameters can be obtained by maximizing, for each cantica, either ℓ_X or ℓ_Y. Let $N_{c,j}$ be the number of occurrences of word j in cantica c. Then, the ML parameters for cantica c are

$$\pi_{c,j} = \frac{N_{c,j}}{N_c}$$

with $N_c = \sum_{j=1}^{m} N_{c,j}$ being the number of words of cantica c. We can observe that $\pi_{c,j}$ is equal to the frequencies of word j in the document.

To compute the frequencies we first need to build the dictionary of all possible words. We consider only words that appear in the training set — if test or validation tercets contain unknown words we can simply discard them, as they would not, in any case, provide support for any of the three hypotheses under consideration.

SE NELLA VALID C'È UNA PAROLA CHE NON ESISTE LA IGNORI

TERZINA

Suggestion: you can treat the tercets of each part as if they were a single, long body of text (the proposed solution, however, explicitly processes the tercet lists).

Once we have computed the dictionary of words, we can proceed with computing the occurrences of each word in each of the three canticas.

DEVI CALCOLARE IL DIZIONARIO

DIZIONARIO PYTHON (PAROLE, OCCORRENZE)

Suggested method 1: We can represent word counts using python dictionaries, where keys are words and values are occurrences

ETICHETTI OGNI PAROLA CON UN NUMERO DA 1 A M-1 E METTI LE OCCORRENZE

Suggested method 2 (alternative to method 1): We can label each word with a progressive number 0 to $M-1$, and create an array for each class storing the occurrences of each word

Finally, we need to normalize counts to obtain frequencies (this is easier if you used method 2).

ESSENDO LA FREQ IL LOG QUELLE CON ZERO AVRANNO -INF COME LOGARITMO

NOTE: for some classes word frequencies may be 0. Since the prediction is based on the logarithm of the frequencies, this implies that we will get `-inf` values for samples that contain words that do not appear in one cantica, corresponding to a class-conditional probability (likelihood) of 0. This may lead to numerical issues, since we may have prediction likelihoods for different canticas that are 0 *for all three classes* (e.g., a tercet containg a word that does not appear in Inferno, a second word that does not appear in Purgatory and a third word that does not appear in Paradiso). A simple solution to avoid this issue is to remove those words that do not appear in all three canticas. However, these words may actually be significant: if a word appears several times in Inferno and never in Paradiso, then it's a good clue for identifying tercets from Inferno. A better solution consists in augmenting the word counts with pseudo-counts, i.e. rather than using the counts $N_{c,j}$ we use $\widehat{N_{c,j}} = N_{c,j} + \varepsilon$, where ε is a (small) value. In this context ε is a hyperparameter, and we would need to estimate its optimal value on a validation set. For this laboratory, you can simply try different values and see how this affects the results. Below you will find the results with $\varepsilon = 0.001$, but higher accuracy can be obtained, for example, with $\varepsilon = 1$. In a Bayesian framework, pseudo-counts $\varepsilon = 1$ can be justified as an approximation of a more complex Bayesian model (but we won't see the details).

ALCUNE PAROLE POTREBBERO APPARIRE ZERO VOLTE IN UNA CANTICA MA PIÙ VOLTE IN UN'ALTRA E AIUTARE A RICONOSCERE QUELLE QUINDI ELIMINARE QUELLE PAROLE NON È UNA BUONA SOLUZIONE

PRIMO METODO

```python
def S1_buildDictionary(lTercets):
    '''
    Create a set of all words contained in the list of tercets lTercets
    lTercets is a list of tercets (list of strings)
    '''

    sDict = set([])
    for s in lTercets:
        words = s.split()
        for w in words:
            sDict.add(w)
    return sDict
```

→ QUESTO CREA SEMPLICEMENTE UNA LISTA IN CUI LE PAROLE → ARRAY [PAROLA 1, PAROLA 2, ...] DI OGNI TERZINA APPARE ALMENO UNA VOLTA.

POI DEVI INIZIALIZZARE LE OCCORRENZE AL VALORE DI ε POICHÉ PER LE PAROLE CHE NON CI SONO POTRESTI AVERE 0 E POI CALCOLANDO IL LOG -inf

LE OCCORRENZE SONO CALCOLATE PER OGNI CLASSE

INFERNO:

PURGATORIO:

PARADISO:

SOMMI TUTTE LE OCCORRENZE PER OGNI CLASSE DI OGNI PAROLA

$$\log \frac{\text{CONTEGGIO PAROLA}}{\text{NUMERO TOT DELLE OCCORRENZE PER TUTTE LE PAROLE IN QUELLA CLASSE}}$$

→ LO CALCOLI PER OGNI PAROLA

IN $h\_cls\log Prob[\underbrace{cls}_{\text{CLASSE}}][\underbrace{w}_{\text{PAROLA}}]$ HAI I LOG DELLA FREQ DI QUELLA PAROLA IN QUELLA CLASSE

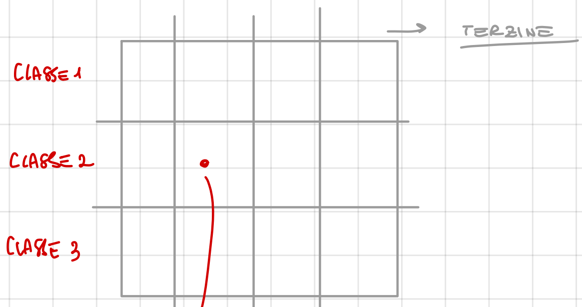
QUINDI ORA HO UNA MATRICE CHE HA COME RIGHE LE CLASSI E COME COLONNE LE PAROLE CON IN OGNI CASELLA LA FREQ PER QUELLA PAROLA

{ LE PAROLE SONO PRESE DA TUTTE LE CLASSI

FREQ DI QUELLA PAROLA IN QUELLA CLASSE

DEVI CALCOLARE LA MATRICE DI VEROSOMIGLIANZA QUESTA PRENDE IN INPUT LA MATRICE IN CUI PER OGNI PAROLA E CLASSE C'È LA FREQ CON CUI APPARE

S1_COMPUTE_LOGLIKELIHOODS → SOMMA LE FREQ PER OGNI PAROLA NEL TERZETTO IN TUTTE LE CLASSI. OGNI DIZIONARIO DI LOG-VEROSOMIGLIANZA VIENE CONVERTITO IN UNA COLONNA DELLA MATRICE DI LOG VEROS.



PROB CHE IL TERZETTO APPARTIENE A QUELLA CLASSE

Predicting the cantica

We now turn to prediction for a given tercet t. Again, we can represent the tercet in terms of its tokens $\boldsymbol{x} = [x_1 \ldots x_N]$ (here N refers to the length of the validation tercet, and x_i is the i-th word of the tercet), or in terms of word occurrences $Y_1 \ldots Y_M$.

In the first case, we can compute class-conditional log-likelihoods as

$$\log f_{\boldsymbol{X}|C}(\boldsymbol{x}|c) = \sum_{i=1}^{N} \log \pi_{c,x_i}$$

For each word x_i, the corresponding log-probability is $\log P(X_i = x_i|c) = \log \pi_{c,x_i}$, and we simply sum the log-probabilities of each word. Note that, in the second case, we should compute class-conditional log-likelihoods as

$$\log f_{\boldsymbol{Y}|C}(\boldsymbol{y}|c) = \sum_{j=1}^{M} y_j \log \pi_{c,j} + \xi$$

however, we can ignore the term ξ and simply compute again the log-likelihoods as $\sum_{j=1}^{M} y_j \log \pi_{c,j}$. As we have seen, the term ξ is indeed irrelevant for inference. y_j is the number of occurrences of word j in the test tercet.

If you used suggested method 2 to represent the probabilities $\pi_{c,j}$, then the result can be expressed as a vector product $\log f_{\boldsymbol{Y}|C}(\boldsymbol{y}|c) = \boldsymbol{y}^T \boldsymbol{w}_c$, where $\boldsymbol{w}_c = [\log \pi_{c,1} \ldots \log \pi_{c,M}]$

In both cases, compute for each sample the class-conditional log-likelihoods (up to the constant factor ξ), and use them to compute class posteriors. As for the continuous case, the suggestion is to create a matrix of scores S, where each row corresponds to a class and each column to a test sample (tercet).

We can now analyze the classification performance for different tasks

1. Closed-set multiclass classification: consider all 3 classes, and compute class posterior probabilities assuming uniform priors $P(c) = 1/3$. Given the score matrix S, this is done exactly in the same way as for the continuous case (Laboratory 5). Measure the accuracy for each class (number of correct predictions for samples of each class over number of samples of the class). If you use the provided split, with $\varepsilon = 0.001$, you should obtain 53% for Inferno, 48% for Purgatorio and 57% for Paradiso. Overall, the accuracy will be 52%. This implies we can identify the cantica with an accuracy better than chance.

2. The lexical choices in Purgatorio are something in between those of Paradiso and Inferno. We thus consider pairs Inferno - Paradiso, Inferno - Purgatorio and Purgatorio - Paradiso. These are 3 binary tasks. Repeat the estimation / inference for these 3 tasks, each time considering only 2 classes. You should get:

   - Inferno - Paradiso: 75%
   - Inferno - Purgatorio: 62%
   - Purgatorio - Paradiso: 65%

   SE LO FAI A COPPIE CAPISCI QUAL È PIÙ DIFFICILE DA DISCRIMINARE

   This confirms that Inferno and Paradiso are easier to discriminate, whereas Purgatorio is more similar to both other canticas.

PUOI "BARARE" E PARTIRE DA QUELLI GIÀ CALCOLATI

NOTE: we can "cheat" and obtain the solution to the binary problems from the score matrix S of the three-class problem. Indeed, if we consider only the rows of S that we are interested in, these give us the matrix S_b of class-conditional likelihoods for the two classes. The difference between this approach and re-training using only two classes is that in the former case the binary matrix S_b contains also conditional likelihoods for words that may not appear in neither of the two considered class, but appear in the third one - these would be discarded if we trained from scratch. However, since the additional words would have a frequency proportional to ε for both classes, their contributions would disappear when computing class posteriors.

Project

Since multinomial models are suited for discrete data, while project samples do not contain discrete features, we won't use these models for the project.