Tips for the NLP Project

Master MIASHS

The goal of this note is to provide a clear and structured pathway for designing a complete NLP project. Although different datasets or tasks may require adaptations, the general logic of a robust pipeline remains the same: **understand the data, make a complete descriptive analysis, represent the language properly, and compare several models rigorously**.

1. Choosing a Dataset and Defining the Task

The first step of your project is to select a dataset that is rich enough to justify an NLP approach. You should then clearly formulate the task: are you trying to classify documents, predict a numerical variable from text, identify topics, perform an unsupervised exploration, or anything else?

Whatever the objective, it must be **stated explicitly** at the start of your NoteBook and presentation, together with the target variable and the expected input format.

2. Importing the Data and Initial Exploration

Once the dataset is chosen, load it and inspect it carefully. A simple `df.head()` is often enough to obtain an initial feeling for the variables, their types, and the general structure of the data.

At this stage, you must identify potential difficulties: missing values, heterogeneous formats, unbalanced classes, or noisy text.

3. Cleaning, Preprocessing, and Descriptive Analysis

Preprocessing. A clean dataset is essential for any meaningful learning process. For textual variables, this step may include lowercasing, punctuation handling, normalization, deduplication, or other linguistic cleaning procedures. However, each preprocessing decision must be justified: removing stop-words, for instance, is not always desirable and should be argued.

After preprocessing, you should provide a brief descriptive analysis to verify the quality of the cleaned data. Typical indicators include text length statistics, vocabulary richness, and any additional measure showing that the dataset is now consistent and ready for the NLP pipeline.

Descriptive Analysis. After cleaning the dataset, produce a **complete descriptive analysis**. Document the number of observations and variables, produce a concise statistical summary for all numerical covariates (mean, standard deviation, minimum, quartiles (with median), and maximum). This initial overview should help identify irregularities or unexpected patterns.

Each variable must be examined for missing values: report the number of NaNs, justify the chosen imputation or removal strategy, and indicate how outliers or aberrant values are handled.

You should also clarify the nature of each covariate (numerical, categorical, or ordinal), as this determines which transformations and visualizations are appropriate.

Graphical exploration is also need when possible. Histograms, density plots, bar charts, boxplots, violin plots, and contingency tables may all be used to highlight the structure of the data and reveal class imbalances or distributional differences.

4. NLP Processing and Representation of Text

Your project must explicitly justify *how* the text is converted into vectors. The choice of representation is one of the most important decisions you will make.

Begin by selecting a tokenizer (naive, SpaCy, or a pretrained transformer tokenizer). Explain why this tokenizer seems to be adapted to your dataset.

Then choose a model to represent the text: classical approaches such as Word2Vec, GloVe, FastText, or more recent architectures such as BERT, RoBERTa, or SBERT. For the chosen model, describe succinctly the **underlying architecture**, the **motivation behind it**, and the **loss function used during training**. Your goal is to show that you understand *how* the representation vectors are created and designed.

Finally, if aggregation strategy is used, justify your approach: SIF embeddings, mean pooling, max pooling, or transformer-specific representations such as the `[CLS]` token.

5. Modelling, Training, and Evaluation

Once the embeddings are computed, choose the appropriate modelling strategy. A classification task will typically rely on logistic regression, SVM, random forests, or neural networks; a regression task will focus on linear models or gradient boosting; unsupervised settings may involve clustering or topic modelling.

Split your dataset into training and test sets. If the task involves categorical labels, consider whether you should **stratify** the split, and *justify* your choice.

Train **several** models and compare them fairly. Performance must be reported on the test set using suitable metrics (accuracy, precision, recall, F1-score, Brier score for classification; RMSE or MAE for regression). Your comparison must include a discussion of discrimination, calibration when relevant, and practical trade-offs.

A complete project never relies on a single model. You are expected to compare multiple approaches and argue why one method is more appropriate for your task.

A successful project is one where every choice is motivated, every method is explained, and every conclusion is supported by evidence. Have fun!