EPITECH. TECHNOLOGY

TRAVEL ORDER RESOLVER

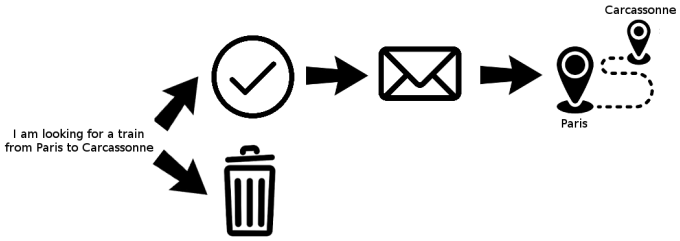
NATURAL LANGUAGE PROCESSING

TRAVEL ORDER RESOLVER

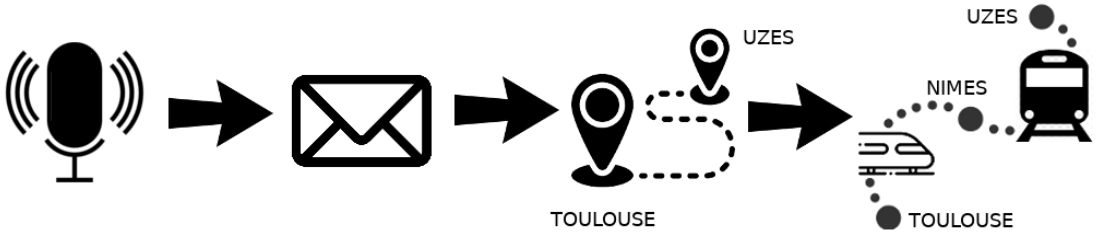
Build a program that processes at least text commands to issue an appropriate itinerary.

More specifically, your software will receive trip orders as text (for instance an email), and potentially from voice (for instance some phone recording), and will output at least one appropriate travel line that fits the expectations of the customer.

It discriminates between valid and non-valid orders, and it identifies departure and destination.



To convert voice data into text data, you may add an optional voice recognition step.



However, keep in mind that the project is mainly focused on NLP, and will be mainly evaluated with written inputs.

After text understanding, you may add an optimization process that finds the best path in a distance graph, in order to get the optimal train connections.

We assume that you already master the basics of Machine Learning and have some practice with related libraries.

Tools and techniques

Natural language processing

So you basically want to extract meaning from a text. NLP is a ever-expanding field of research of more than 50 years, with a wide range of techniques. That means, you have a lot of research to perform and choices to make before writing any line of code.

Some algorithms work with mere bags of words and use simple statistics on the data. They are very useful for categorization, or identification of author, context, etc.

Some other algorithms try to preserve the order of words or understand the grammar of specific sentences. They are often necessary for understanding fine information on small data.

For this, one must establish relations between words that may be in the beginning or in the end of the sequence of words. *Transformers* is one of the most effective architecture for handling this (the T of ChatGP**T** stands for Transformers).

Transformers were introduced in Attention is all you need.

Example sentences

We have high expectations on the sentences understanding. Finding town name and differentiating origin and destination is a complex task that should not be underestimated.

Some town names are made of common words. For instance *Port-Boulet* is a french town, but both “port” and “boulet” are common nouns.

Some town names may also represent people name, such as: Albert, Paris, Lourdes, …

> Comment me rendre à *Port Boulet* depuis *Tours* ?
> Je veux aller à voir mon ami Albert à *Tours* en partant de *Bordeaux*.

Recognizing origin and destination should not be underestimated, as neither the order of the names (usually origin first, destination second) neither the adverbs (like “de”, “depuis”, “à”, “vers”) are sufficient for this task.

> Je voudrais un billet Toulouse Paris.
> Je souhaite me rendre à Paris depuis Toulouse.
> A quelle heure il y a-t-il des trains vers Paris en partant de Toulouse ?

Keep in mind that people may not use capitals, accents, hyphens, …
This must be handled.

How about handling misspelled words and incorrect grammar ? (Bonus)

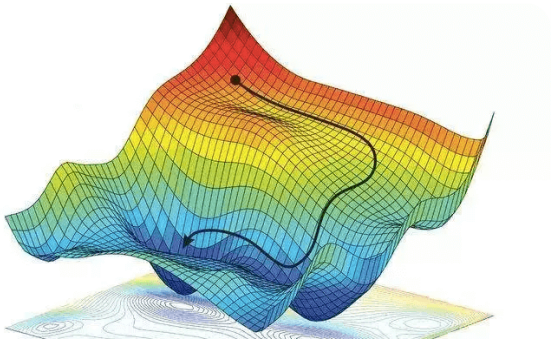
Graph optimization

Finding an optimal path in a graph is a deterministic problem, which belongs to the **P** class.

Obviously, since there are a finite number of train stations, some brute-force search would work. But do you really want to explore an exponential number of routes? Certainly, you do not!

Take a look at the literature about algorithmic and complexity, before jumping right ahead to an implementation of your algorithm. It is often more interesting to program it by yourself from scratch.

More generally, optimization algorithms are at the core of many ML solutions. In particular deep learning, which is based on backpropagation.



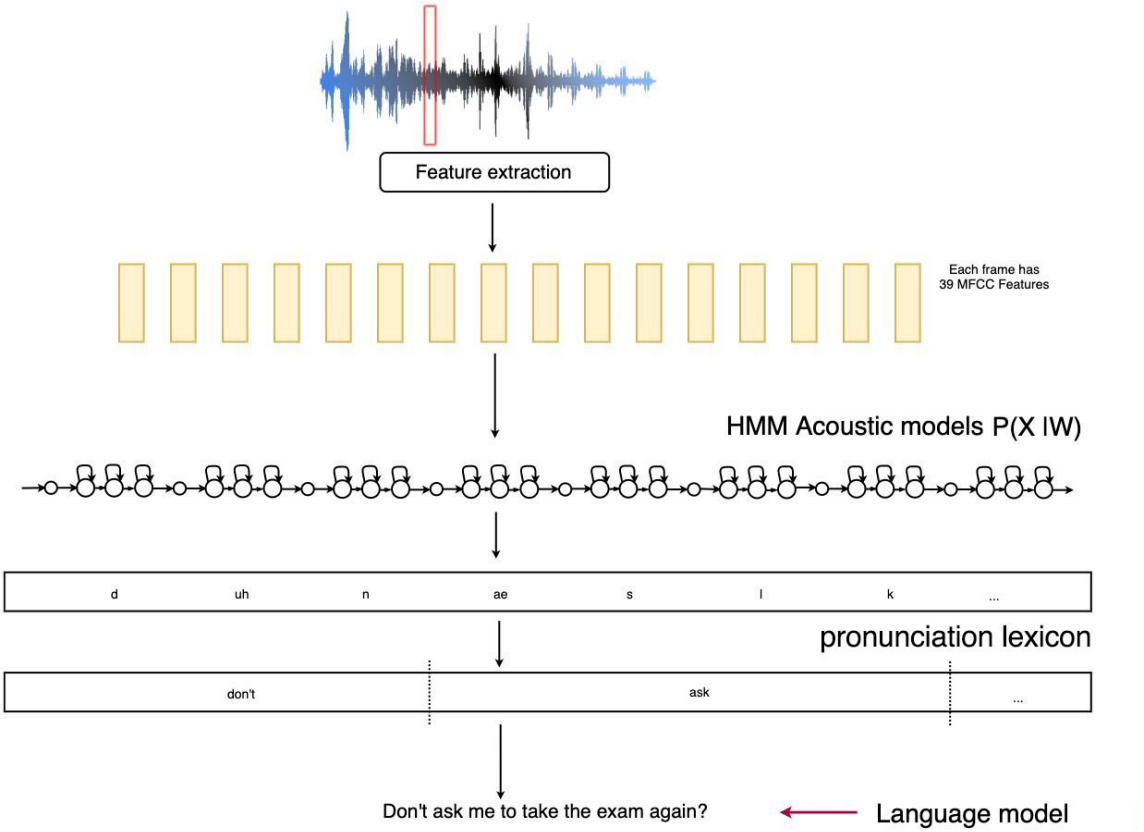
This is a *not* an large task compared to NLP: some classical (and easy to implement) algorithms can be sufficient.
However you are expected to understand the algorithm(s) and their complexity.

Voice recognition

Since this is an isolated component of your architecture, and because it refers to a well-defined single problem, voice recognition should actually be the easiest part of the job here.

Nevertheless, the mathematics behind voice recognition are extremely interesting, and we strongly advise those who want to specialize in signal recognition to investigate how Hidden Markov Models work.

Building such a model from scratch is far beyond the scope of this first project, but reaching a deeper understanding of how the components of a voice recognition toolkit work together would be great.



This is a light task compared to NLP, because you can use predefined components or API. Be careful that some toolkit may do part of the NLP workflow (e.g. identifying the town names), and you should **not** rely on that.

Specifications

Input/output

The main **NLP** component should be able to:

- ✓ receive some text file in input ;
  To be evaluated, your NLP part shall read sentences (one per line) from any file or URL.
  A simple option is to build a command line program that read sentences on stdin (and use, for instance, cat or curl to get the sentences).
  Each line shall start have the following format : *sentenceID*,*sentence*
  The *sentenceID* will not be directly used by your program, but simply printed with the result, for validation.
- ✓ for each sentence, discriminate if the text is a trip order **formulated in french** ;
  - return a negative answer if it isn't, using the format : *sentenceID*,*Code*, where code can be 'NOT_FRENCH', 'UNKNOWN', 'NOT_TRIP'
  - return a triplet *sentenceID*,*Departure*,*Destination* if it is.

All input and output files **MUST** use UTF-8 coding.
Strictly respect the output format, since it will be used when evaluating of your work.
Two files (`sample_nlp_input.txt` and `sample_nlp_output.txt`) are provided to guide you.

The optional **voice component** should be able to:

- ✓ record a vocal signal or read from an audio file;
- ✓ write the sentence (either on stdout or into a file) if the input is **formulated in french**, using a format compatible with the NLP component.

The final pathfinder component must be able to:

- ✓ receive as input a triplet *sentenceID*,*Departure*,*Destination* ;
- ✓ return a **minimal train route** as a sequence of cities:
  sentenceID,*Departure*,*Step1*,*Step2*,...,*Destination*.

Datasets

Many models, such as *CamemBERT* are pre-trained. However they have been trained with generic texts, and they need to be **fine-tuned** (by doing additional training) for a given problem

You will probably not find any existing dataset, and it is your job to build a training set for the NLP component (as well as a test set).

Be careful to include various kinds of trash texts, and more important that real orders explore the variety of grammatical structures.

As every person write differently, you will get a much wider dataset with more diversity if you do it collectively, on any collaborative media (Teams, Discord, Github, …).

On the other hand, the dataset `timetables.csv`, used for the pathfinder, is given to you at the start of the project. So you can evaluate the performance of your algorithm as you build it.

For the sake of simplicity, you can assume that changes do not cost any additional time (duration from A to C through B = duration from A to B + duration from B to C).

You may also use SNCF open data information to have town names instead of station names.

Deliveries

An intermediary delivery is expected, including the pathfinder program.
This part will **NOT** be evaluated after this first review.

The main delivery consist of the NLP and the integration of the several parts, as well as documentation.

For instance, delivery may consist of separated binaries, using Unix pipes to integrate them.
The NLP part **MUST** be isolated for testing and evaluation purpose.

We do **NOT** expect an integrated web application!!!

We obviously also expect a strong documentation, specifically on the NLP part (and preferably in classified PDF files), which includes:

- ✓ the full architecture of the various layers in your application ;
- ✓ a description of the training process (including the delivery of the sets) and the final parameters ;
- ✓ a detailed example of what happens to a given text all through the process ;
- ✓ explanation of different experiments and results.

You must evaluate your NLP processing, and document the progress you've done on the NLP, for instance with the fine tuning.
You obviously need **metrics** to measure this !

Bonus

You can improve this project in many ways, including:

- ✓ benchmarking different models, according to performance criteria ;
- ✓ handling intermediate stops in the route (*Je voudrais aller de Toulouse à Paris en passant par Limoges*) ;
- ✓ considering waiting times in intermediary stations (see `data_sncf.zip` and note that it will change algorithm complexity!) ;
- ✓ hosting your solution on cloud platform (e.g. Microsoft AI platform) ;
- ✓ monitoring CPU/RAM (and therefore cost or carbon footprint) for each request ;
- ✓ handling destination not in the timetable file ;
- ✓ searching through other means of transport ;
- ✓ …

Be careful in that later case that the problem becomes way more difficult that way, both for parsing the data, and because of computational complexity.

v 3.0