T9 - Artificial Intelligence

T-AIA-901

Travel Order Resolver

Natural Language Processing

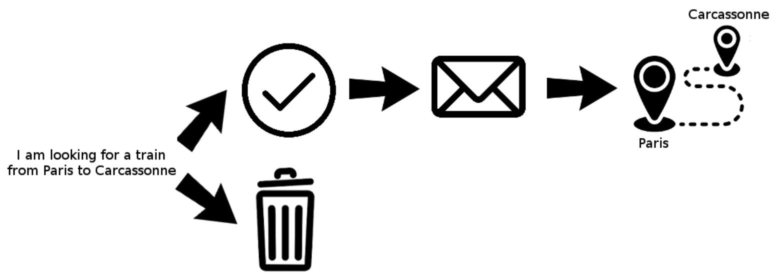
1.1.3

Travel Order Resolver

binary name: *tor_$AcademicYear_$GroupNumber.zip*
language: Python or R or anything else that gets the job done
build tool: no need here

You're expected to build a program that processes voice and text commands to issue an appropriate itinerary.

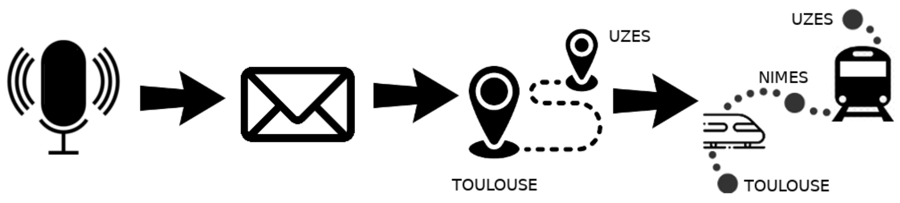
More specifically, your software will receive trip orders as an email or a phone recording, and will ouptut at least one appropriate travel line that fits the expectations of the customer.



You'll discriminate between valid and unvalid orders, and you'll identify departure and destination.

In addition, you might need extra steps to provide absolute satisfaction.

Before text understanding, you should add a voice recognition step, to convert voice data into text data.



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.

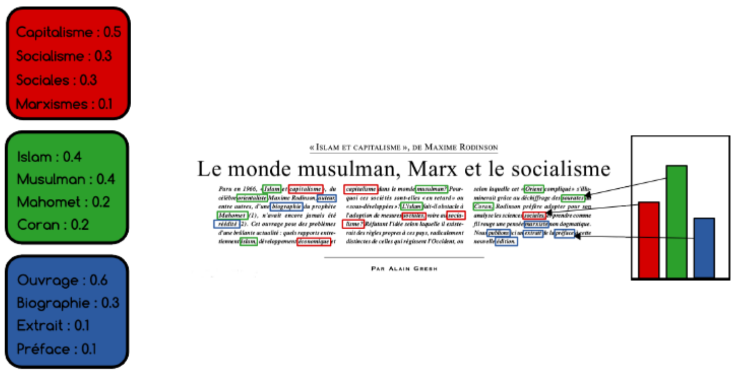
We assume that you already master the basics of Machine Learning and have some practice with related libraries

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.

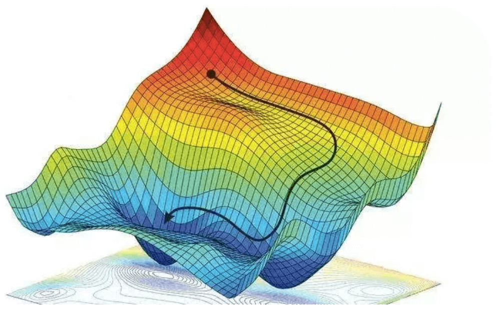
Graph optimization

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 bruteforce search would work. But do you really want to explore an exponential number of routes? Certainly, you do not!

Take a look at the litterature on algorithmics 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.

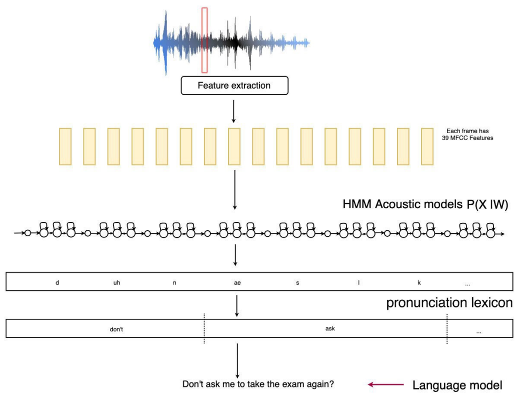


Voice recognition

Since this is an isolated component of your architecture, and because it refers to a well-defined and well-studied 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.


Feature extraction
Each frame has 39 MFCC Features
HMM Acoustic models P(X IW)
d
uh
n
ae
s
l
k
...
pronunciation lexicon
don't
ask
...
Don't ask me to take the exam again?
Language model

Specifications

Input/output

The central component (NLP) should be able to:

- receive as input some raw text file
- discriminate if the text is a trip order **formulated in french**
    - return a negative answer if it isn't
    - return a pair *Departure,Destination* if it is

The voice component should be able to:

- record a vocal signal
- return a text file if the input is **formulated in french**

The final pathfinder component should be able to:

- receive as input a pair *(Departure,Destination)*
- return a **train route**, as a sequence of cities *(Departure,Step1,Step2,...,Destination)*, which is minimal, as far as total travel time is concerned among all possibilities

Datasets

It is your job to build a training set for the NLP component.

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

The test set will be given to you only at the end of the project by your referent, so your algorithm must be trained in a robust way.

On the other hand, the dataset 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. This is the file *timetables.csv*.

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).

Delivery

Your delivery consists in the binary (we recommend you separate the three components so they can be tested individually) and a strong documentation, 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

Bonus

You can improve this project in many ways, including:

- benchmarking different models, according to performance criteria
- answering other order types than travel (eat, drink, sleep, ...)
- searching through other means of transport
- processing other languages than french
- adding a vocal authentication protocol
- considering waiting times at intermediary stations (*data_sncf.zip* might help you with this)

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.