Travel Order Resolver

François DE VATHAIRE

Tio GOBIN

Ryan HEADLEY

Geoffroy HUCK

EPITECH

T-AIA-901 Artificial Intelligence

January 30, 2022

Abstract

The travel order resolver was an application created containing three components: The voice, the Natural Language Processing (NLP) and the pathfinder. The voice component records the voice of a user to a file. The NLP component transcribed this file and saved it to a text file. Later the text is analyzed for a valid train travel request and the departure and destination extracted. The pathfinder component took these two locations and used a graph structure to determine the fastest route between them. Later an application was created to use this functionality.

Keywords: Natural Language Processing, Pathfinder, Artificial Intelligence, train

Travel Order Resolver

The goal of the project was to create a travel order application, which takes the vocal input of a user and returns the shortest train journey between their departure and destination. The results are only provided if the input does in fact contain a request for a train journey between two cities in France.

All information that follows, details the methodology applied to arrive at this application, following its functional order.

Method

Participants

First and foremost, it must be stated that all work completed in this project was equally divided and prepared by all **four** members of the team.

Audio Recording

In order to be able to record an audio message and send it to the transcription part we use WebAudioRecorder.js, a JavaScript library written in 2015 by higuma that can record audio and encode to common formats directly in the browser. It uses external JavaScript libraries to convert the raw audio to wav format. Since JavaScript is slower than native code our encoding times are higher[1].This library is loaded and used as Web Workers which prevents the browser tab from becoming unresponsive while the audio encoding is underway. 3 button are available in this part, a button to record the voice, one to stop recording and one to send the voice message to the transcription part

Transcription

Completing a transcription is a simple process: dividing the audio into individual phonemes, and matching each one with the characters that pronounce them. However, the complexity arrives when many combinations of characters produce extremely similar phonemes. Also, extending all the possibilities into all the existing languages quickly extends the analysis to an incredibly large dataset that would require extensive computing power. As a result, for the scope of this project, a free, open source speech to text software was used: Vosk [4]. Vosk, being available in French and several other languages, takes an audio sample and returns the transcription. Their model was based on BERT and provided results in a matter of seconds.

For the application, the pre-trained Vosk model for the French language was downloaded into the code directory. After the creation of a new audio file, which was always converted to the *wav* format, it was sent as an input to the Vosk model for transcription. The model opened an input stream and transcribed each section of the audio bit by bit, allowing the output to be read almost instantaneously. Since the scope of the project was for small audio samples, the potential for immediate output stream was not taken advantage of since the entire output was found to output at an almost immediate rate.

The outputted sentences from Vosk were then sent to forward to the text extraction functions.

Text Extraction

Travel Request Detection

Once the audio recording has been transcripted, each sentence is passed through a sentence transformer[2] to determine its similarity with the model sentence: *"Je veux prendre un*

train de paris à lyon". This sentence provided one of the simplest and direct requests to travel by train between two places.

After all the provided sentences have been encoded, the cosine similarity was calculated for each, thus providing an absolute measure between zero and one. A visual explanation can be seen below in *Figure 1*.

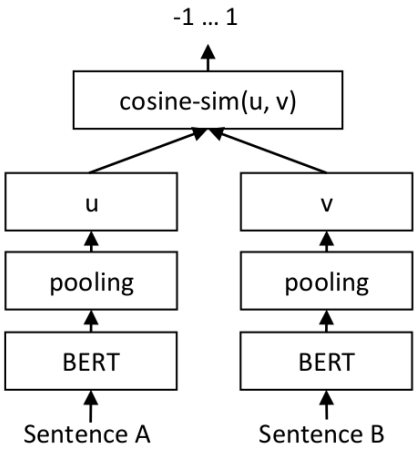

Figure 1: Sentence transformer with cosine similarity between two sentences.

Through trial and error, it was determined that a similarity of higher than 0.75 provided a correct result at least 95% of the time. Therefore, in this stage, the message received was classified as either spam or a true request to travel by train.

Departure & Destination Detection

After the travel request was identified, the departure and destination cities were determined. The spaCy [3] library was applied with the *fr_core_news_md* dictionary to determine the location words in the sentence. For each resulting city, the word before the city in the

sentence was extracted and compared with the predetermined list of words typically used before a departure and a destination in the French language.

Pathfinder

Dataset

The dataset provided for the project incorporated the SNCF train schedules and a variety of information, in several different *txt* files. Since the dataset size is minimal, no database was created but instead the pandas [5] library was incorporated to extract the data. The totality of the dataset structure can be seen in *Figure 2* below.

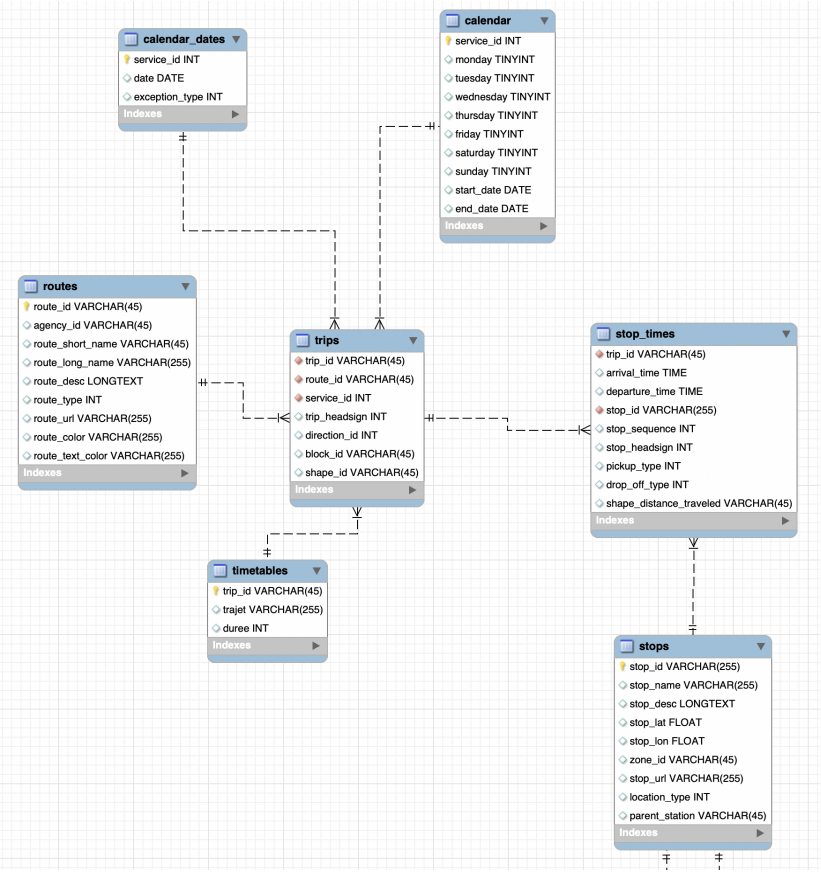

Figure 2: SNCF dataset provided for the project

With the provided tables, all information regarding routes, stations, trip time between specific stations and more could be easily accessed.

Departure and Destination Stations

The pathfinder component receives the city names of the departure and destination, as a string. Therefore a function was written to determine the most relevant train station in the city mentioned. We used the train stations provided by the sncf datasets.

Fastest Route

To get the fastest route, we first needed to identify each route we can take to get from the departure to the destination. To make this happen we first gathered all the information about the existing routes, and the train stations on them. We then made a concordance between the stop point on the routes and the train stations names through the stops.txt, stop_times.txt and trips.txt. From this, we could then generate a graph-like structure containing an entry for each train station, and the other train station they are linked to, allowing us to explore the routes availables and looking for the fastest one. An example of this graph can be seen in *Figure 3* below.

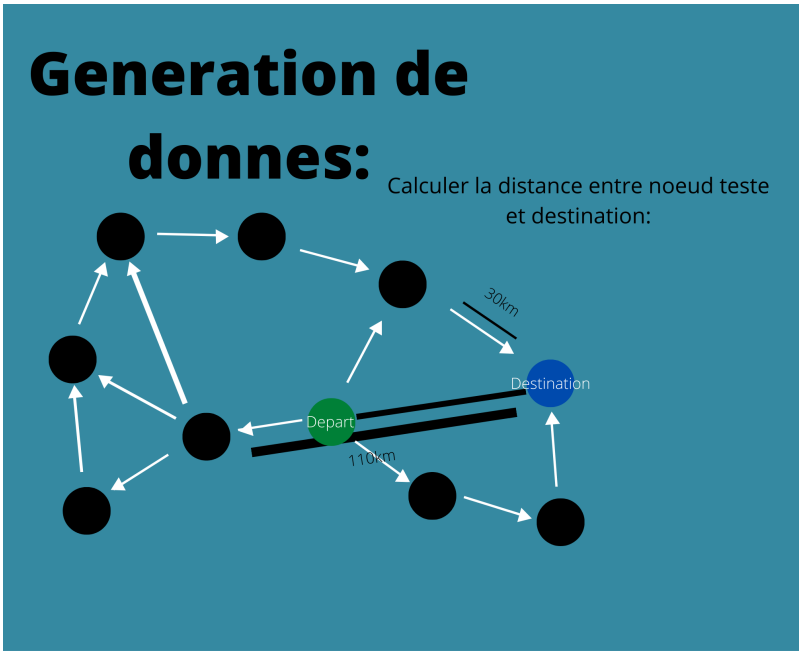

Figure 3: Graph-like structure

The fastest route was calculated based on the stops a train made during a journey on a train line. The duration of the travel between each station necessary was added up to get the total duration of the travel, for a route, and was then compared to every other routes' duration to only retain the shortest one.

The stops contained in this route were then converted to train stations names and retransmitted to the front for display purposes.

Results

The result and the duration of the shortest journey is then displayed on the front end

Speech to Text

To convert an audio file to text, we decided to use Vosk, an open source voice recognition api, as it allowed us to not depend on any external services and not to depend on an internet connection to work efficiently.

We tried a few available pre trained model, and decided to go with Linto, a french model, as it was the most efficient while not being too heavy.

Using it, we transcript an audio file into a list of text, that will then be sent to the NLP component.

Natural Language Processing (NLP)

Each NLP component of the process was found to be effective and efficient. As a result of the complexity and large datasets required to develop personal machine learning models, a variety of libraries with pretrained models were applied to parse the text. Vosk was used for the transcription, while spaCy, Geopy and Sentence Transformers were used to determine the presence of a train travel request and then extract the departure and destination from the transcribed audio. The resulting departure and destination were found to be 99% correct, testing over a variety of combinations of sentences with extremely similar nature, such as:

Voyager en train de lille à lyon Les trains sont mieux.

J'irai de Lille à Lyon à toulon et prendre un bus à marseille.

A toulon et prendre un avion à marseille.

A toulon et marcher à marseille.

Manger des fruits Nager a la plage.

Pathfinder

The pathfinder provided the resulting train route as a string containing each stop included in the route. An image of this can be seen from the application in *Figure 4* below.

Figure 4: Screenshot from application result

Discussion

Users

There was no doubt that the use of an application has been used by the majority of the population with access to a mobile smartphone. The goal of this project was to recreate this functionality given only a fixed dataset from the SNCF database. While the resulting application was proved to provide accurate results, there were many factors that were determined to be lacking in efficiency and practically for the end user.

Firstly, while French is widely spoken throughout the world, there are many people that would prefer to speak their native language. Thus adding a multi-language transcription process would widely increase potential users.

Secondly, a user providing a city name resulted in one train station being selected for every trip. While this would not be a problem in a city with only one station, any city that contains multiple train stations would contain multiple train lines and the total trip time could **drastically** change. This was a feature that was noted to be top priority for any advancement in the development of the application.

Pathfinder

Considering the simplicity of the task, the decision was made not to investigate a Machine Learning solution. This could be an option that would have allowed us to go more in depth for optimization purposes, and where a A*-like system could have been relevant using an angle calculation based on the departure and arrival point (as seen in *Figure 5*, bellow), combined with the duration time between two nodes.

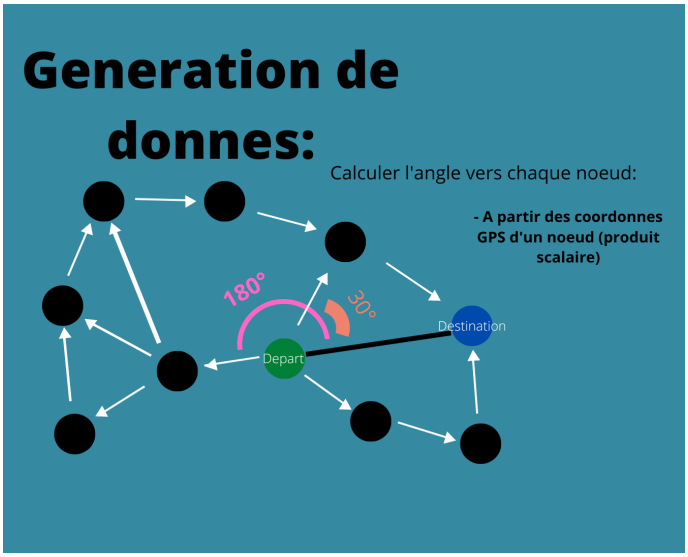

Figure 5: Angle calculation for ponderation purposes

References

[1] Octavian Naicu. "RECORD AUDIO IN HTML5" Web. 22 Jan. 2022.

[2] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence Embeddings Using Siamese Bert-Networks." *ArXiv.org*. 27 Aug. 2019. Web. 27 Jan. 2022.

[3] Honnibal M, Montani I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[4] "Vosk Speech Recognition." *Vosk Offline Speech Recognition API*. Web. 28 Jan. 2022.

[5] "User Guide¶." *User Guide - Pandas 1.4.0 Documentation*. Web. 28 Jan. 2022.

[6] "Welcome to GeoPy's Documentation!." *Welcome to GeoPy's Documentation! - GeoPy 2.2.0 Documentation*. Web. 28 Jan. 2022.