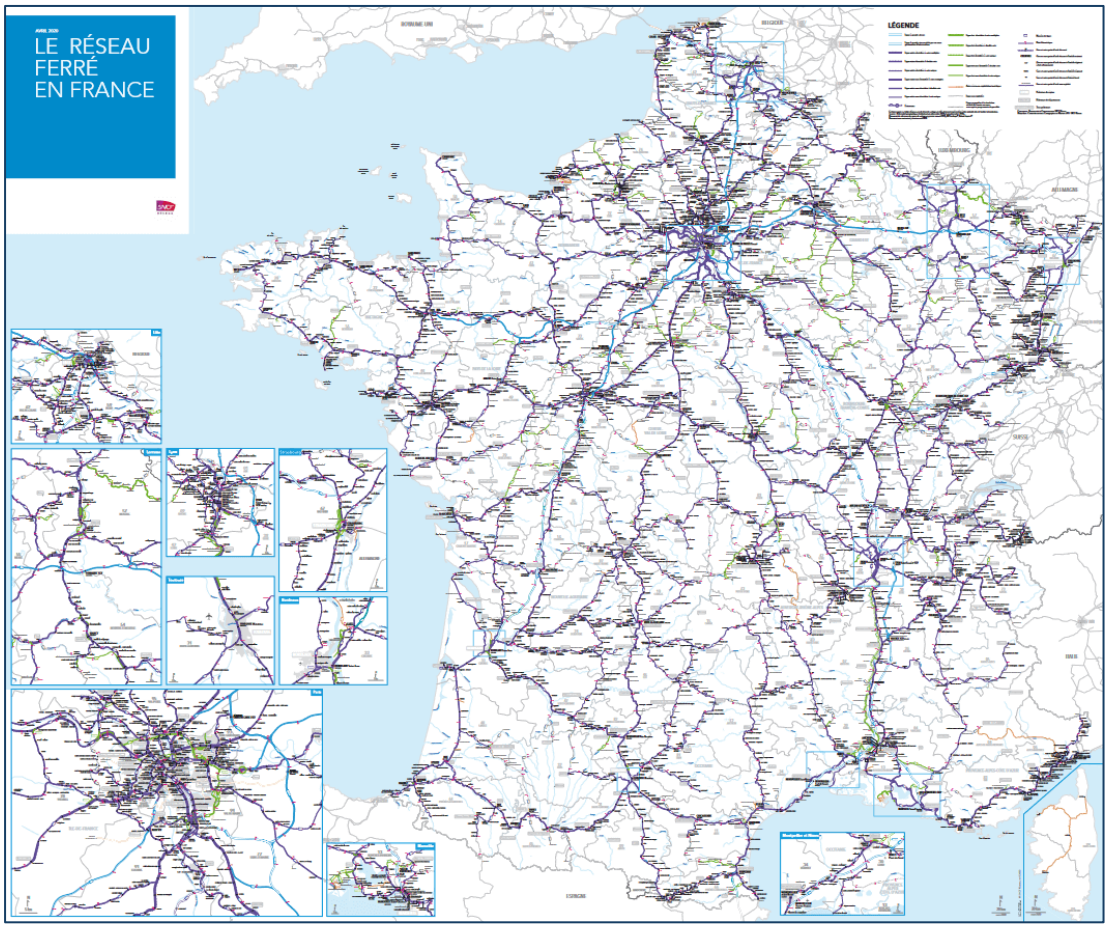
Web scraping Project

Léo RINGEISSEN and Santiago MARTIN – DIA3



Issue / Hypothesis

In today's more climate conscious culture, people are more aware of how their everyday actions play a role in the potential contamination of our planet. For these reasons, many of these people seek to live greener lives, driven by healthy decisions for the environment.

Unfortunately, when it comes to travel it's more difficult to be fully environmentally friendly with the options we have today. Most modes of transportation across long distances have an inevitably large carbon footprint.

It is our mission to help climate conscious travelers mitigate the contaminating effects of their travels, by facilitating the process of planning a fun trip that remains earth-friendly.

Objective

We want to develop a system that allows a user to enter a review/description of a past trip they enjoyed and then makes a travel itinerary recommendation based on their input and the user's ecological goals to reduce their carbon footprint during their travels.

Data Research Method

We will use two sources of information to base our system on :

- An API with carbon emissions data on different travel itineraries provided by SNCF
- Web scraping reviews of travel destinations from TripAdvisor

We will use an NLP information retrieval model like BM25 to process the different reviews of the locations and return a destination recommendation based on matching reviews/descriptions. It will be trained on a corpus composed of the concatenated reviews of destinations we web scraped from TripAdvisor.

Links to Data Sources

The links to our sources are the following :

API : https://ressources.data.sncf.com/explore/dataset/emission-co2-perimetre-complet/information/

Example page to web scrap : https://www.tripadvisor.fr/ShowUserReviews-g187253-r86823297-Marseille_Bouches_du_Rhone_Provence_Alpes_Cote_d_Azur.html

Dataset Production and New Scope

Data Production Notebook

The first step of the project would be to produce a csv dataset out of the information we extracted online thanks to API calls and web scraping. Our process is described in our first data production notebook, which was composed of the following four parts:

- First, we import all of the libraries we'll need to complete this phase of the project

- Second, we perform our API data retrieval from the SNCF API for emissions information on different itineraries and we pair them up with their respective TripAdvisor links

- Third, we use the corresponding links to scrape the TripAdivsor pages for reviews of the destinations

- Finally, we merge all of the gathered data into two csv files, one that's not aggregated and one that is, with concatenated titles and reviews and averaged ratings

As a result of this first phase of development, we came to realize several aspects that would require us to adjust the scope of the project.

Problems and Adjustments to the Project Scope

We learned over the course of this phase of work that a typical method used by companies to prevent bots from illegally scraping their web pages is to deliberately complicate the repetitive tasks a bot would do.

An example of this is having inconsistent page scrolling, where going to page 2 from page 3 won't be the same as if you'd gone from page 1. Another example is not having simple logic between the URLs of different pages. There are also dynamic elements like text translations that do not get retrieved in the initial soup provided by Selenium.

We initially thought that these were merely poor implementations from TripAdvisor, but we now understand that these inefficiencies are deliberate to prevent any automated web scraping processes from taking place, we can therefore not bypass most of these issues without partnering with or paying TripAdvisor to access their data.

As a result of these limitations, as well as data limitations on the end of the contents of the SNCF API, we've reduced the scope of our project to only trips by train whose origin is Paris and whose destination has a page with sufficient reviews in French on TripAdvisor. In addition, many destinations didn't have review pages at all, so we took the reviews of that destination's most iconic landmark, which would serve as an adequate substitution.

Data Quality and Project Scalability

With these adjustments to the project scope, we end up working with around a third (36) of the itineraries accessible with the API (119), and half of the itineraries whose origin is Paris (75), which still leaves us with ample room to make fun and ecological travel recommendations based on traveler preferences and ecological goals, which was the objective we set out to achieve with this project.

Since we couldn't scroll through review pages without being thrown out for bot detection, we resolved the issue by manually selecting the URLs to the first and second review pages (if a second review page is even available). We recognize that this solution would not be scalable for a project with bigger scope, as with more destinations and review pages it would require a lot of manual labor to copy all of the URLs by hand. Fortunately, for the goals of this project, which are to learn to web scrape data and utilize it for Machine Learning applications, this is not a concern, and we can still move forward with our project.

Using automated google engine searches to automatically find URLs was not a viable option too, as the google search engine results are also made difficult to use via a bot, and we still would have to manually monitor the quality of the first URLs yielded by our bot, so it would need more complex and powerful libraries than selenium to be crate a more scalable solution.

Final CSV Generation

By the end of this notebook, we generate an aggregated and a non-aggregated CSV file containing our API and web scraped data. The columns are origin, destination, links for first and second review pages, distance between origin and destination, train trip carbon emissions, the scraped URL (only in the non-aggregated file), the review title (concatenated in the aggregated file), the review content (concatenated in the aggregated file), and the review rating (averaged in the aggregated file).

Machine Learning Methodology

The Machine Learning section of this project was mainly done in a separate ML notebook in which we simply loaded the csv dataset we created in the previous phase of the project.

Information Retrieval Models

Now having our final dataset, composed of review and emissions data for 36 different destinations available from Paris via the SNCF, we could get to the Machine Learning phase of this project.

As specified in our initial project scope statement, our intention was to allow the users to input a review they had for a trip they enjoyed and then we'd return a travel recommendation that would also allow them to achieve their ecological objectives. For this type of task, we'd need information retrieval models, i.e. models that can be queried by training them on a corpus of data. These models would be things like BM25, TFIDF, T5 Flan, and BERT. These are all models we learned about during our NLP course and its various projects, and it's what gave us the confidence that they were indeed adapted to the type of task we required for this web scraping project.

Data Preprocessing

The models we'd be using would require to be trained on a corpus, so a compilation of all our textual data. The production of this corpus took several steps.

First, we had to clean the dataset of any columns we wouldn't be using, such as the origin (since all trips are leaving from Paris), the links to the web scraped pages, and the titles of the reviews. By the end, our cleaned data frame used for the Machine Learning would only contain the destination, the distance and carbon emissions of the train trip from Paris to the destination, the concatenated reviews for the destination, and the average rating associated to the reviews.

Secondly, we'd focus on the reviews column, as our corpus would be built from it. Since different NLP models perform better on different types of data, we decided we'd produce one corpus composed of non-preprocessed data, in which we'd simply tokenize the raw reviews, and one composed of data preprocessed using standard NLP techniques, such as:

- lowercasing all text,
- removing stop words
- removing punctuation and other non-word characters,
- and lemmatization.

We knew that most of our models would likely perform better when trained on the NLP preprocessed data, as it allows for emphasis on essential words/semantic features; however, there are models like BERT that tend to perform better on a maximization of information, so without the removal of stop words and punctuation, or the lemmatization of words. This why in the end we produce one standard corpus and one preprocessed corpus, to test model performance of both data preparation methods.

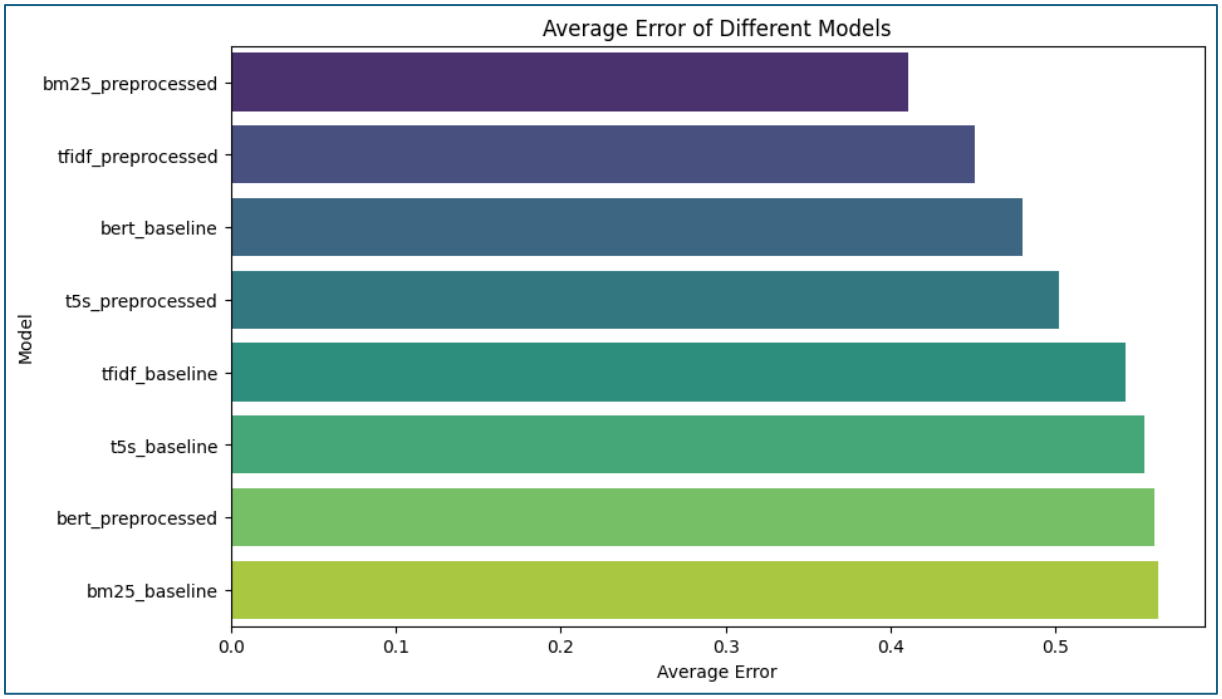
Model Experimentation

As previously mentioned, we sought to test our project idea on a variety of information retrieval models like BM25, TFIDF, T5 Flan, and BERT on both a standard and preprocessed corpus of our reviews.

The way these models function is with a query system. After training the model on the corpus of choice, one can feed the model a query (in our case a review from the corpus linked to one of our destinations) and the model will return the document (in our case the destination) with the highest semantic similarity. In essence, we feed the model a review from one of our destinations, and by seeing how similar that review is to the ensemble of reviews for each destination, it'll return the destinations that best correspond with the queried review.

We had to think about how we'd evaluate our models' performance. The only numerical metric we had at our disposal to associate to the destinations was the average rating determined from the compilation of all of its reviews. With this, we'd be able to calculate the absolute difference between the rating of the queried destination and the top recommendation of the model. The smaller the error, the closer the recommendation would be in quality to the query. We'd then test each model on all 36 destinations and compute the average absolute error to view its final performance.

Final Model Performance



From this chart of our different models' performance, we could confirm our hypothesis that the models trained on preprocessed data would perform better, with the exception of BERT. We see that the top three models were BM25 and TFIDF on preprocessed data and BERT on standard data.

Recommendation and User Application

Ensemble model

Now that we had determined which models were the most reliable for querying with input trip reviews, we could build an ensemble model that would leverage the predictions of our three best models, those being BM25 and TF-IDF on preprocessed data and BERT on standard data.

With this ensemble, we could make a final top 3 most similar destinations prediction based on the user query. The metric used to measure similarity between destinations' reviews is cosine similarity, which will take into account semantic similarities in the reviews.

Integrating Emissions Data

With the ensemble model returning the top 3 most recommended destinations based on review similarity with the user's input, we now just had to integrate the emissions data we fetched from the SNCF database for each trip available.

Each of the top 3 destinations is linked to its train trip distance and carbon emissions data. We can use this to produce an equation that measures the balance between similarity score and emissions of a trip to a certain destination.

Streamlit Application

For our final user application, we use the Streamlit python library, which allows to build a basic and functional app. This app is powered by the ensemble model we've discussed previously. The interface allows the user to input a review or description of a past trip they enjoyed, and it will output the top 3 most recommended destinations, as predicted by the ensemble model, which will each be labeled with their respective distance and emissions data. Additionally, the destination with the best similarity score – emissions balance will be highlighted, which can be considered as the best recommendation.

This interface ultimately allows the user to be recommended several options for travel purely based on the quality of the trip being similar to what they enjoy. Then the ecological aspect is that the user can see directly what the carbon footprint of each trip will be, and which of the three destinations is the most planet-friendly.

Application Examples

In the examples shown below, we see the user interface and the output after a user input. In both examples, we can observe how key words such as "noël", "marchés", "bord de mer", and "lac" were successfully processed by the ensemble model to make appropriate destination recommendations. And at the bottom we see the top recommendation by the app based on eco-friendly goals.

Travel Review Prediction with Ensemble Models

Enter your travel review query:

J'aime beaucoup visiter les marchés de noël.

User input received: J'aime beaucoup visiter les marchés de noël.

Top 3 Predictions:

Destination: Mulhouse

Score: 2.76

Emissions: 1.44 kg CO2

Destination: Metz

Score: 1.51

Emissions: 1.02 kg CO2

Destination: Strasbourg

Score: 1.35

Emissions: 1.31 kg CO2

Best Recommendation based on Score - Emissions:

Best Destination: Mulhouse

Travel Review Prediction with Ensemble Models

Enter your travel review query:

Je voudrai partir au bord de mer ou d'un lac

User input received: Je voudrai partir au bord de mer ou d'un lac

Top 3 Predictions:

Destination: Marseille Saint-Charles

Score: 4.50

Emissions: 2.18 kg CO2

Destination: Lausanne

Score: 1.81

Emissions: 1.63 kg CO2

Destination: Annecy

Score: 1.60

Emissions: 1.58 kg CO2

Best Recommendation based on Score - Emissions:

Best Destination: Marseille Saint-Charles