Nowcasting Socio-Economic Trends: An LSTM-based Approach Leveraging Google Trends, GDELT, and Eurostat Data

Cristina Jiménez, Ayah Dahmani, Ricardo González Otal and Juan Miguel López Piñero

TABLE OF CONTENTS

1.OVERVIEW

This project aims to develop nowcasting models using advanced machine learning techniques, such as random forests, extreme gradient boosting, stacked ensembles, and neural networks. These models will be employed to predict, in real-time, a set of socio-economic variables in relation to labor market integration, providing valuable insights for decision-making processes. Moreover, this project will serve as a tool for new graduates from the Data Science program 2023-2024 of The Bridge School, allowing them to apply and showcase their skills in machine learning and data analysis.

2.INTRODUCTION

In today's rapidly evolving world, the ability to predict and understand socio-economic trends in real-time has become increasingly valuable. This project, created by the company Rambee and assigned to a team of four graduates from The Bridge School, aims to develop a nowcasting model based on machine learning techniques to forecast a specific socio-economic variable using big data obtained from Google Trends, the Global Database on Events, Language, and Tone (GDELT), and Eurostat.

Nowcasting,("now" and "forecasting"), refers to the process of predicting the present or the very near future by analyzing current and historical data. This approach is particularly useful in situations where traditional forecasting methods may not be as effective due to the lack of timely data or the need for more immediate insights.

The main goal of this project is to develop and validate an LSTM neural network nowcasting model for sequential data analysis. The model will utilize three key data sources: Google Trends for search volume trends across countries, GDELT for sentiment indicators and topic popularity from news articles, and Eurostat for official socio-economic statistics from the European Union. By combining these diverse data sources, the model aims to provide insights into consumer behavior, sentiment trends, and socio-economic indicators for forecasting purposes.

3.AIMS AND OBJECTIVES

The primary tasks involved in this project are as follows:

1. Extract information on search volume queries from Google Trends for a predefined set of categories, with weekly frequency, across different countries.
2. Retrieve sentiment indicators and topic popularity rates from GDELT for a series of relevant socio-economic themes, in the form of "Article Tone" and "Topic Popularity Rate" data.
3. Obtain relevant socio-economic data from Eurostat for the variable of interest.
4. Construct and validate an LSTM-based nowcasting model for a specific socio-economic variable of interest, using the extracted data from Google Trends, GDELT, and Eurostat as predictors.

The final nowcasting model will enable a deeper understanding of current trends and facilitate proactive decision-making processes.

4. PYTRENDS API

4.1. API Overview

Google Trends is a powerful tool that provides insights into the popularity of search queries over time. To check the potential of this data, the team turned their focus to the PyTrends API, a Python library that offered an interface for accessing and analyzing Google Trends data.

4.2. Accessing the Gtrends

To create an API that extracts data from the Google Trends platform, we used the unofficial PYTRENDS API as a base. Our goal was to download the data in CSV format and preprocess it for further use. We chose to build the API using FastAPI, a framework we are familiar with thanks to The Bridge.
This API facilitates access to Google Trends data through the unofficial Pytrends API and includes specialized endpoints to extract the average search interest for a defined topic over a specific time period. Initially, we created endpoints with predefined date and country parameters. However, we are considering the option to include these parameters as part of the endpoint calls or the extraction function, allowing for more flexibility in specifying the desired countries or date ranges.

4.3. Challenges and Solutions

We developed two versions of the same API, one with the cleaning function integrated within each endpoint, and another with an independent endpoint for final data cleaning. One of the challenges we faced was ensuring that multiple API calls and corresponding CSV files do not overwrite each other. To address this, we plan to implement a timestamp or a similar mechanism to prevent data overwriting. Additionally, we need to verify that the data we obtain is monthly rather than weekly, as weekly data tends to become null. Furthermore, we implemented a

function to clean the retrieved DataFrame by removing rows with null values and duplicates, ensuring data quality and consistency.
To handle potential errors during the data retrieval process, we created a function that includes a try-except block that retries the request after a brief delay if an error occurs, preventing system saturation and the dreaded 429 error.

However, It is crucial to note that Google Trends imposes rate limits on the number of requests that can be made within a certain time frame. If these limits are exceeded, the Pytrends library may encounter errors or receive incomplete or inaccurate data.
In addition to that, we also encountered further issues while working with the Google Trends API. These issues included inefficiency, with the API functioning only one out of every six attempts, and failures in cases of multiple retries. If more than four retry attempts were made, we had to wait until the following day to try again. Furthermore, we experienced excessively long wait times, requests exceeding the timeout limit, and various HTTP errors (400, 429, 500, etc.). There was also a risk of our IP address being blocked by Google due to excessive requests or potential misuse of the API. These challenges highlighted the limitations and uncertainties associated with relying on third-party APIs, particularly those provided by large tech companies with strict usage policies.

5. GDELT API

5.1. API Overview

The Global Database of Events, Language, and Tone (GDELT) is a massive open-source database that monitors, captures and codes events from news articles and classifies them into a structured database. T
These attributes make GDELT a valuable resource for analysts, and data scientists interested in studying global events. In the following section, we will explore how the team managed to

access the GDELT database, extract relevant data, transform and format it for analysis, and address common challenges encountered while working with this data source.

5.2. Accessing Extracting and Transforming the GDELT Database

The team objective was building an API that could extract and analyze data from GDELT , accessing and working with such a vast and complex dataset has certainly been a challenge. After familiarizing ourselves with the GDELT documentation and understanding the different types of data available, we decided to focus our efforts on extracting sentiment indicators (tone) and topic popularity data. These two aspects would provide valuable insights. We chose to build our API using the FastAPI framework, because it offered an efficient approach. The first step was to set up the project structure and install the necessary dependencies, including the GDELT doc library, which would serve as our gateway to the GDELT database.

With the project setup complete, we began designing the API endpoints. We wanted to provide users with the flexibility to extract data at different time intervals – daily, monthly, and quarterly. This would cater to various use cases and allow for more granular or aggregated analysis as needed. The first endpoint we built was */diary/extraction*, which allowed users to extract daily data for a specific keyword and country. We implemented filters to narrow down the search results based on the provided parameters. Users could also choose to download the extracted data as a CSV file for further analysis or storage.
Next, we tackled the */monthly/extraction and /quarterly/*extraction endpoints. These endpoints would aggregate the daily data into monthly and quarterly summaries, respectively. We used pandas' grouping and aggregation to achieve this, ensuring that the data was correctly grouped and summed based on the specified time intervals. To streamline the data extraction process, we created a */project* endpoint that would automate the extraction and aggregation of data for a predefined list of countries. This endpoint would generate CSV files for each country, containing the tone and popularity data at monthly and quarterly intervals.
As our API grew more complex, we recognized the need for data cleaning and preprocessing. We implemented a /clean and a /EDA ,endpoint that would take the extracted data, handle missing dates, remove duplicates, and ensure that the data was in a consistent format for further analysis. Finally, we added a /mean endpoint that would calculate the mean values for a

specified column across multiple CSV files. This would allow users to easily obtain the average tone or popularity for a given time period, providing a high-level overview of the data.

5.3. Challenges and Solutions

As we progressed, we faced several challenges. During the development of our project, we encountered an issue with the GDELT API, due to an excessive number of requests made by one of our team members, our IP address was temporarily banned from accessing the GDELT API. A formal email was drafted to the GDELT team, explaining the circumstances surrounding the excessive requests and emphasizing the academic nature of our project. We acknowledged the unintentional violation of the API usage guidelines and assured the GDELT owner that appropriate measures had been taken to prevent such incidents from occurring in the future. Throughout the whole development process, we encountered numerous challenges and learned valuable lessons. We had to adapt our approach based on the limitations and quirks of the GDELT API, and we had to be creative in our solutions to overcome obstacles. In the end, we were proud to have built a robust and feature-rich API that could extract, aggregate, clean, and analyze data from the GDELT database. Our API would serve as a valuable tool for researchers, data scientists and analysts

6. Eurostat API

6.1. API Overview

Once again, we chose to build our API using the FastAPI framework, as it had proven to be a reliable and efficient choice for our previous project with the GDELT database. We set up the project structure and installed the necessary dependencies, including the pandas library for data manipulation and the requests library for making HTTP requests.

6.2. Authenticating, Retrieving Socio-Economic Indicators, Data Cleaning and Handling Missing Values

Upon receiving the response from the Eurostat API, we saved the compressed data to a local file and then decompressed it using the gzip module. This decompressed data was then read into a pandas DataFrame for further processing. We realized that the raw data from Eurostat required extensive cleaning and preprocessing. We had to remove unnecessary columns and handle missing values. Additionally, we had to map the time periods from quarters to specific months, as our analysis required monthly data.

Next, we needed to split the data by country and gender, as our analysis required separate datasets for each combination of country and gender. We created a dictionary of DataFrames, with each key representing a country and the corresponding value being a list of two DataFrames (one for males and one for females). Once the data was cleaned and preprocessed, we saved the resulting DataFrames to separate folders for raw, preprocessed, and processed data. This organization would make it easier to track the different stages of data processing and facilitate future analysis or updates.

To further enhance our analysis, we implemented functions to complete missing time series data and generate a combined series representing the European Union as a whole. This would allow us to analyze trends and patterns not only at the country level but also at the broader regional level.

6.3 Challenges and Solutions

One of the challenges we faced was filtering the data based on specific criteria. We wanted to focus on the employment rates of foreign citizens who were actively employed. To achieve this, we applied multiple filters to the DataFrame, selecting only the relevant rows based on citizenship status and employment status.

7. DATA INTEGRATION

7.1 Overview

The objective of the data integration phase was to incorporate the datasets from Gtrends and Gdelt, provided in JSON format, into a single dataframe. After extracting the timestamps and values from the JSON data using regular expressions, each dictionary was converted into a pandas DataFrame. This involved creating an inner dictionary with timestamps as keys and corresponding values. The resulting list of DataFrames, one for each dictionary in the original JSON data, provided a structured and tabular representation of the data. This format facilitated further analysis and manipulation to calculate the mean totals using pandas.
The next critical phase of the project was preparing the data for modeling. Three main objectives were, firstly, to calculate monthly averages for the dataset. These averages would serve as inputs for prediction models. Secondly, calculate the quarterly averages from the data. The third objective was to create a consolidated DataFrame that would serve as the foundation for the nowcasting model.

7.2 Merging Data from Multiple Source

7.3. Feature Engineering and Selection

8. Modeling