Google Places Enricher

A tool that Makes It Easy to Get and Enrich Google Places API Data

Fernanda Regina Gubert
Federal University of Technology - Parana
Curitiba, Brazil
fernandagubert@alunos.utfpr.edu.br

Thiago H Silva
Federal University of Technology - Parana
Curitiba, Brazil
thiagoh@utfpr.edu.br

Abstract

The growing number of available APIs means that more developers and interested users need to learn to use unfamiliar interfaces, requiring a learning curve that can compromise productivity. Thus, it becomes important to find ways to facilitate their usability. This work presents a tool that facilitates using the Google Places API, simplifying multiple API calls to cover a region of interest. In addition, the proposed tool also provides features for the enrichment of these data, extending the PoI data from that region with categories from other sources. It is hoped that developers and users without much computer knowledge can benefit from Google Places Enricher, helping to ease the development of new sophisticated urban applications and services.

Keywords: API, google places, data enrichment, semantic matching

1 Introduction

The use of frameworks and APIs has steadily increased and is becoming indispensable in software development; however, it is known that they are often dificult to learn and use [2] [10] [7] [14]. An API is an interface that transparently provides developers and interested users functionality to perform their tasks or obtain data. APIs provide many benefits, such as code reuse and high-level abstraction, which makes it essential to find ways to facilitate usability [13]. These possibilities can range from well-written documentation to tools that enable learning and using the API.

This work aims to provide a tool that facilitates the use of the Google Places API, simplifying multiple API calls to cover a region of interest. In addition, it also provides enrichment features of these data with extra specific characteristics for the establishments and the correspondence between the categories present in the data with the categories of some other database, extending the PoI data from that region with categories from other sources. In this way, it helps the user to focus their efforts on enabling the development of new and more sophisticated urban applications, such as more detailed market analysis to map the offer of services in a region or to carry out a study on the cultural characteristics of the place.

In: XXI Workshop de Ferramentas e Aplicações (WFA 2022), Curitiba, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2022.

ISSN 2596-1683

2 Background

2.1 The Importance of Tools that Make APIs Easy to Use

The growing number of available APIs implies that more and more developers and users need to learn how to use unknown interfaces. After demonstrating it with exploratory studies, it was possible to identify the existence of difficulties inherent to this process [1]. According to Robillard and DeLine [14], using large APIs requires a learning curve that can decrease developer productivity. After conducting a study with Microsoft developers and surveying the main obstacles faced in this regard, the authors described some factors that can help API learning, such as code examples and clarity in combining API elements with the application scenario of the developer. Tools that facilitate the learning and use of APIs are also cited as an area of research to mitigate these difficulties.

Along these same lines, Hou and Li [6] analyzed the discussions about APIs in groups of developers and identified other obstacles faced. Among them is the lack of clarity, immediately after reading the documentation, on composing calls to the APIs to form the final solution they need, as a translation of the API resources with the sought solution is often required. It is also discussed that tools built to help the use APIs facilitate the extraction of relevant information.

In addition to tools that facilitate the use of an API, many are created to add new features and functionality, even combining with external resources to deliver a result applicable more directly to the real problems and scenarios of developers and users [5] [11]. The tool built in this work brings together both approaches, delivering tasks that facilitate obtaining data from the Google Places API and that promote the enrichment of this data with standard content, but which can still be customized by the user.

2.2 Google Places Enricher Usage Contexts

One of the benefits of this tool is helping to facilitate the process of getting data from the Google Places API, considering both the task that assists in the calculation of geographic coordinates, given an area and a radius, as well as the process of multiple calls to the API, with the paging management and handling of returned data. In this way, Google Places Enricher can be used by any project interested in the data of this API, for example, in the works of Satman and Altunbey [15], and Sen and Quercia [16].

The work of Falher, Gionis, and Mathioudakis [8] address the problem of comparing neighborhoods between cities. They used geolocated data from Foursquare in cities in Europe and the USA. The authors proposed a methodology to describe neighborhoods in terms of the activities that occur in them, using data from establishments. The Google Places API has a significant worldwide coverage [3] and provides data with good quality, which makes the proposed tool a good option for works that need to use data of establishments independent of the region, such as the one mentioned above, providing the consumption of this data in a simple way and focusing the researcher's efforts on other tasks.

Finally, the enrichment of the data provided by Google Places Enricher can help different studies interested in data of establishments with detailed categories, such as the work by Martí et al. [9]. Promoting the debate on the spatial definition of neighbourhood boundaries, the authors conducted a study using Google Places data. One of the challenges was the recategorization of the data to ensure a more detailed analysis. The tool proposed here makes it possible to categorize data, providing standard entries that can be customized according to the user's needs.

3 Tool Description

The Google Places API is a service that returns geolocated data of establishments and points of interest – for simplicity, it will be used only the term establishments to refer to both. In addition to the latitude and longitude of establishments, they are associated with at least one category to describe their type – in total, there are 141 categories. However, these categories do not have the level of specificity required for certain purposes. For example, the API provides the restaurant category to establishments that classify themselves as such but does not provide a more specific category on the gastronomic type, such as Italian or Japanese.

The Places API provides different options to perform requests; this tool specifically uses the Place Search option with the Nearby Search component [4]. The optional keyword parameter was included in the calls to enrich the data. The Google Places service searches the text of this parameter for all the indexed content, returning the establishments ordered based on the perceived relevance. Even though it is not a specific parameter for searching for types of establishments, the API documentation guarantees to return valid results if the entries are a place name, address, or category of establishments, thus making it a convenient option for the proposal of data enrichment.

Google Places Enricher is a library, and the parameters must be entered manually according to the users' interests. The tool provides three different tasks: Coordinates Calculation, Retrieve Google Places Data, and Matching Categories, which are described below. Figure 1 helps to understand and organize the requirements for the execution of each task, also denoting the expected output for each of them.

3.1 Coordinates Calculation

This task generates a CSV file with the geographic coordinates (latitude and longitude) that compose a rectangular area according to a predetermined step in meters. First, the user needs to inform the northeast and southwest extremes, in terms of latitude and longitude, to delimit the rectangle that encompasses the area of interest. For example, Figure 2 (left) shows the rectangular area that could be used to retrieve data for the city of Toronto – the points that should be informed are highlighted in the figure. Next, it is necessary to specify the radius in meters (parameter *RADIUS*) that correspond to the range of each of the sub-areas that are calculated. Figure 2 (right) illustrates the sub-areas with a particular radius.

The process for calculating sub-areas starts from the southwest extreme of the rectangular area, to which $3/4 \times RADIUS$ is added horizontally and vertically to define the center of the first sub-area. To create the following sub-areas, a step corresponding to $(1 + 1/2)RADIUS$ is calculated; this step is used as the distance between the coordinates, both vertically and horizontally. That is, starting from the first defined coordinate, a vertical step is added to define the next one. This is done until the upper limit of the rectangular area is exceeded. When this happens, a horizontal step is added, and the process is repeated, from bottom to up in the rectangular area, and ends when the right side boundary of the rectangular area is exceeded. These values calculated from the radius were defined to cover the entire area of interest. As it can be seen in Figure 2 (right), the sub-areas overlap, but this is handled when performing API calls, removing duplicate establishments.

This task helps determine the geographic coordinates that will be used later in API requests, facilitating the composition of this parameter. It can also collaborate with the decision-making on the radius size to use due to the number of coordinates needed to cover the entire area of interest. And by retrieving the cost per request, the user can predict the total cost. However, depending on the application scenario and the number of coordinates, a manual intervention to adjust the coordinates may be interesting, ensuring a more adequate coverage of the area. That is, this task helps the process of generating the coordinates, but it may not be the best solution for all cases.

3.2 Retrieve Google Places Data

This task performs requests to the Google Places API according to the geographic coordinates defined in the input file and a predetermined radius. It retrieves places information according to the categories defined in the input file. These categories are used in the API keyword parameter, and each

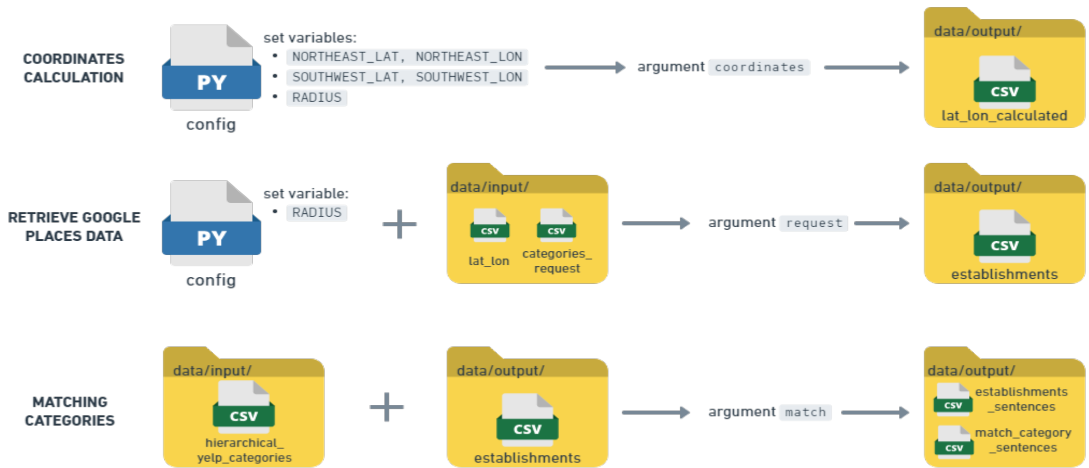

Figure 1. Input requirements and expected outputs for each task.

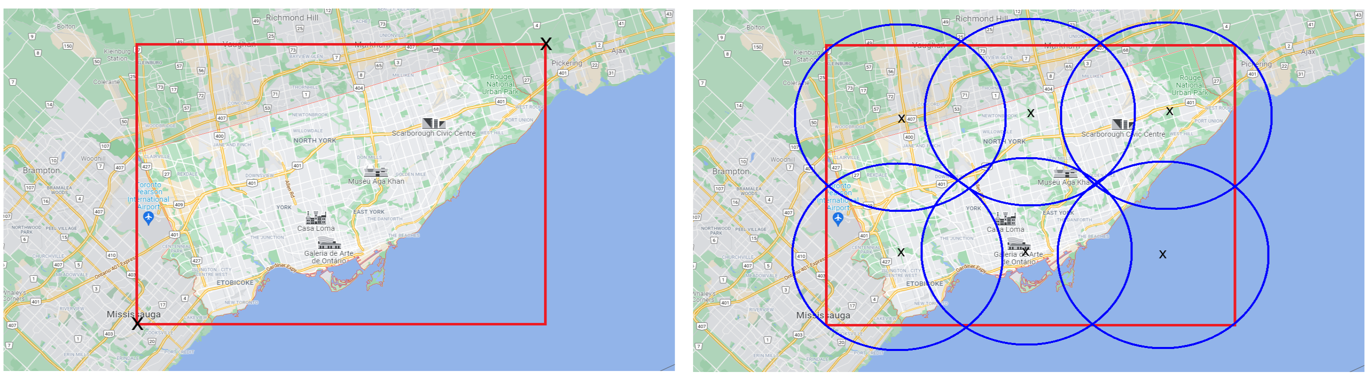

Figure 2. Rectangular area delimited for the city of Toronto (left). Examples of sub-areas considered by the proposed tool (right).

request corresponds to a coordinate and a category; that is, the number of requests for each coordinate corresponds to the number of categories. In addition, the data returned by the API is processed, making them formatted in a structured way in a CSV file.

The default content of the category list consists of the fourth-level categories of Yelp's hierarchical base; however, any categories are accepted. These categories are linked to the establishment data, assigning more characteristics to them, and according to the user's needs. But if there is no interest in enriching the data, the researcher only needs to leave the input file empty.

3.3 Matching Categories

The objective of this task is to perform a correspondence between the categories present in the data of establishments with the categories of some other database that is of interest; by default, Yelp was used. To increase the semantic capacity and consequently the mapping accuracy, sentences were created for each establishment. These sentences were composed of a Yelp category with all the existing Google categories for that establishment. That is, if an establishment has the categories Amusement Parks and Water Parks from Yelp, and the categories Park and Tourist Attraction from Google, the sentences are:

- Amusement Parks Park Tourist Attraction
- Water Parks Park Tourist Attraction

The data of the establishments with their respective sentences are exported in a CSV file in case the user wants to combine them later with the Yelp sentences. As stated before, the Yelp categories are arranged in a 4-level hierarchical structure. With the same intention of increasing the semantic capacity, the Yelp sentences were created using all levels; that is, for each category of the last level, the associated sentence consists of all categories all the way from the first level.

The mapping process was carried out with a structure called Sentence Transformers, which calculates embeddings from sentences and texts, and can be compared to find semantic similarities between the sentences [12]. Several pretrained models with a large and diverse dataset of more than 1 billion training pairs are available, and the model choice was empirical. To compare the generated embeddings, the cosine similarity was calculated, and for each sentence related to the establishments, the Yelp sentence with the highest score was retrieved. These matches with their respective scores are also available in a CSV file. It is important to emphasize that a validation of the scores is necessary to consider whether the sentences present high similarity since the tool applies no limit regarding the score. Note that other categories besides Yelp could be used, although they are not available by default in the tool as of the date of this publication.

3.4 Privacy, License, and Access

This work describes an academic tool at the technological and scientific level, which can be implemented without territorial limitations, for private use (individual or legal entity), which does not own the data collected and is not responsible for its use.

The tool is open source licensed under the terms of the GNU General Public License, which means that the software can be used, modified and/or shared under the specified conditions [1]. The repository address [2] has also been made available, as well as a practical video with the demonstration of the tool [3].

4 Final Considerations

This work presents a tool that facilitates the use of the Google Places API, from the determination of mandatory parameters and the execution of requests to the treatment of the obtained data, making them available in a structured way. The tasks of the Google Places Enricher tool also provide enrichment of this data, such as the correspondence between the categories present in the data with the categories of any other database of interest. These last tasks generally use standard content but can be customized according to the user's interest.

With this tool, it is expected that both developers and users without much computer knowledge will be able to easily collect Google Places data and make the described adjustments to enrich them and obtain correspondence with other databases according to their needs. Thus, developers can focus their efforts on their application scenario, as the tool already translates the API resources with the solution sought, not interfering with their productivity, which would have a learning curve to perform this task. In addition, it also allows users from other areas to work with this data more independently. In future work, the tool can provide different options for a list of categories and bases for tasks related to data enrichment so that the user can choose the one that adds more value to their application.

[1]https://www.gnu.org/licenses/gpl-3.0.pt-br.html.
[2]https://github.com/FerGubert/google_places_enricher.
[3]https://youtu.be/sQhkDPcFIko.

Acknowledgements

All stages of this study were financed in part by CAPES - Finance Code 001, project GoodWeb (Grant 2018/23011-1 from São Paulo Research Foundation - FAPESP), and CNPq (grant 310998/2020-4).

References

[1] Ekwa Duala-Ekoko and Martin P Robillard. 2012. Asking and answering questions about unfamiliar APIs: An exploratory study. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 266–276.
[2] Gerhard Fischer. 1987. Cognitive view of reuse and redesign. *IEEE Software* 4, 4 (1987), 60.
[3] Google 2021. *Detalhes de cobertura da Plataforma Google Maps*. Google.
[4] Google 2022. *Nearby Search*. Google.
[5] Yoan Gutiérrez, Sonia Vázquez, and Andrés Montoyo. 2016. A semantic framework for textual data enrichment. *Expert Systems with Applications* 57 (2016), 248–269.
[6] Daqing Hou and Lin Li. 2011. Obstacles in using frameworks and APIs: An exploratory study of programmers' newsgroup discussions. In *2011 IEEE 19th International Conference on Program Comprehension*. IEEE, 91–100.
[7] Amy J Ko, Brad A Myers, and Htet Htet Aung. 2004. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE, 199–206.
[8] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. 2015. Where is the Soho of Rome? Measures and algorithms for finding similar neighborhoods in cities. In *Ninth International AAAI Conference on Web and Social Media*.
[9] Pablo Martí, Leticia Serrano-Estrada, Almudena Nolasco-Cirugeda, and Jesús López Baeza. 2021. Revisiting the Spatial Definition of Neighborhood Boundaries: Functional Clusters versus Administrative Neighborhoods. *Journal of Urban Technology* (2021), 1–22.
[10] Samuel G McLellan, Alvin W Roesler, Joseph T Tempest, and Clay I Spinuzzi. 1998. Building more usable APIs. *IEEE software* 15, 3 (1998), 78–86.
[11] Maria N. Pavlova and Asen Alexandrov. 2018. GLOBDEF: a framework for dynamic pipelines of semantic data enrichment tools. In *Research Conference on Metadata and Semantics Research*. Springer, 159–168.
[12] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084
[13] Martin P Robillard. 2009. What makes APIs hard to learn? Answers from developers. *IEEE software* 26, 6 (2009), 27–34.
[14] Martin P Robillard and Robert DeLine. 2011. A field study of API learning obstacles. *Empirical Software Engineering* 16, 6 (2011), 703–732.
[15] Mehmet H. Satman and Mustafa Altunbey. 2014. Selecting Location of Retail Stores Using Artificial Neural Networks and Google Places API. *International journal of Statistics and Probability* 3, 1 (2014), 67.
[16] Rijurekha Sen and Daniele Quercia. 2018. World wide spatial capital. *PloS one* 13, 2 (2018), e0190346.