Syntactic and Semantic Analysis of REST, and GraphQL APIs to Assess and Compare their Linguistic

Krishno Dey[1], **Hung Cao**[2], **Francis Palma**[1]
[1] *SE+AI Research Lab, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada*
[2] *AELab, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada,*
{krishno.dey, hcao3, francis.palma}@unb.ca

Short Abstract

Application Programming Interface(API) serves as the bridge that enables communication and data exchange between different software systems. A well-designed API is assumed to have **linguistic patterns (good design practices)** and a poorly designed API is assumed to have **antipatterns (poor design practices)**. The representational state transfer (REST), and GraphQL architecture are the most popular approaches to developing web APIs. To assess the linguistic design of APIs, we implemented detection algorithms for **ten (anti)patterns** employing the **detection heuristics**. Detection algorithms rely on **syntactic and semantic analysis** to assess the **linguistic quality** of APIs. We conducted our study on 33 APIs (21 **REST,** 12 **GraphQL**) on a set of 1655 endpoints. Our detection algorithms detected linguistics (anti)patterns with an average **accuracy of** 93%, a significant improvement from state-of-the-art studies.

Application Programming Interfaces (APIs)

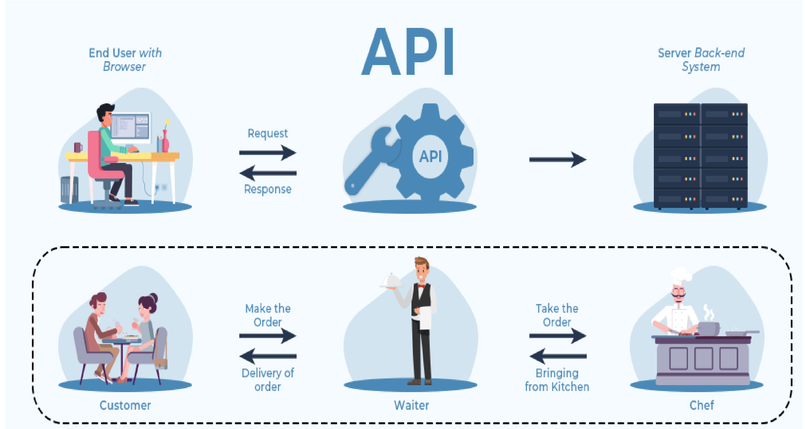

Figure 1. Working principle of Application Programming Interfaces (APIs). Source: (geeksforgeeks.org)

Conclusion

- Our detection algorithms yield better detection performance compared to the state-of-the-art methods.
- Our findings confirmed linguistic antipatterns exist in both REST and GraphQL APIs.
- We observed that both REST and GraphQL APIs are prone to linguistic antipatterns, with REST having slightly more antipatterns.
- The most commonly occurring linguistic antipatterns are *Unversioned Endpoint*, *Amorphous Endpoint*, and *Pluralized Nodes* regardless of the API category.
- We also plan to improve the detection performance and analyze more APIs and endpoints of other API categories such as gRPC, OData, and GData, to investigate their linguistic design quality.

Proposed Methodology

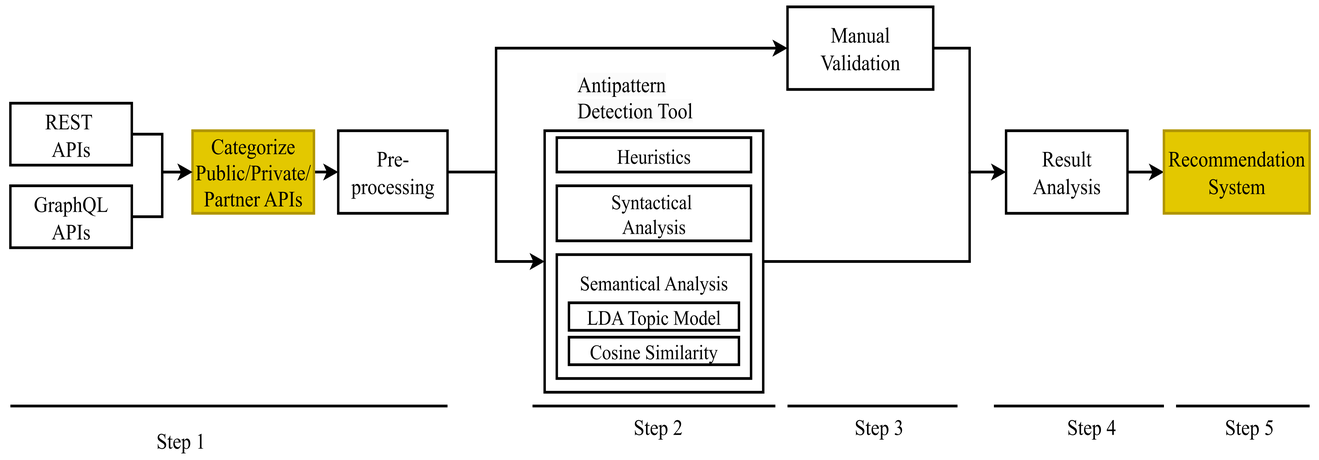

Figure 2. Proposed Methodology. White rectangular boxes represent the steps that are completed and yellow rectangular boxes represent steps that are to be completed in the future.

Contributions and Results Discussion

- We analyze and present a dataset consisting of 1655 endpoints from 33 REST and GraphQLAPIs.
- Poor linguistic design (anti)patterns are present in REST and GraphQL APIs, i.e., despite the wide adoption of REST and GraphQL APIs, they still lack quality design.
- Our detection algorithms achieved an average accuracy of 93.08%, precision of 79.9%, recall of 86.59%, and F1-score of 85.98%.
- Antipatterns are more prevalent in REST compared to GraphQL APIs, i.e., GraphQL APIs are well designed compared to REST APIs in terms of linguistic quality, although the margin of difference is very small.

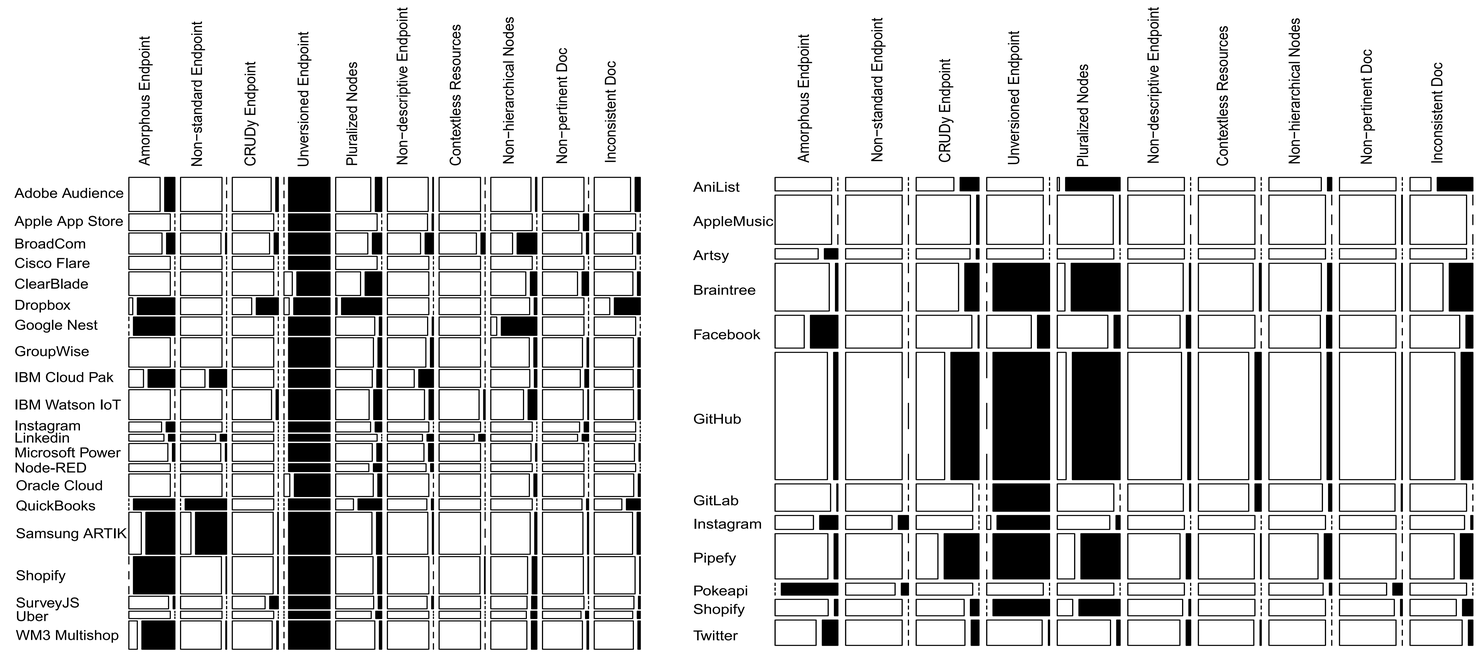

Figure 3. Detection of (Anti)Patterns in REST APIs (left) and GraphQL APIs (right). The black portion represents antipatterns and the white portion represents patterns.