BDM - P2: Formatted and Exploitation Zones

Jose Carlos Lozano Dibildox
Jakub Kwiatkowski
Nils van Es Ostos

1 Introduction

Despite the numerous study programs and scholarships applicable to them, accessing information about the matches available for potential candidates is quite challenging. To address this, we are developing a platform that offers personalized recommendations for degrees - bachelor's and master's programs, and scholarships.

During the previous project, we created data collectors and stored gathered files in the Landing Zone. This document presents the following steps taken: integration of given data, enrichment of the dataset, development of recommendation system, and usage of ML models to infer deeper insight into courses offered on our platform. We will also present the final architecture of our product, from collecting the data to delivering its final form on the webpage.

Additionally, all the information regarding the procedure can be found in github[1].

2 Data integration

Disclaimer: the sections 2, 4 and 5 can be found in identical wording in both of the reports of the group. This is because the data integration and the final architecture was done collectively, and the descriptive and predictive analysis was managed by the extended team. Regarding data integration, all the data from various sources was combined and loaded into Spark. Then, the data was cleaned and transformed into the features that were used in the final version.

Among the different transformations made, it is important to pay special attention to the generation of both comments and users in the data. Unfortunately, it was not possible to find many reviews of certain programs. Hence, some were scrapped from the various sources but the majority was randomly generated. The main goal of this is to create a rich dataset that will serve as a base for the analysis that will be better ran later on.

Once all the data was ready, dataframes were filtered and joined, turning into the final database that was loaded into the graph.

3 Distributed machine learning and real-time data prediction

In this section, we will discuss the different algorithms developed for adding new features to our initial data. Among the features added, we can find a recommendation system for programs and a classification between positive and negative reviews of programs.

3.1 Recommendation System

In order to build a recommendation system, first we tokenized the program titles and removed the *stop words*, that is, the set of commonly used words in any language that do not add any value. Once the relevant words were outlined, we created a *cleaned title*, composed only with relevant information on the programs.

After, we turned those *cleaned titles* into *TF-IDF vectors*, that is, vectors with weights were assigned to each of the words on the title depending on their importance on the dataset. By doing this, we were able to compute the similarity between different program titles. Notice, that we created 5 recommendations for each program using the method described. However, one of the recommendations is always going to be the program itself, since the similarity between a program and itself is 1. This leaves us with 4 recommendations for every program.

3.2 Binary Classification

For this section, we classified reviews as good and bad experiences with the program. In order to do this, we started extracting the embeddings for the different reviews on the dataset. Notice that the embeddings were extracted using a pre-trained model.

After extracting the embeddings, arbitrary good and bad reviews were defined, and a default good and bad review vector was created. In this high-dimensional space, we defined reviews as good or bad depending on the distance to these arbitrary vectors.

In Fig(1), the results of the classification can be found. Notice, that most of the reviews are defined as good, that is because the dataset contains many more positive reviews than negative ones.

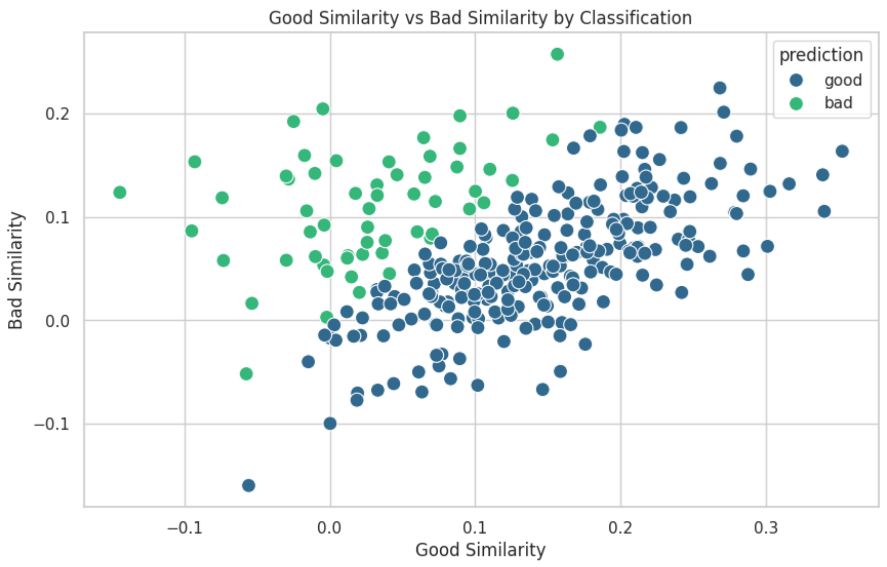


Figure 1: Classification of reviews by positive/negative comments.

Model Validation

In order to measure the accuracy of the classifier, it is important to mention part of the reviews were randomly generated. Hence, the generated reviews were easily labelled. By comparing the labels for these comments and the predictions made by the binary classifier, we were able to compute the accuracy model: 0.831.

4 Final Architecture

The final flow of the data can be described by the following steps:

1. Multiple data files from different sources are gathered by the collectors.
2. The data gathered by the collectors is stored in the HDFS.
3. Single data rows are processed and connections between different sources are created.
4. New information on courses is inferred by classification & clustering methods.
5. Recommendations for each course are selected with NLP analysis.
6. All the data is stored in Delta Lake.

7. TBOX and ABOX for the graph database are being created based on integrated sources.
8. The backend connects to the database and queries it, filtering by given attribute values.
9. Queried information is displayed on the webpage.

In Fig.(2), this pipeline described is represented as the final architecture.

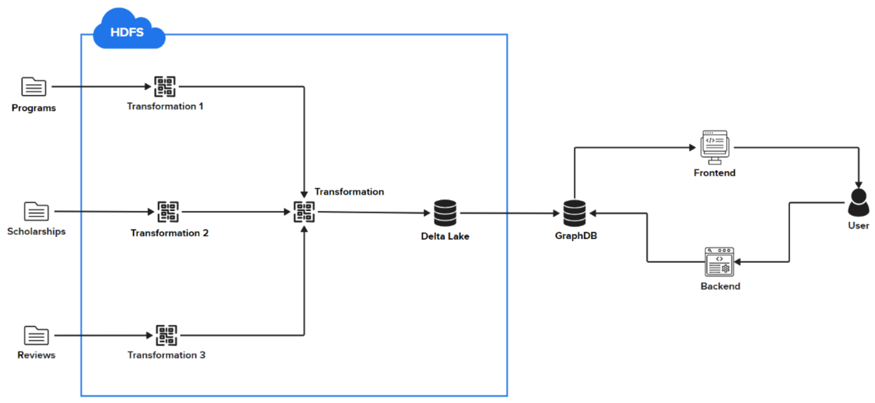
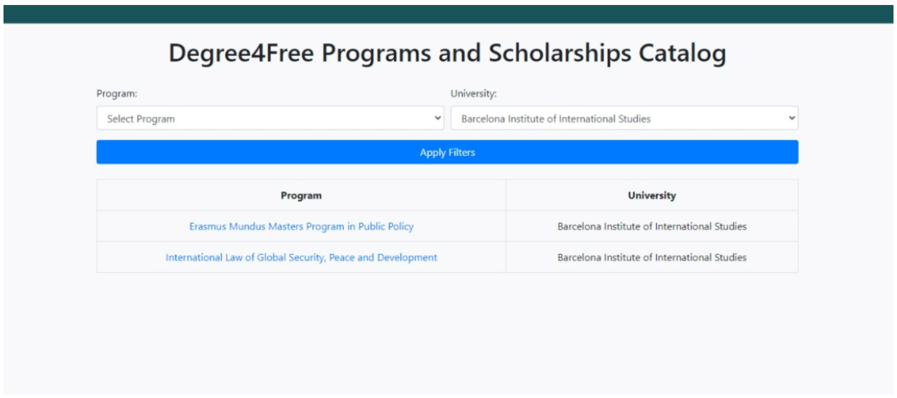


Figure 2: General architecture of the pipeline.

The final part was created with Python, using SPARQLWrapper for connecting to the GraphDB database and executing queries, as well as Jinja2 for conditional logic of some components and iteration over data. For HTTP requests, template rendering, and handling different path routing we leveraged Flask. In the frontend there are technologies such as Bootstrap, jQuery, and Popper.js. Thanks to them we could manage pre-designed components, tooltips, and overall deal with page responsiveness.

The example of the final view can be seen in the figure below.



Degree4Free Programs and Scholarships Catalog

Program	University
Erasmus Mundus Masters Program in Public Policy	Barcelona Institute of International Studies
International Law of Global Security, Peace and Development	Barcelona Institute of International Studies

Figure 3: Example view of the data presented on the final webpage.

5 Future Challenges

We can already define the next improvements that we would apply to the prototype. First of all, we could add data from other countries. That would lead to much more data on degrees or scholarships. Other countries lead to more prerequisites, like other language requirements, different grading systems, and funding types.

Adding an extended user profile could help improve the recommendation system by adjusting to all the information uploaded by the user. More user data could be followed by extending universities' requirements - universities could define their ideal candidate silhouette. Based on both, our system could count a score of how well the user profile matches university requirements.

In the future, our analyses could become even more precise. As we collect user data over time, we can analyze the historical data of students who have been accepted into various programs in previous years. This analysis would allow us to identify the actual candidate profiles that programs tend to accept, as opposed to the ideal profiles universities would advertise. By incorporating these insights into our predictive algorithms, we can provide new users with more accurate and actionable advice. To achieve this, we would need to continuously evolve and manage our data storage systems to handle the increasing volume and complexity of the historical data.

References

[1] Nilsvanesostos. (s. f.-a). Degree4Free/Spark at main · Nilsvanesostos/Degree4Free. GitHub. https://github.com/Nilsvanesostos/Degree4Free/tree/main/Spark