ICT-enabled
integration facilitator
and life rebuilding guidance
Re_Build

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

REBUILD
ICT-enabled integration facilitator and life rebuilding guidance
Project start date: 01/01/2019 | Duration: 36 months

Deliverable: D3.5 Skills and needs matching for migrant guidance

DUE DATE OF THE DELIVERABLE: 28-02-2021
ACTUAL SUBMISSION DATE: 16-02-2021

Project	REBUILD – ICT-enabled integration facilitator and life rebuilding guidance
Call ID	H2020-SC6-MIGRATION-2018-2019-2020 – DT-MIGRATION-06-2018
Work Package	*WP3 – Data Analysis and skills matching*
Work Package Leader	*Universidad Politécnica De Madrid UPM*
Deliverable Leader	*Universidad Politécnica De Madrid UPM*
Deliverable coordinator	Silvia Uribe sum@gatv.ssr.upm.es
Deliverable Nature	Demonstrator
Dissemination level	Public (PU)
Version	0.6
Revision	Final

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

DOCUMENT INFO

AUTHORS

Author name	Organization	E-Mail
David Martín Gutierrez	UPM	dmz@gatv.ssr.upm.es
Silvia Uribe Mayoral	UPM	sum@gatv.ssr.upm.es

DOCUMENT HISTORY

Version #	Author name	Date	Changes
0.1	David Martín	15-01-2021	Starting version
0.2	David Martín	18-01-2021	Matching methodology
0.3	Silvia Uribe	01-02-2021	Executive summary, state of the art
0.4	Silvia Uribe	03-02-2021	Final version to be reviewed
0.5	Panagiotis Stalidis	05-02-2021	First revision
0.6	Silvia Uribe	16-02-2021	Final version after peer-review

DOCUMENT DATA

Keywords	D3.5 Skills and needs matching for migrant guidance
Editor Address data	Name: David Martín, Silvia Uribe Partner: UPM Address: Av Ramiro de Maeztu 7 Phone: +34 91 464160 ext. 142 Email: [dmz,sum]@gatv.ssr.upm.es
Delivery Date	28-02-2021
Peer Review	*Dimitris Demertzis (OMNES), Luigi Laura (UNINETTUNO)*

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

EXECUTIVE SUMMARY

This deliverable is presenting the matching engine implemented in REBUILD project's WP3 based on the previously designed skill-matching methodology from D3.2, which is focused on the application of different artificial intelligence techniques to the user's profile defined in Task 3.1.

As it was explained in the previous version of this deliverable, this module aims at providing an end-2-end matching based on two main inputs, that is, on the user profile on one hand and on the description of the different elements to be recommended on the other. This matching will be considered as a recommendation itself, so the recommendation engine from T3.3 directly obtains this results by means of a specific API.

The implementation of this module has been modified according to the requirements of the different use cases that have been defined, providing a complete user-driven solution for the job seeking, educational training and social mentoring scenarios. In this regard, the use of the European Skills, Competences and Occupation Ontology (ESCO), as mentioned before, is an advantage when trying to normalize the entire process with a common understanding.

The document is organised as follows: The first section makes an introduction to this document. The second section is devoted to make a final review of the state of the art relating the matching algorithms finally used in this implementation. Third section is intended to describe the matching technology that has been implemented, including the details for the different use cases deployment. Finally, information about the software releases are given in section four and section five includes the conclusions.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

TABLE OF CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

INDEX OF FIGURES

INDEX OF EQUATIONS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

1 INTRODUCTION

REBUILD project's objective is to provide a toolbox of ICT-based solutions that will help to the smooth integration of refugees and migrants. REBUILD refers both to the facilitation of the local authorities' management procedures and to the migrants' life quality improvement. To achieve these goals, REBUILD is designed as a user-centered application that attempts to recognize users' needs and give them personalized recommendations and targeted solutions. To assess this purpose, personal information for each migrant is required in order to learn profile patterns and link to needs and resources. For the gathering of those necessary data, all users will have to consent in order to provide anonymized, GDPR-compliant information that will be used by AI-based methods. In particular, the proposed technological solutions include an AI-based profile analysis to enable the personalized support, an AI-based matching tool in order the migrants' needs and skills to be matched with services provided by local authorities in each pilot country and a set of tools such as a chatbot or audio visual communication to enable personalized two-way, effective communication between the final users, i.e. migrants and local service providers.

More specifically, this project follows a user-centered and participated design approach, aiming at addressing properly real target users' needs, ethical and cross-cultural dimensions, and at monitoring and validating the socio-economic impact of the proposed solution. Both target groups (immigrants/refugees and local public services providers) will be part of a continuous design process; users and stakeholders' engagement is a key success factor addressed both in the Consortium composition and in its capacity to engage relevant stakeholders external to the project. Users will be engaged since the beginning of the project through interviews and focus groups; then will be part of the application design, participating in three Co-Creation Workshops organized in the three main piloting countries: Italy, Spain and Greece, chosen for their being the "access gates" to Europe for main immigration routes. Then again, in the 2nd and 3rd years of the project, users' engagement in Test and Piloting events in the three target countries, will help the Consortium fine-tuning the REBUILD ICT toolbox before the end of the project.

The key points regarding technology solutions proposed are:

- GDPR-compliant migrants' integration related background information gathering with user consent and anonymization of personal information;
- AI-based profile analysis to enable both personalized support and policy making on migration-related issues;
- AI-based needs matching tool, to match migrant needs and skills with services provided by local authorities in EU countries and labour market needs at local and regional level;
- a Digital Companion for migrants enabling personalized two-way communication using chatbots to provide them smart support for easy access to local services (training, health, employment, welfare, etc.) and assessment of the level of integration and understanding of the new society, while providing to local authorities data-driven, easy to use decision supporting tools for enhancing capacities and effectiveness in service provision.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

As it was mentioned above, this deliverable is focused on the AI-based tools that are provided within the project in order to improve the Quality of Experience (QoE) of users.

2 SKILL MATCHING: BACKGROUND AND STATE OF THE ART

2.1 STATE OF THE ART IN SKILL-MATCHING

As explained in D3.2, skill-matching is a wide branch of research in recent years where multiple approaches are being followed. Different methodologies were deeply explained since they were used for the first approach of the matching engine, but the evolution not only of the needs and requirements, but also of the research area itself, made necessary a complete revision of the state of the art again.

According to the current status of the research, and considering REBUILD scenario, where both users and items share the same feature, there are two innovative ways to perform the matching that we are looking for:

- The use of Transformers (Vaswani et al.,2017), which are mainly focused on items characterization by means of generating different embeddings to be used when calculating the matching score.
- To directly use different statistical methods such as Jaccard similarity (Halkidi et al., 2002) or fuzzy matching (Sloan, 2018) to obtain the same score.

Transformer architectures are currently applied in so different areas. In this regard, as explained in (Woolf, 2019, 2020), Natural Language Processing (NLP) is one of the most benefited research areas since Transformers have facilitated building efficient models and high capacity pretraining methods. Moreover, as an open-source library provided by a unified API, its application is spreading among the AI community.

Nevertheless, as a feed-forward sequence model, Transformers fail to generalize in some simple tasks that other methods such as recurrent models handle easily. For this reason, some current researches try to solve this main disadvantage. This is the case of the work presented in (Dehghani, 2019), where authors propose the Universal Transformer (UT), a short of parallel-in-time generalization of the initial model for addressing this issue. Moreover, with regards to its quadratic complexity, researches such as (Katharopoulos, 2020) tries to solve their slow response by reducing it to a linear one achieving similar performance.

With regards to the fuzzy matching, this is a solution based on a typical statistical method applied to this field that provides very promising results. Different research areas such as the spam detection on social networks (Kumar, 2019), similarity join (Wang, 2019) and even string matching (Abrahamson, 1987) apply different approach of this method.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

3 MATCHING METHODOLOGY FOR MIGRANT GUIDANCE

The goal of this section is to describe the main developments that have been implemented in order to provide final users (in this case migrants, refugees and asylum seekers) with meaningful notifications which are collected using the matching criteria that is going to be presented in this specific section.

Firstly, we will present an overview of the pipeline that is followed in order to have a general picture of how this component interacts within the rest of the application of the system. Then, an explanation regarding embedding generation will be introduced to add some guidelines regarding the collection of potential vectors that can be used later on for the matching analysis. Finally, a summary of the matching algorithms will be presented throughout examples and use cases.

3.1 PIPELINE OVERVIEW

This initial section is devoted to describing how the matching interacts within the rest of the components of the system, especially with the ones involved in WP3 and WP5.

Fig. 1 shows the pipeline that is driven to retrieve the final recommendations of the system once the matching has been applied. In particular, the following stages are needed:

I. The user enters in the REBUILD APP and selects a specific domain, where here, a domain corresponds to the use cases that were designed by the Consortium of the project including health, legal, social life, job seeking, etc.
II. The recommender API calls the matching service for such a specific domain.
III. The matching component will require a set of features for the selected domain in order to produce its outcome. More specifically, it will need to retrieve the information of both users and items from the database. In this case items are the contents that are available in the platform so that they may be jobs, mentors and courses among others.
IV. Then, the matching generates embeddings for both users and items. To do so, it will call the user profiling component and the item profiling function.
V. Finally, the matching component will compute its procedures to generate the outcomes that will be forwarded to the recommender system API. More specifically, this set of outcomes will include identifiers of the items that will be provided to the end-user.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

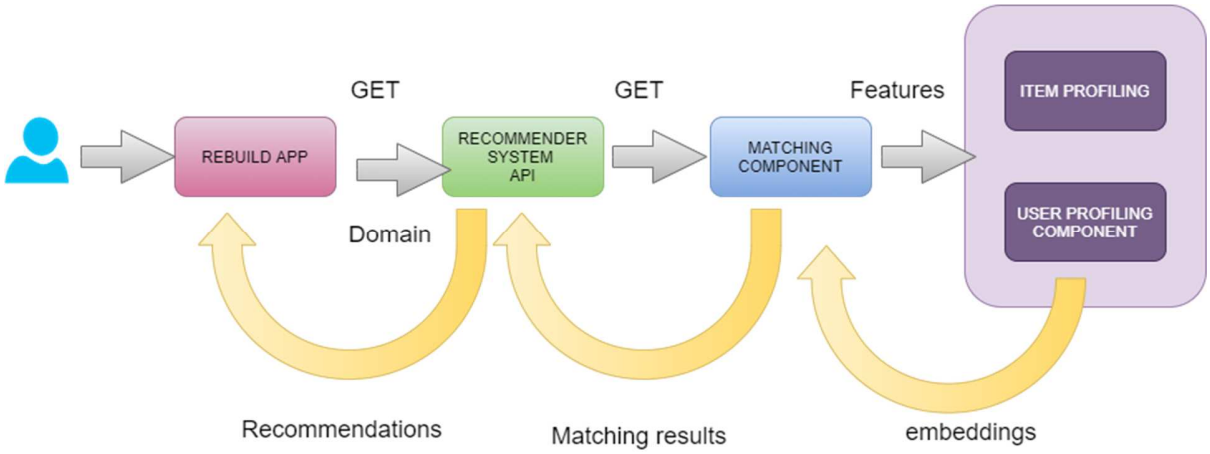

Fig. 1. Different stages of the pipeline to perform the matching

To illustrate the procedure of the matching from a data-flow perspective Fig. 2 shows the inputs that are provided to the component, its interaction with the database of the REBUILD project to retrieve the information needed for the analysis regarding the domain, and finally, the collection of items that are produced after the analysis. This set of results will contain the identifiers of the items to be later on provided via the User Interface (UI).

Moreover, all the information that is needed for the matching is kindly asked to the end-user by the chatbot which is the bridge element between the components of WP4 and WP3.

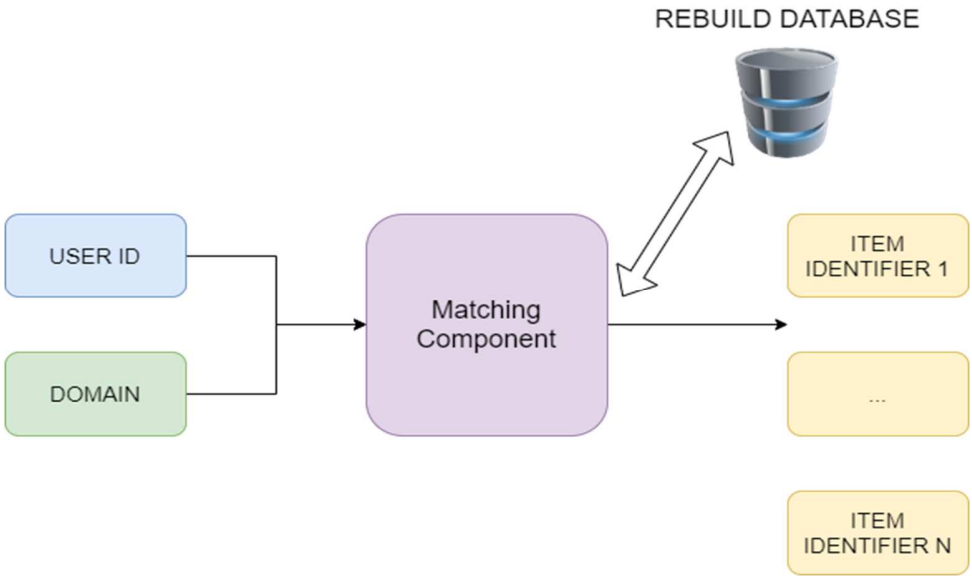

Fig. 2. Inputs and outputs that are provided by the matching component.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

3.2 GENERATING EMBEDDINGS FOR USERS AND ITEMS

This section is devoted to explaining how the embeddings for both users and items are generated to then perform the matching. As a brief summary, an embedding consists in a low-dimensional vector representation of an input element which provides a better level of compression. This fact makes embeddings better since similarities between pairs of them can be conducted in a fasted manner due to the low-dimensionality of the information. Therefore, they are being used in many applications related to information retrieval and recommender systems. More specifically, embeddings are usually produced via intermediate layers of powerful Deep Learning Models once they have been properly trained. In particular, we have taken advantage of the latest state-of-the-art models denoted as Transformers in order to produce these representations.

Since D3.4: *User's profile clusterization* includes a specific section to explain how to produce user embeddings is included, so that this deliverable only provides a general overview of how this process is followed to support the general understanding of the document. Fig. 3 represents the phases that are needed to produce the embedding vector.

As expected, depending on the domain, the input features may differ. For instance, to produce a matching between users and jobs, the features that have in common are the set of skills needed for the job or the previous experience. On the other hand, when the domain is the social life, the features are related with hobbies, interests or educational experience. Thus, a preprocessing of the features is needed to properly provide the input to the model that will produce the final embedding.

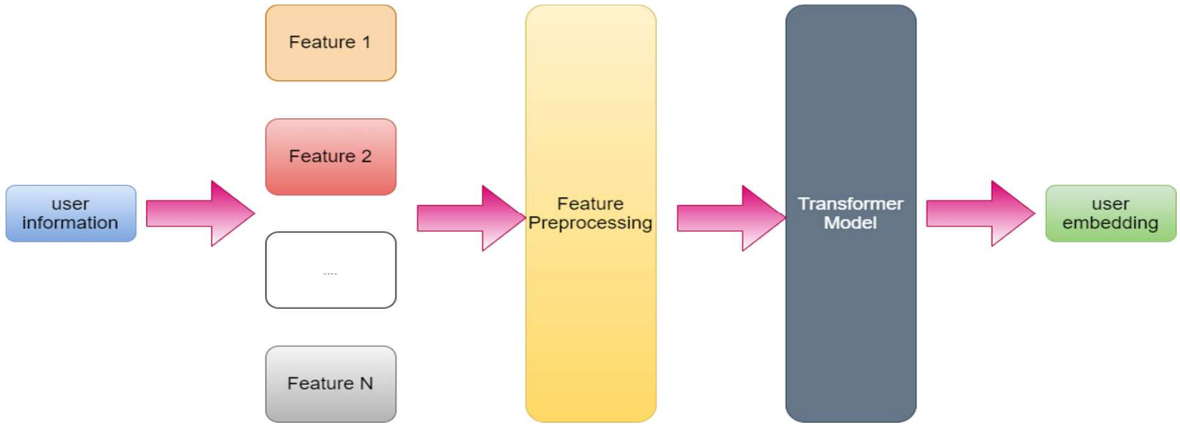

Fig. 3. Stages needed to generate the final user embedding (from D3.4)

As it is remarked in D3.4, one of the main benefits when using Transformers for extracting embeddings from text-based features lies in the management of different languages. In particular, Transformers have been trained using datasets from different languages with the goal of finding similarities among words that have the same meaning in different languages. This fact provokes that these models will

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

produce vectors that will be closed in case the meaning is similar and far otherwise. Hence, they can be used for tasks such as the one presented in this document. The following subsections will describe the cases that determine how to proceed when generating potential embeddings.

Finally, in order to generate the embeddings throughout Transformers, two main resources have been employed:

- The so-called Flair[1] library which is a Python package that provides multiple procedures and models for solving many Natural Language Processing downstream tasks.
- The Hugging Face[2] repository to investigate and retrieve the information of the different Transformer models that better fit with the presented application.

3.2.1 Users and Items share the same Features

It is important to emphasize that this approach is only employed whether users and items have some common features. This scenario is very well illustrated in the previous version of this deliverable: D3.2, where a first approach conducted throughout Graph Theory methodologies is described. Fig. 4, which was precisely extracted from that deliverable, illustrates the scenario where users and items share some specific features. In particular, this figure provides the shared latent information that both users and items have. In this case, items correspond to jobs and as it is expected, the skills are indeed the features that relate these two sets.

Following the example of the figure below, it is feasible and more appropriate to perform a matching throughout embedding rather than following the graph-based criteria for simplicity, scalability and efficiency. These aspects are directly related with the amount of data available in the database.

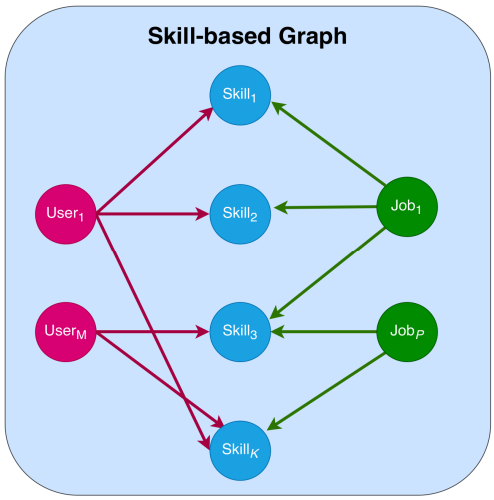

Fig. 4 Relation between users and items (jobs) via graph theory methodologies (taken from D3.2)

[1] Flair https://github.com/flairNLP/flair
[2] HuggingFace https://huggingface.co/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

Furthermore, the embeddings of the items are determined in the same fashion, so that via Transformers models which makes the process more robust in comparison with the previous version where graph-based methods were employed and the preprocessing of the data required more resources (a Graph-knowledge needed to be computed and updated).

3.2.2 Users and Items do not share the same Features

This second scenario implies that the set of users and items do not share similar features and therefore, additional steps are required in order to later on perform the matching. This is the case where we want to associate mentors with mentees following a certain criterion between them such as the schedules they have, the preferences of the mentee with regards to mentors. In this case, we will need to perform the matching having information that may belong to different modalities by producing rule-based conditions that will improve the matching for these cases. For example, if the user has some gender or age preferences, the matching system takes this into account to delete from the analysis the items that do not satisfy these pre-conditions.

3.3 Matching Techniques

During this section, all the algorithms and procedures that have been implemented to build the matching component will be described. As it was previously introduced, depending on the domain, the matching criteria may differ since distinct features from both users and items will be considered.

More specifically, the section is divided in the following steps:

- A preliminary filtering of the data to avoid calculations that are not necessary (for example, analyse jobs which are in a different country where the user is staying).
- The set of procedures for executing the matching. In this case, we will analyse two ways for this purpose:
    - Using Transformers for generating embeddings that subsequently are used for computing the cosine distance to get the matching score.
    - Using statistical methods such as Jaccard similarity or Fuzzy matching to perform the same analysis.

3.3.1 Filtering Data By Criteria

The first stage when retrieving information from a large dataset is to filter out irrelevant data that may not be useful for the final user and avoid unnecessary calculations at the same time.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

For instance, if the user is allocated in Spain, it does not make sense to provide him/her with jobs posted in Italy, even if those jobs are available in the same dataset.

Another example of filtering data before dealing with the matching occurs in the social life scenario. If the migrant, refugee or asylum seeker selects some preferences regarding gender or location, the system employs this set of inputs to firstly filtering the items.

3.3.2 MATCHING VIA TRANSFORMERS

As we already mentioned, two main manners of computing the matching are handled regarding the set of features that are available from both users and items.

Moreover, in such scenarios where both users and items share the same features (i.e. job seeking), the following pipeline presented in Fig. 5 is executed to compute a matching score between the input user and one specific item.

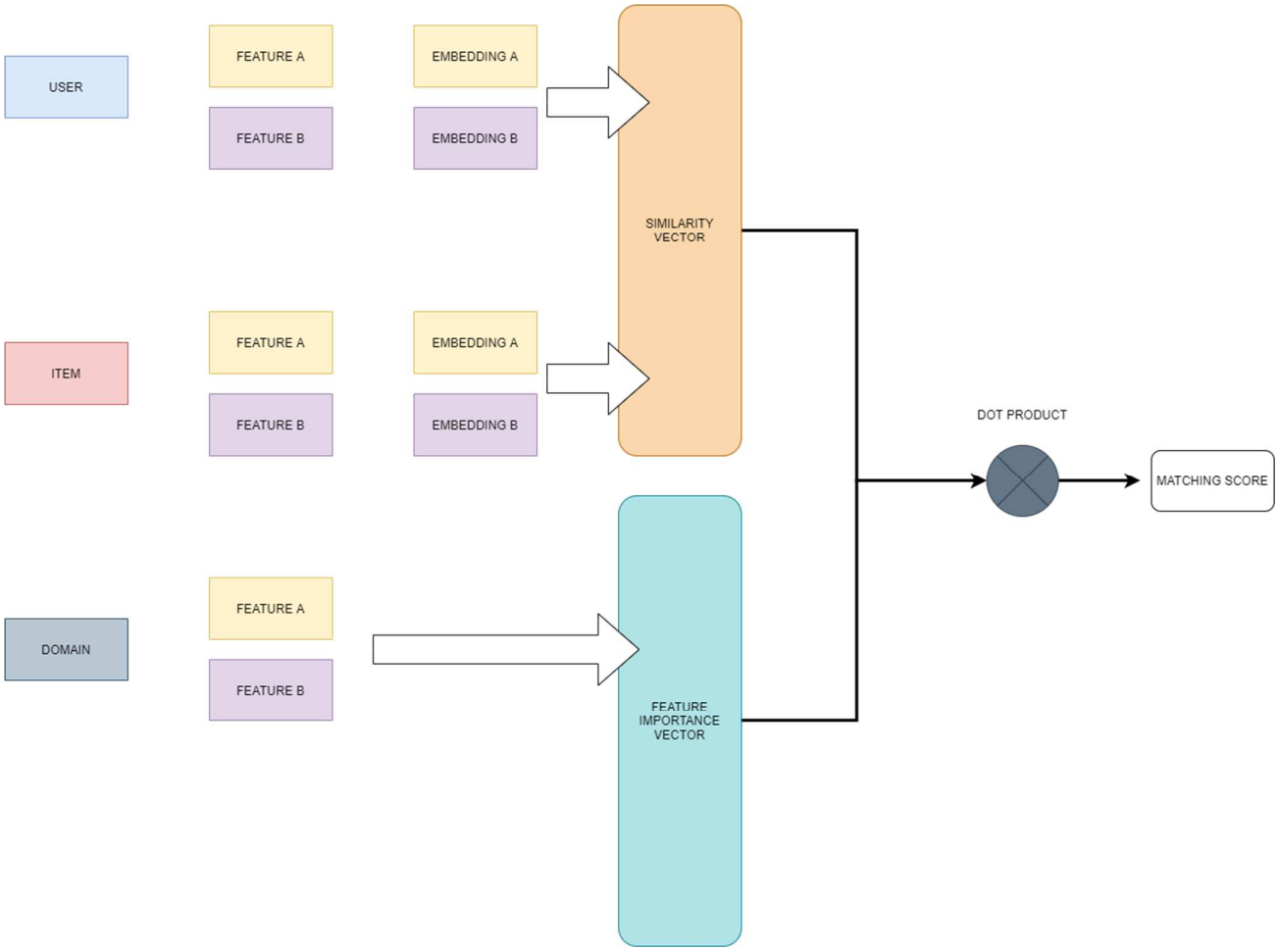

Fig. 5. Principal phases needed to compute a matching score when both users and items share the same features

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

Considering the figure above, the first stage consists in retrieving the selected features according to the domain:

- Each feature will have an associated weight which informs about the importance that such feature must have in the final scoring. To do so, many discussions were arranged with the end-users of the Consortium in order to clarify how these importance or weights may need to influence in the final score. In addition, the weights have been normalized to sum up one in order to keep the probability meaning of the final score.
- For each item and user, we collect the embeddings considering separated features which later on are used to perform a distance metric throughout the so-called cosine distance. More specifically, each pair of features (feature A from User N and feature A from Item N) will produce a single distance which will be stored in the distance vector. As expected, this vector will have the dimension of the number of features. A similarity vector is obtained by subtracting one to the distance vector (closest vectors imply highest similarity and vice-versa).
- Finally, a dot product between the feature importance vector and the similarity vector is computed to obtain a certain matching score. As one may observe from the above figure, the final matching score depends on the similarity as well as on the influence of each feature in the sense that some features will have more influence in the final score than others.

It is important to remark that the aforementioned stages are computed for all possible combinations of items for the input user. Thus, instead of talking about vectors, a more compact similarity matrix is computed.

Once the scores have been computed for each item, the next step consists in sorting them by highest value which indicates that those items that are more similar or adequate for the user will be the first ones and so on.

Finally, once they have been sorted, the matching produces as output the best M items according to these criteria that will be properly forwarded to the recommender system to be later on provided to the final user via UI.

The aforementioned steps are summarized in the Figure above to make the explanation clearer. For example, regarding the Job Seeking scenario, each input user will have a set of similarity scores for all the available jobs in the database and the more adequate ones will be provided in the specific section of the UI of the REBUILD APP after the process has been completed.

Additional verifications are also included in the analysis. For example, if the number of desired returning items (M) is higher than the number of available items in the database (N), the matching will provide N outcomes.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

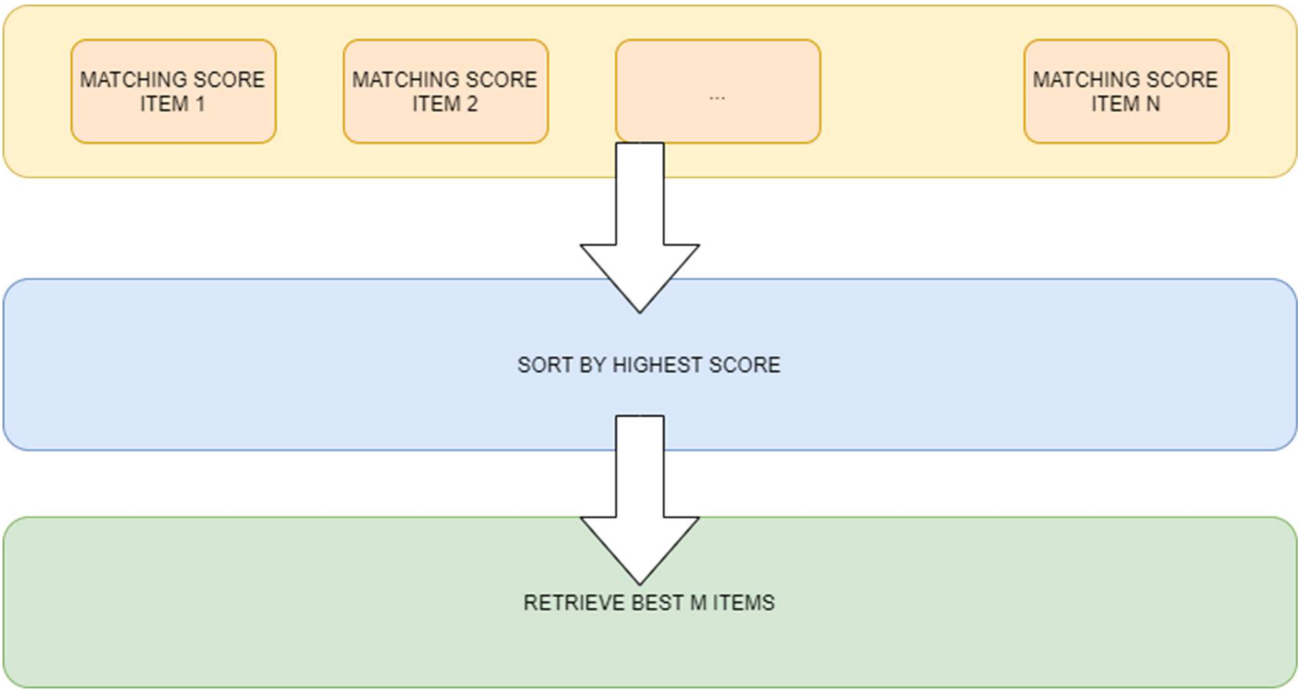

Fig. 6. Pipeline for retrieving the best matching results of a certain input user.

3.3.3 MATCHING VIA JACCARD SIMILARITY & FUZZY MATCHING

A second option for providing matching outcomes based on similarities between Users and Items can be addressed following the so-called Jaccard Similarity distance metric whose equation is presented below:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Equation 1. Definition of the Jaccard similarity metric for two sets A,B.

As it is observed from the previous equation, the Jaccard metric attempts to compute a similarity score between two sample sets, A, B by taking the ratio between the Intersection of both sets and their union. Furthermore, this approach is very appropriate when working with lists and arrays structures in order to easily compare them based on the aforementioned ratio. An example of this case is provided below:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

```
In [10]:
# Hobbies for different Users
hobbies_A = ["reading", "music", "working out", "cooking"]
hobbies_B = ["music", "cooking", "travel"]
hobbies_C = ["learning", "programing", "going out"]
hobbies_D = ["music", "cooking"]
```

```
In [11]:
def jaccard_similarity(list_A, list_B) -> float:
    intersection = len(list(set(list_A).intersection(list_B)))
    union = (len(list_A) + len(list_B)) - intersection
    return float(intersection) / union
```

```
In [12]:
similarity_A_B:float = jaccard_similarity(list_A=hobbies_A, list_B=hobbies_B)
print(f"Jaccard Similarity score between A and B: {similarity_A_B}")

similarity_C_D:float = jaccard_similarity(list_A=hobbies_C, list_B=hobbies_D)
print(f"Jaccard Similarity score between C and D: {similarity_C_D}")

similarity_B_D:float = jaccard_similarity(list_A=hobbies_B, list_B=hobbies_D)
print(f"Jaccard Similarity score between B and D: {similarity_B_D}")
```

```
Jaccard Similarity score between A and B: 0.4
Jaccard Similarity score between C and D: 0.0
Jaccard Similarity score between B and D: 0.6666666666666666
```

Fig. 7. Jaccard similarity metric example to produce a matching score based on the hobbies of different sample users

One of the main advantages when using the Jaccard metric lies in the fact that it is very intuitive, easy-to-use and explainable for end-users. However, this metric has also many constraints when working with text-based features such as it occurs in REBUILD:

- It depends on the language of the data
- It does not distinguish between lower/upper cases
- It is not fair when the sets have considerable different sizes

Thus, in order to be used many preprocessing stages are needed including the need of a translation component to put all the data in the same language and a string preprocessing method to remove lower-upper cases as well as other modifications in the strings.

On the other hand, Fuzzy Matching or Fuzzy String Matching which is also denoted as Approximate String Matching, consists in a process which attempts to find strings or documents that more or less match a certain pattern. Many of these applications involve the calculation of the so-called *Levenshtein* Distance which is defined as follows:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Equation 2. Definition of the Levenshtein Distance

To carry out the implementation of the Fuzzy String Matching techniques, we have employed the Python Library named as fuzzywuzzy[3] which provides several methods to compute ratio scores between pairwise strings. In our approach, we have combined some of these methods to generate a final matching score including:

- **Partial token sort ratio:** which is based on tokenizing the string, sorting the tokens alphabetically, and then joining them back into a string.
- **Partial token set ratio:** in this case, the method first tokenizes both pairwise strings, but instead of immediately sorting and comparing, it splits the tokens into two groups: intersection and remainder which are finally employed to build up a comparison string.
- **Sort ratio:** A restrictive version of the partial token sort ratio.
- **Set ratio:** A restrictive version of the partial token set ratio.

More information about the whole set of methods that this library provides can be found at: https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

More specifically, the procedure when using Fuzzy matching is summarized in the following figure:

[3] https://github.com/seatgeek/fuzzywuzzy

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

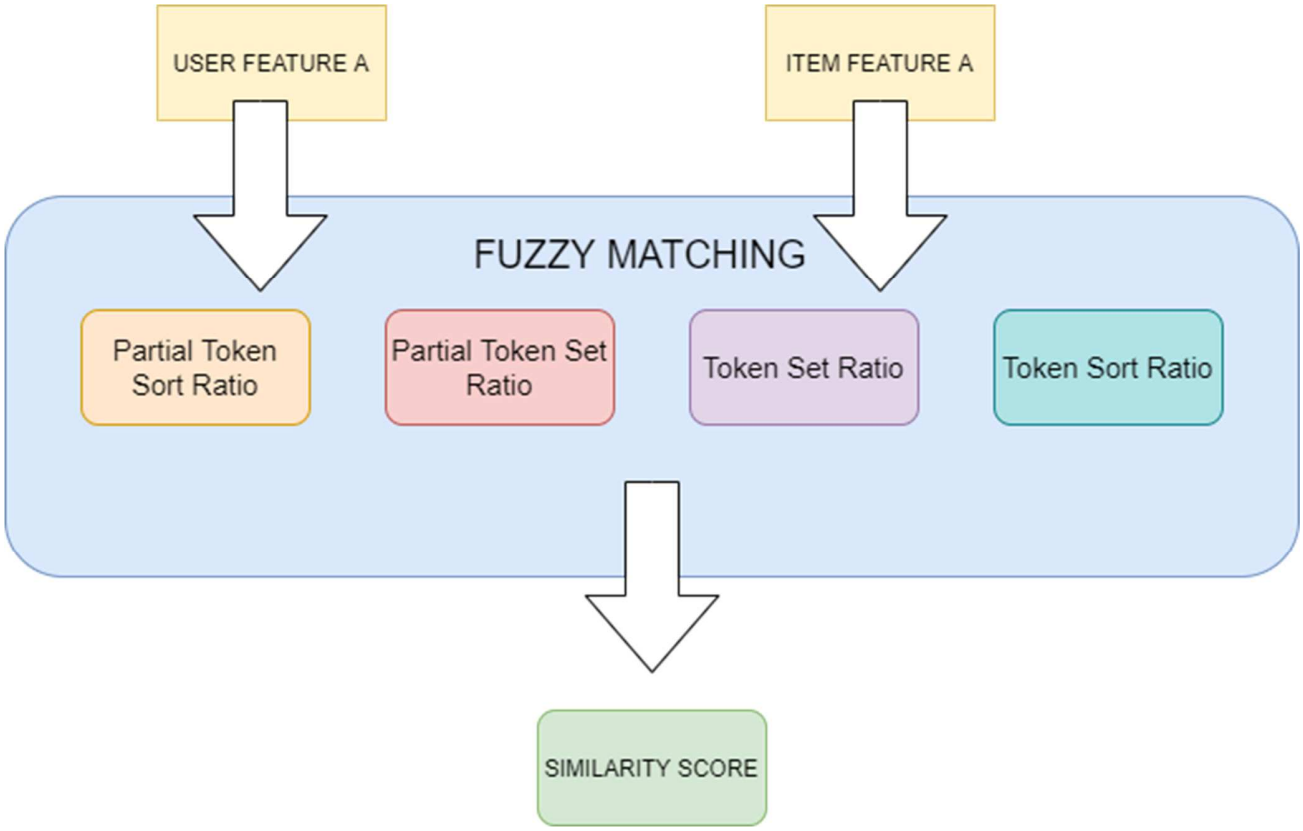

Fig. 8. Pipeline for computing the matching score when using fuzzy matching

In the above Figure, the set of steps that are needed for computing the matching score when following the fuzzy matching schema is presented. In particular, once the features are extracted and preprocessed, four different scores are retrieving from the different fuzzy methods. Finally, an aggregator score based on a weighted average is produced in order to generate the final similarity score.

As expected, this step is performed for all the features. As we saw in the previous section devoted to Matching via Transformers, a dot product is computed between the feature importance vector and the similarity vector. The only modification is that in this case, the similarity vector will be produced via Fuzzy matching algorithms rather than the cosine distance between pairwise embeddings generated via Transformer models.

Moreover, like in the Matching via Transformer models pipeline, this process is performed for all the user-item pairwise. Then, the final matching scores are sorted and the system returns the best M results as it is illustrated in Figure XYZ.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

3.4 Matching Features per Use Case

The objective of this section lies in providing an overview of the features that are used when performing the matching on a specific use case or scenario.

3.4.1 Job Seeking Scenario

The following features are collected from Jobs regarding the job seeking scenario. More specifically, this data has been previously posted in the database in order to be reachable by the matching component.

Table 1. Set of features collected regarding Jobs

Input Feature	Type	Description	Preprocessing	Importance
title	String	Job title	Preprocessing needed to remove stop words and non-relevant characters	High
description	String	Job Description	Preprocessing needed to remove stop words and non-relevant characters	Low
localization	ISA-Location	Job localization	"The localization object is preprocessed to extract from both the country and the postcode, the location of the user to perform a first matching by localization"	High
active	Boolean	Indicator whether the job offer is still open or not	No needed	High
skills	Array<Skill>	Array with skill objects from ESCO framework required for the job.	For each skill, retrieve name and importance	High

Table 2. Set of features regarding users

Input Feature	Type	Description	Preprocessing	Importance
workExperience	Array<WorkExp	User Work Experience	Preprocessing needed to remove stop	Low

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

	erience>		words and non-relevant characters	
residencyLocation	ISA-Location	User localization	"The localization object is preprocessed to extract from both the country and the postcode, the location of the user to perform a first matching by localization"	High
workPermit	Boolean	Indicator whether the user has or not permission for working	No needed	Medium
softSkills	Array<Skill>	Array with skill objects from ESCO framework acquired by the user.	For each skill, retrieve name and importance	High
professionalSkills	Array<Skill>	1st matching: Array with skill objects from ESCO framework acquired by the user.	For each skill, retrieve name and importance	High

3.4.2 EDUCATIONAL TRAINING SCENARIO

The following features are collected from courses and training material regarding the educational training scenario. More specifically, this data has been previously posted in the database in order to be reachable by the matching component.

Table 3. Set of features collected regarding courses

Input Feature	Type	Description	Preprocessing	Importance
courseTitle	String	Course title	Preprocessing needed to remove stop words and non-relevant characters	Medium
courseDescription	String	Course Description	Preprocessing needed to remove stop words and non-relevant characters	Low
country	String	LSP country	No needed	High

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

coursePrerequisites	Array<Skill>	Array with skill objects from ESCO framework required for the course.	For each skill, retrieve name and importance	High

Table 4. Set of features collected regarding users

Input Feature	Type	Description	Preprocessing	Importance
EducationalExperience	Array<EdcationExperience>	User Educational Experience	Extract educational level	Medium
residencyLocation	ISA-Location	User localization	"The localization object is preprocessed to extract from both the country and the postcode, the location of the user to perform a first matching by localization"	High
softSkills	Array<Skill>	Array with skill objects from ESCO framework acquired by the user.	For each skill, retrieve name and importance	High

3.4.3 Social Mentoring Scenario

In this case, the following features are collected from both Mentors and Mentees in order to produce the matching among them:

Table 5. Set of features collected regarding both mentors and mentees

Input Feature	Type	Description	Preprocessing	Importance
knownLocation	String	Location of the user	The localization object is preprocessed to extract from both the country and the postcode, the location of the user to perform a first matching by localization	High

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

availabilities	Array<String>	available schedule of the user	The list is preprocessed to make easier the process of the matching	Medium
languageToSpeak	ISA-Language	Spoken language in the service	Extract language name and the level	High
otherLanguages	Array<ISA-Language>	List of additional languages	Extract language name and the level for each language	High
hobbies	Array<ISA-Hobbies>	List of hobbies	Extract set of hobbies from embedded object	Medium
matchingLimitations	Array<String>	List of constraints	Get constraints	High
ageRange	String	Age range	Preprocess age range	High
expectations	Array<String>	list of expectations	Preprocess strings	Medium

3.5 MATCHING IMPLEMENTATION: A SOFTWARE OVERVIEW

To produce the implementation of the algorithms, an Object Oriented Programing (OOP) fashion has been followed. The main set of algorithms have been implemented in a single class which contains the main methods and functions which are used in all the available domains including social life, job seeking and educational training. A screenshot with some of the functionalities is provided in Fig. 9. As expected, all the methods are static and can be used in further scripts and applications.

On the other hand, depending on the final domain, a set of specific functions has been released as well following the same OOP manner as before. In particular, several classes were implemented using heritage to make the code structure more efficient. This is illustrated in Fig. 10, where a screenshot of some of the functions related to the matching for the job seeking scenario is shown.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

```python
class Matching(object):

    @staticmethod
    def compute_matching_by_distance(document: object, country_name: str,
                                     location_key: str = "knownLocation",
                                     postcode_key: str = "postCode") -> dict:
        geocode_data: dict = {}
        try:
            # 1. Get input object country code
            country_code_input: str = get_country_code_by_name(
                country_name=country_name)

            # 2. Get geo data
            postcode: str = document.__getattribute__(
                location_key).__getattribute__(postcode_key)
            geocode_data: dict = get_geocode_data_from_postcode(
                country_code=country_code_input, postcode=postcode)
        except Exception as e:
            logger.error(e)
        return geocode_data

    @staticmethod
    def sort_similarity_scores(similarities: np.ndarray) -> np.array:
        sorted_indexes: np.array = np.array([])
        try:
            # 1. Sort results
            sorted_indexes: np.array = np.argsort(similarities.flatten())[::-1]

        except Exception as e:
            logger.error(e)
        return sorted_indexes

    @staticmethod
    def generate_document_embedding(input_object: dict, feature_names: list,
                                    embeddings: list, doc2vec: str,
                                    eos: str = "#") -> np.ndarray:
        input_object_emb: np.ndarray = np.array([])
        try:
            input_object_str: str = DocumentEmbedding.get_encoded_str_document_from_object(
                obj_doc=input_object,
                feature_names=feature_names, eos=eos)

            input_object_emb: np.ndarray = DocumentEmbedding.generate_doc_embedding(
                document=input_object_str, embeddings=embeddings,
                doc2vec=doc2vec)
        except Exception as e:
            logger.error(e)
        return input_object_emb
```

Fig. 9. Some of the main methods that are employed along all the domains to compute the matching

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

```python
class JobSeekingMatchingAlg(Matching):
    @staticmethod
    def retrieve_esco_occupation_from_object(data_obj: dict,
                                             occupation_feature: str = "escoOccupation") -> str:
        occupation_str: str = ""
        try:
            # Retrieve ESCO occupation
            occupation_str: str = data_obj.get(occupation_feature).get("title")

        except Exception as e:
            logger.error(e)
        return occupation_str

    @staticmethod
    def generate_job_document_embedding(job_obj: dict, embeddings: list, doc2vec: str,
                                        occupation_feature: str = "escoOccupation") -> np.ndarray:
        job_object_emb: np.ndarray = np.array([])
        try:
            job_str_doc: str = JobSeekingMatchingAlg.retrieve_esco_occupation_from_object(
                data_obj=job_obj, occupation_feature=occupation_feature)
            job_object_emb: np.ndarray = DocumentEmbedding.generate_doc_embedding(
                document=job_str_doc, embeddings=embeddings,
                doc2vec=doc2vec)
        except Exception as e:
            logger.error(e)
        return job_object_emb

    @staticmethod
    def generate_user_document_embedding(user_obj: dict, embeddings: list, doc2vec: str,
                                         occupation_feature: str = "escoOccupation",
                                         feature_name: str = "workExperiences") -> np.ndarray:
        user_object_emb: np.ndarray = np.array([])
        try:
            user_str_doc: str = ""

            # For work Experience in all experiences
            for data in user_obj.get(feature_name):
                user_data_str: str = JobSeekingMatchingAlg.retrieve_esco_occupation_from_object(
                    data_obj=data, occupation_feature=occupation_feature)
                user_str_doc += "#" + user_data_str
            user_object_emb: np.ndarray = DocumentEmbedding.generate_doc_embedding(
                document=user_str_doc, embeddings=embeddings,
                doc2vec=doc2vec)
        except Exception as e:
            logger.error(e)
        return user_object_emb
```

Fig. 10. Some of the main methods that are employed by the Job seeking use case

4 SOFTWARE RELEASES

The scope of this section is to describe how the different services have been deployed from a technical perspective.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

Moreover, for each of the available scenarios, a RESTFUL API has been deployed using mainly Python, Flask[4], Docker. More specifically, Python has been used for programming the matching functionalities whereas Flask has been employed to create the server of the application. Finally, Docker is used to virtualize the application making it easy-to-deploy by automatically creating instances of the application in a virtual environment.

Fig. 11 summarizes the aforementioned aspects that have been considered when designing the component for the different scenarios.

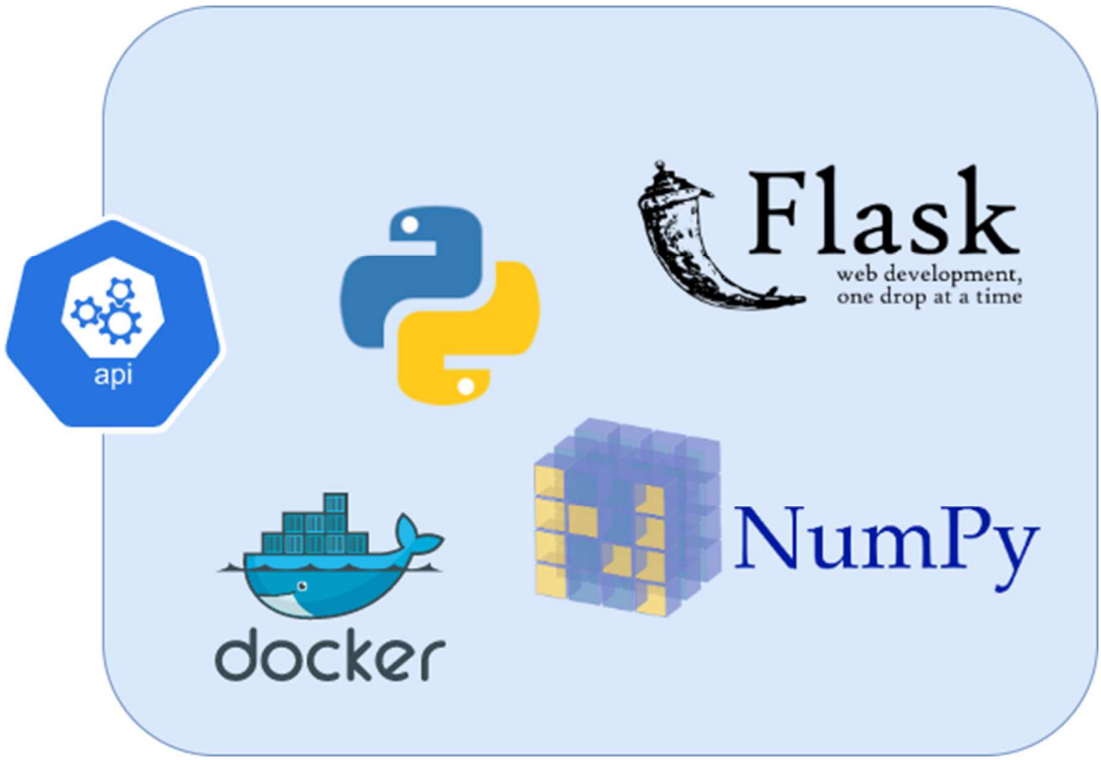

Fig. 11. Representation of the deployment of the services from a technological perspective

Depending on the scenario, the service contains different endpoints that can be called to retrieve the matching results. In fact, the recommender engine component, which is full-described in D3.6, is the key element that generates the requests in order to retrieve the results that later on will be provided to the user throughout the UI of the REBUILD application. On the other hand, the matching algorithms are included in all the services as an additional library dependency. The description of the different APIs that have been developed and deployed for the sake of the project are completely explained on D3.6.

[4] Flask Library: https://flask.palletsprojects.com/en/1.1.x/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

5 CONCLUSION

This deliverable presents the prototype regarding the skill-matching component of the REBUILD project. The goal of this component is to provide useful information for migrants across different domains such as job seeking, social life or educational training by providing recommendations using a set of matching procedures. In this regard, this document has focused on the techniques and procedures that are followed when computing the matching for the different domains.

Furthermore, it also presents an updated state-of-the-art within the matching framework by providing some details about the techniques and approaches that are currently followed in other environments or applications including novel techniques such as Transformers.

In addition, this deliverable remarks the pipeline that was designed and implemented to easy-deploy the matching algorithms as a collection of services in a RestFul API fashion which is more appropriate in production environments and applications such as the one presented in REBUILD.

On the other hand, the document also describes the main options that have been considered when conducting the matching approaches including using Transformers for creating potential embeddings from both users and items as well as fuzzy matching techniques when the usage of Transformers is not valid due to the nature of the data.

Along this document we have also described how the different services regarding different use cases were developed, deployed and integrated within the REBUILD framework throughout Docker (for virtualization) and Flask (for creating a web service in Python).

Finally, we conclude this document by remarking that the services have been deployed and validated along with the current version of the REBUILD application in order to verify the performance within a real production environment.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 822215

6 REFERENCES

Abrahamson, K. 1987. Generalized string matching. *SIAM J. Comput. 16, 6 (December 1, 1987), 1039–1051. DOI:https://doi.org/10.1137/0216067*

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. Universal Transformers. *International Conference on Learning Representations (ICLR'19)*.

Halkidi, M., Batistakis, Y., and Vazirgiannis, M. 2002. Cluster validity methods: part I. *SIGMOD Rec. 31, 2 (June 2002)*, 40–45. DOI:https://doi.org/10.1145/565117.565124.

Katharopoulos, A., Vyas, A., Pappas, N. & Fleuret, F.. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. Proceedings of the 37th International Conference on Machine Learning, in Proceedings of Machine Learning Research 119:5156-5165 Available from http://proceedings.mlr.press/v119/katharopoulos20a.html.

Kumar, A. (2019). Fuzzy String Matching Algorithm for Spam Detection in Twitter. In Security and Privacy (pp. 289–301). Springer Singapore.

Sloan, S. and Lafler, K. P. Data Preparation and Fuzzy Matching Techniques for Improved Statistical Modeling. 1 Jan. 2018 : 367 – 375.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. 2017. Attention is all you need*. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

Wang, J., Lin, C. and Zaniolo, C., MF-Join: Efficient Fuzzy String Similarity Join with Multi-level Filtering, *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, Macao, Macao, 2019, pp. 386-397, doi: 10.1109/ICDE.2019.00042.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing*arXiv e-prints*, arXiv:1910.03771

Wolf, A. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38–45). Association for Computational Linguistics.

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215

REBUILD

ICT-enabled integration facilitator and life rebuilding guidance

Deliverable: D3.5 Skills and needs matching for migrant guidance

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 822215.