A General and NLP-based Architecture to perform Recommendation: A Use Case for Online Job Search and Skills Acquisition

Rubén Alonso
R2M Solution s.r.l.
Pavia, Italy
ruben.alonso@r2msolution.com

Danilo Dessí
Knowledge Technologies for Social Sciences Department
GESIS Leibniz Institute for the Social Sciences
Cologne, Germany
danilo.dessi@gesis.org

Antonello Meloni
Department of Mathematics and Computer Science
University of Cagliari
Cagliari, Italy
antonello.meloni@unica.it

Diego Reforgiato Recupero
Department of Mathematics and Computer Science
University of Cagliari
Cagliari, Italy
diego.reforgiato@unica.it

ABSTRACT

Natural Language Processing (NLP) is crucial to perform recommendations of items that can be only described by natural language. However, NLP usage within recommendation modules is difficult and usually requires a relevant initial effort, thus limiting its widespread adoption. To overcome this limitation, we introduce , a novel architecture that can be instantiated with NLP and Machine Learning (ML) modules to perform recommendations of items that are described by natural language features. Furthermore, we describe an instantiation of such architecture to provide a service for the job market where applicants can verify whether their curriculum vitae (CV) is eligible for a given job position, can receive suggestions about which skills and abilities they should obtain, and finally, can obtain recommendations about online resources which might strengthen their CVs.

KEYWORDS

Recommendation, Transformers, Natural Language Processing, E-Recruitment

1 INTRODUCTION

Nowadays, recommendation systems are becoming essential to support people in discovering the best items that fit their interests. However, due to the large variety of domains and heterogeneous elements, it becomes challenging to implement such systems. This is even more stressed in platforms that aim at recommending complex items which cannot be described by metadata alone but need more complex descriptions by means of data that are not machine-interpretable. This is the case of items that can only be described by natural language text. In this scenario, items can be objects represented by textual features such as a set of skills required for a job, a set of prerequisites to attend a course, a set of textual metadata describing an object, and so on. Recommending such items on the bases of natural language features is not simple, and requires embedding state-of-the-art Natural Language Processing (NLP) and Machine Learning (ML) approaches in the process. To do so, there already exist several research implementations. For example, in a recent study, NLP has been employed to find institutes' core research areas, group them semantically under their most popular group member, and use them to recommend collaboration across various research institutes [3]. One more recent study investigates how to help job seekers to find suitable job opportunities on the basis of NLP and ML analysis [1]. Broadly, all the existing systems have a core infrastructure that analyzes an input text and exploits NLP and ML to perform recommendations on the underlying domain and task. However, a missing piece of the puzzle is the provision of a general architecture that can be adapted to a variety of use cases. This would make the development of such systems efficient and enable their fast adoption. In this direction, this demo paper proposes a flexible architecture, called , that can be configured to deal with natural language data and exploits NLP and ML to perform recommendations. As a use case scenario, the task of finding a suitable job based on the skills of applicants is presented and implemented in different facets.

In summary, the contribution of this paper is threefold: (i) we propose , a flexible architecture that can be instantiated to recommend items based on natural language descriptions, (ii) we show the use of to support applicants in evaluating the strength of their CVs for a specific job, and (iii) we release the source code of its implementation

2 THE FORESEE ARCHITECTURE

The schema of is shown in Figure 1. It includes an *Interface Module* to take input from the user, a *Database Module* to store lexical resources of the underlying task and domain, an *NLP Text Extractor Module* which parses the text and identifies relevant text spans, a *Transformer-based Module* that exploits state-of-the-art transformer models to convert textual data into vector representations, and

Demo source code: https://gitlab.com/hri_lab1/onet-db26-transformers-demo

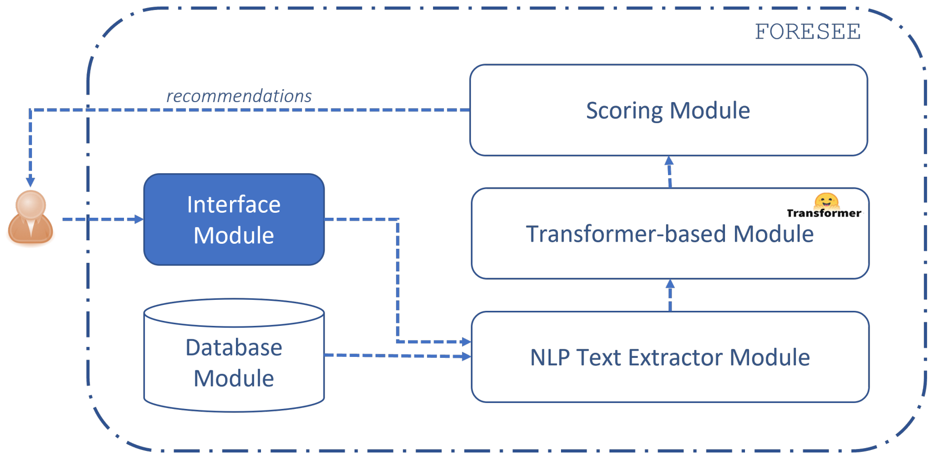

Figure 1: The architecture

finally, a *Scoring Module* that takes the vector representations and generates a ranking of items for the user.

Interface Module. The 's interface collects input from users in natural language. The interface can be a web page, a virtual assistant, or more generally, any kind of system that can collect natural language input.

Database Module. This module is thought to store the lexical resources for the particular domain and task to be addressed and related to the items that will recommend. Each item to recommend should be associated with several textual attributes, each of them may include a score (e.g., for describing a job we might have a field describing the job duties, a field that lists the required skills, a field for the abilities, etc.).

NLP Text Extractor Module. This module parses the input natural language text to detect relevant mentions and text spans (e.g., it can make use of any vocabulary to detect text spans relevant to a given domain to grasp the knowledge from textual content). This module works on both the input provided by the 's user and the dataset attributes.

Transformer-based Module. The Transformer-based Module uses the state-of-the-art framework *SentenceTransformer* [4] to convert an input text into a vector representation. Due to the fact that transformers can be fine-tuned on specific domains, this module can easily be adapted to create context- and domain-dependent vectors.

Scoring Module. This module takes vector representations of the extracted text spans from both the user's input and the database to generate a ranking of items to recommend to the user.

3 E-RECRUITMENT AND JOB SEEKERS

Use case scenario. Looking for a job or selecting a new candidate is key to the success of both professionals and companies. Unfortunately, these activities are becoming more and more time-consuming due to the overwhelming amount of online information that needs to be processed. On the one hand, people who are looking for a position do not have any guidance on the application process, e.g., to understand whether their skills match the desired position, and often do not receive advice to improve themselves and become eligible for it. On the other hand, companies are often struggling to parse the large number of applications they receive for their open positions. These issues make the whole process slow and error-prone; many curriculum vitae (CV) are submitted without carefully reading job postings, overwhelming companies, and candidates do not have a complete overview of tasks that must be carried out making it difficult to understand whether they fit the role. Moreover, it is challenging to collect resources that can be used to acquire the missing skills. To address this scenario, we show an instantiation of which supports job applicants to verify whether their CVs satisfy the necessary requirements for a given job position, identifies which skills are missing, and recommends online courses that can be attended to acquire the missing skills.

Implementation. This section describes how the modules have been instantiated and how they work for the use case scenario. The reader can access the demo at http://192.167.149.11:8000/. The Database Module is instantiated with the O*NET [2] database descriptors of about a thousand occupations covering the entire U.S. economy. Descriptors are lists of textual data including general skills (e.g., *Critical Thinking*), technical skills (e.g., *Microsoft Excel*), work activities (e.g., *Updating and Using Relevant Knowledge*), and so on. The Interface Module is configured to receive an input text or a PDF file containing the CV of an applicant and the desired job. The NLP Text Extractor Module uses the TextBlob library to detect nouns and noun phrases and discards all the textual parts that are not relevant to the proposed scenario (e.g., verbs are discarded by this module for this use case although they play a crucial role in many other NLP applications). The Transformer-based Module embeds the *all-mpnet-base-v2* SentenceTransformers model to generate embeddings about both noun and noun phrases extracted from the NLP Text Extractor and the textual occupations' descriptors. We chose *all-mpnet-base-v2* model because of its performance and

SentenceTransformer: https://www.sbert.net/

O*NET: https://www.onetcenter.org/database.html
TextBlob: https://textblob.readthedocs.io/en/dev/
https://huggingface.co/sentence-transformers/all-mpnet-base-v2

the large variety of heterogeneous sources (e.g., Reddit comments, Stack Exchange, WikiAnswers, etc.) that were employed to collect 1 billion text pairs it was trained on. The Scoring Module uses the cosine similarity to score how the applicant's input matches the occupations contained in the database, returns an overall score for the submitted CV, and generates a ranking of skills that an applicant can obtain or improve to be eligible for the desired job position.

Demo execution. A user is looking for the job *Pharmacists*. After selecting it, he/she can write his/her skills in a text box or upload a CV in PDF format. Let's assume that the applicant uploads the CV available here. Then, the system analyzes the received input extracting the noun and noun phrases and performing a match between them and the skills, tech skills, abilities, etc. required by the *Pharmacists* job position. The system returns a score of 41.85% indicating to the applicant that his/her CV is *almost eligible* for the selected job. It also shows the strengths of the CV for the job, and weaknesses that should be improved. For example, the uploaded CV is eligible for the tech skills (60.82%) and knowledge (66.62%), almost eligible for the skills (43.63%), and not eligible for abilities (3.93%), work activities (35.42%), and tasks (22.32%). After that, it suggests to the user which skills, tech skills, abilities, etc. he/she can acquire to strengthen his/her CV for the selected job position. For example, it suggests *Reading Comprehension* and *Active Listening* as skills, *Epic Systems* and *MEDITECH software* as tech skills, *Oral Comprehension* as abilities, and so on. In addition, the system recommends online courses that can be attended for acquiring them. For example, it recommends the course *DELF B1: Listening Comprehension + Interview with examiner* for acquiring the *Oral Comprehension* ability, showing that the overall eligibility can raise to 45.59% if successfully completed. The running demo also enables us to explore which skills, tech skills, abilities, and so on are required to be eligible for a job position, and can recommend possible jobs given the set of characteristics of the demo's user.

4 CONCLUSION

In this demo paper, we presented , an architecture that can make recommendations based on a natural language input provided by the user and textual descriptors from a database of items. The architecture is general and flexible as each module can be implemented and used to work with any textual data. Efforts are only required within the *NLP Text Extractor Module* to detect the pieces of text and match the features of the recommended items. The architecture can be applied to several domains considering the number of transformer-based models available today. We are now working on developing a virtual assistant to improve the interaction with , and evaluating the prototype within the H2020 STAR project [5]. We would also like to extend its functionalities to deploy a full-fledged framework.

ACKNOWLEDGMENTS

This research was partially funded by the H2020 project STAR - Novel AI technology for dynamic and unpredictable manufacturing environments (grant number 956573).

REFERENCES

[1] Azimh Abdul Ghapar, Feninferina Azman, Masyura Ahmad Faudzi, Hasventhran Baskaran, and Fiza Abdul Rahim. 2022. JOB OPPORTUNITIES RECOMMENDATION FOR VISUALLY IMPAIRED PEOPLE USING NATURAL LANGUAGE PROCESSING. *Journal of Theoretical and Applied Information Technology* 100, 2 (2022).

[2] Michael J. Handel. 2016. The O*NET content model: strengths and limitations. *Journal for Labour Market Research* 49, 2 (2016), 157–176. https://doi.org/10.1007/s12651-016-0199-8

[3] Hiran H Lathabai, Abhirup Nandy, and Vivek Kumar Singh. 2022. Institutional collaboration recommendation: An expertise-based framework using NLP and network analysis. *Expert Systems with Applications* 209 (2022), 118317.

[4] 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

[5] Joze M. Rozanec, Patrik Zajec, Klemen Kenda, Inna Novalija, Blaz Fortuna, Dunja Mladenic, Entso Veliou, Dimitrios Papamartzivanos, Thanassis Giannetsos, Sofia-Anna Menesidou, Rubén Alonso, Nino Cauli, Diego Reforgiato Recupero, Dimosthenis Kyriazis, Georgios Sofianidis, Spyros Theodoropoulos, and John Soldatos. 2021. STARdom: An Architecture for Trusted and Secure Human-Centered Manufacturing Systems. In *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems - IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5-9, 2021, Proceedings, Part IV (IFIP Advances in Information and Communication Technology)*, Alexandre Dolgui, Alain Bernard, David Lemoine, Gregor von Cieminski, and David Romero (Eds.), Vol. 633. Springer, 199–207. https://doi.org/10.1007/978-3-030-85910-7_21

https://github.com/PolyAI-LDN/conversational-datasets/tree/master/reddit
https://huggingface.co/datasets/flax-sentence-embeddings/stackexchange_xml
https://github.com/afader/oqa#wikianswers-corpus
Demonstration CV: http://192.167.149.11:8000/cvdemo.pdf
SBERTmodels:https://www.sbert.net/docs/pretrained_models.html