FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Personalized recommendations for students
Student:	Bc. Čeněk Žid
Supervisor:	doc. Ing. Pavel Kordík, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

Survey content-based and collaborative filtering recommender systems. In content-based systems, focus on systems utilizing neural embeddings of text attributes. Design and implement a prototype of a recommender system for anonymous students that is capable of recommending various opportunities (e.g. theses, mentors, courses) just based on their historical interactions with the system. Also, consider basic profiling of students and find out if you can improve recommendations given student profiles and metadata. Evaluate the performance of the prototype on data provided by the faculty. Analyze errors and suggest improvements. Also, if possible, consider generating explanations for users.

Electronically approved by Ing. Karel Klouda, Ph.D. on 30 September 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Personalised Recommendations for Students

Bc. Čeněk Žid

Department of Information technologies
Supervisor: doc. Ing. Pavel Kordík, Ph.D.

May 4, 2022

Acknowledgements

I would like to thank my supervisor doc. Ing. Pavel Kordík, Ph.D. for his guidance, all the consultations and valuable advice he gave me during the creation of my Master's Thesis. Furthermore, I would like to thank my family and all my dear ones for their endless support.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 4, 2022 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Žid, Čeněk. *Personalised Recommendations for Students.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstract

This thesis provides an overview on job recommendation for students. The research part describes current state-of-the-art methods of recommender systems and analyses current research for student profiling.

The experimental part focusses on the implementation of several different methods described in the research part. These methods are tested and one of those methods is selected, specifically a method based on the term frequency-inverse document frequency algorithm with a custom set of keywords. The general model deals with recommendation based on interactions and is extended with recommendation based on student profiles using the selected method. The presented recommendation system is tested in two experiments. Overall, the results suggest a significant improvement with recommendation using the presented method.

Keywords recommender system, job recommendation, student profiling, NLP, collaborative filtering, content-based model, embedding

Abstrakt

Diplomová práce poskytuje analýzu doporučovacích systémů pro studenty. Teoretická část popisuje současné nejmodernější metody v oblasti doporučovacích systémů. Dále se zabývá analýzou současného výzkumu pro profilování studentů.

Experimentální část se zaměřuje na implementaci různých metod popsaných v rešeršní části. Tyto metody jsou testovány a je vybrána nejvhodnější metoda, specificky metoda založená na term frequency-inverse document frequency algoritmu s využitím vlastního výběru klíčových slov. Obecně je navržen model, který se zabývá doporučováním na základě interakcí a je rozšířen o doporučování na základě profilů studentů pomocí vybrané metody. Prezentovaný rekomendační systém je otestován ve dvou experimentech. Celkové výsledky naznačují výrazné zlepšení při použití navrhované metody.

Klíčová slova rekomendační systém, doporučování práce, profilování studentů, NLP, kolaborativní filtrování, atributový doporučovací model, embbeding

Contents

List of Figures

List of Tables

Introduction

In this thesis, we deal with the issue of recommending work for university students specifically for students of the Czech Technical University in Prague. There have been many papers and research on job recommendation, but there has been very little research on job recommendation for students.

We see the significance of our work in the connection of school and industry, which is in our opinion missing at most universities. The job market can be often very large and confusing, especially for inexperienced students. Many students have no idea what they want to do once they finish their studies, and it might be difficult for them to weigh all the opportunities available.

We want to create a portal that can help students navigate not only on the labour market, but also recommend them potential interesting school and work opportunities.

We choose this issue because we think it could be beneficial for both students and employers to connect them. We believe that this work could help graduate students find their first employment. It could also help undergraduate students find their first job in the field of their studies. We believe that this could help them realise what specialisation they want to pursue in their future studies and work.

First, we want to explore current state-of-the-art methods concerning work recommendation, such as collaborative filtering and content-based models. We also aim to describe the current research that has been done regarding the recommendation of work for students.

In the practical part of our thesis, we want to cover our findings regarding job recommendation for students based on the historical interactions of students with the system. We also want to consider basic profiling of students and to see if we can improve recommendations based on student profiles and metadata. We evaluate the performance of the prototype on the data provided by the faculty and we perform two A/B testings.

Chapter 1

State-of-the-art

In the first chapter, we focus on research of current state-of-the-art methods. We describe the state-of-the-art recommendation system methods and focus more on job recommendation systems, especially for students. In the last part of this chapter, we take a look at student profiling in more detail.

1.1 Recommender Systems

Recommender systems have been widely used in recent years. They have found their place in e-Commerce, on-line advertisements, etc. Recommender systems are used by most of the media orientated web services such as Netflix, Spotify, Youtube. People encounter recommender systems on a daily basis on the Internet. [1]

The goal of recommender systems is to give a certain user a list of recommended items that they might be interested in. Recommender systems typically collect all the interactions the user makes (e.g. searching, clicking, buying). The recommender system can then use past interactions alongside user and item parameters to improve the recommendation not only for the user who made the interactions. The recommendation problem can be interpreted similarly to the predictive modelling task and can be approached as such a model, for example during its tuning. [2]

The goal of the predictive modelling task is to predict what the given user might like (which interaction he is likely to do).

The recommender system consists of the following types of data that are used for the recommendation:

- item attributes,
- user attributes,
- interactions. [2]

Each datatype has its purpose when it comes to recommendation. We can use the item attributes to recognise similar items and, based on the item similarity, we can then recommend related items. The user attributes can be used for the analogical purpose to separate users that are different and to unite those that are similar.

The interactions are used to distinguish users who share a common taste for items. The similar taste for items can be shared across multiple user groups (based on their attributes regardless on their, for example, age, gender, etc.). [1]

1.1.1 User-item Interactions

User and item attributes might seem like the most important part of the recommendation system, but in many practical situations, user-item interactions are, in fact, the most important source of data.

We distinguish two different types of interaction (implicit and explicit). Explicit interactions are those where the user explicitly rates an item:

- likes vs dislikes – we differentiate negative and positive feedback,
- ratings in stars – the item can be assigned a number or explicit number of stars.[2]

Implicit interactions are those in which the user does not explicitly say whether he likes or dislikes a certain item. We can collect the following basic interactions:

- Detail views – an event that occurs when a user views a product's detail page (used in e-commerce systems).
- Cart addition – this interaction occurs when a user adds a certain product to the cart (used in e-commerce systems).
- Bookmark – a type of interaction where user marks an item that he/she wants to visit later (e.g. video streaming services, forums)
- Plays – this event occurs when the user streams a song or a video.
- Purchases – a user purchases an item (used in e-commerce systems or classifieds portals)
- Likes and shares – a concept commonly known from social networks. If there is no possibility of a negative rating, the interaction is still considered an implicit interaction. [2]

1.1.2 Recommender Methods

In this section, we focus on different state-of-the-art recommender system methods and cover the two main paradigms, collaborative filtering and content-based methods along with hybrid recommender systems.

1.1.2.1 Collaborative Filtering

The basic and original implementation of collaborative filtering recommends to the active user the items that had been interacted with or liked by similar users in the past. The similarity of two users is calculated solely based on the similarity of the historic interactions (e.g. views, likes). That is the reason why this method is sometimes referred to as a people-to-people correlation. This method is considered to be the most widely used technique in recommendation systems. [3]

The interactions are stored in the user-item interaction matrix. Collaborative filtering methods can be split into two subcategories: memory-based and model-based. The memory-based method works only with the user-item interaction matrix, and the recommendation can be done using a nearest-neighbour search. The model-based approaches assume that there exists an underlying model which can explain the interactions. [4]

The biggest advantage of a collaborative approach is that it does not require information about users or objects and can therefore be used in many situations. In addition, the system is improved with each new user and every interaction.

On the other hand, the biggest issue related to collaborative approaches is the so-called cold start problem, which addresses the situation with a new user (or a new item). When there are no interactions for a certain user (item), it is impossible to recommend anything to that user (or recommend the item to any user).

There are many solutions to the cold start problem. The new user can be recommended a list of random items, the list of the most popular items, or a list of various items, which helps the system to categorise the user. The most sophisticated approach to addressing the cold start problem is to have a different method from collaborative filtering in the early stages of the new user (item). The early stages method can then utilise the attributes of an user (item) to spot a similarity (we suppose that the user/item attributes are available). [1]

1.1.2.2 Content-based Methods

The other family of recommender systems are content-based methods. In contrast to collaborative filtering, content-based methods use information concerning users and/or items. The main idea of these methods is to recommend items similar to those that the given user liked or interacted with in the past.

For example, if a user liked a rock song, the algorithm would then tend to recommend songs of the same genre. Or, on the other hand, recommend the same things to the users that have the same attributes (e.g. age, sex). [4]

The goal of content-based models is to construct a model that explains the user-item interaction previously collected using available attributes (features). For example, we can distinguish which movies interest the most young women compared to older men. [3]

Content-based methods suffer much less from the cold start problem. New users (or items) can be characterised by their attributes and given a relevant recommendation on their first visit. The only time the cold start problem occurs is when new, previously never seen attributes are introduced to the system. [1]

Although not suffering from the cold start problem, the content-based methods have a tendency to create a so-called "filter bubble". The filter bubble problem arises when the recommendations are over-specialised and the system recommends only items to those that were already consumed. [5]

1.1.3 Knowledge-based Methods

A knowledge-based model is not built on the user rating history but is created based on the user's historical queries. The queries specify what the user wishes to see and can return desired results.

These systems can be implemented, for example, on the housing website, where the user specifies the price range, number of rooms, or location. An issue might arise questioning the necessity of the recommendation system when the user can define which results he/she wants to see. User queries may be too specific, resulting in the filtration of all available results. Later, these queries can be used to purpose at least similar items to those the user would like to see. [6]

The system can apply similarity metrics to be able to find the results that do not exactly match. For each query, the relative importance and a utility function that describe the distance between two items need to be known. These functions can be set by an expert as well as extracted from the user feedback. [7]

1.1.4 Hybrid Recommender Systems

Hybrid recommender systems can be considered nothing more than a combination of collaborative filtering and content-based methods. The reason we combine these methods is to overcome the drawbacks that each of the methods faces when used separately. [8]

The architecture of hybrid recommender systems can often vary. The collaborative filtering and content-based parts of the algorithm can be imple-

mented separately to generate predictions and then combined; or they can be built by adding the capabilities of one method to the other. [9]

There are many different types of hybrid recommendation system:

- **Weighted RS** – we define a few models that can each perform the recommendation. We then take the outputs and combine the results using weights.
- **Switching RS** – the RS switches among many redommender systems based on the specific situation.
- **Mixes RS** – first we take the user profile and the features to generate a different set of candidate datasets. We then provide different datasets to different RSs that generate results that are later combined.
- **Feature Combination RS** – we create a contributing RS that generates additional features to the main RS. We can inject a recommendation from the collaborative filtering method into a content-based RS.
- **Feature Augmentation RS** – a contributing RS is used to augment the user profile that is later sent to the main RS. The idea behind this architecture is to improve performance without changing the main RS.
- **Cascade RS** – the main RS is the first module that generates primary results that are then modified in a minor manner by a secondary RS.
- **Meta Level RS** – it uses the model learnt by one RS as input for another. The idea is similar to the feature augmentation RS. The difference is that the contributing RS completely replaces the original data source with the trained model which the main RS uses in its computations. [10, 9, 11]

1.1.5 Content-based Methods Utilising Neural Embeddings

To determine the similarity between items (users) in content-based methods, it is useful to process all available attributes. It is easier to process some features – usually the numerical or categorical ones (e.g. movie genre, price). There are some features that are harder to process such as item description, book description. To use the features that contain continuous text, we need to address them as a natural language processing (NLP) task. [12, 13]

1.1.5.1 Feature Processing Using NLP Methods

In NLP tasks, we usually want to create some kind of embedding that represents continuous texts. To achieve that, we can use classical methods (such as *tf-idf*, term frequency-inverse document frequency) that create larger more sparse vectors that are derived from the keyword extraction process; or we

can use machine learning methods (such as *word2vec* – for more information, see Section 1.1.6) that extract dense vectors which are harder to interpret but are capable of capturing the context between words. [14, 13]

The typical baseline method for NLP tasks is *tf-idf*. The biggest advantage of *tf-idf* is its simplicity for both implementation and computation. The biggest handicap is that it cannot capture any semantic meaning – it takes into account the importance of the words, but it cannot derive the context of the words in which they are used. [14]

In recent years, machine learning methods (such as Word2Vec or BERT-like architectures) have become popular and, in many cases, the state-of-the-art solution when dealing with NLP tasks. [15]

1.1.6 NLP Using Neural Embeddings

Natural language processing (NLP) and machine learning communities have greatly focused on continuous space word embeddings because of their ability to model term similarity and other relationships. The methods derived from Word2Vec, introduced in 2013, have shown that neural embeddings are capable of capturing not only individual words but also the context of those words. Neural network approaches for NLP tasks have evolved since Word2Vec through Doc2Vec (similar idea to Word2Vec) to transformer-like architectures utilising self-attention layers represented by revolutionary BERT, GPT-2 to GPT-3. [16, 17, 18]

1.1.6.1 Word2Vec

Word2Vec can be viewed as not a singular algorithm, but rather a family of model architectures that can be used to learn continuous word embeddings from large datasets. The goal of these algorithms is to search for a more compressed representation, which is achieved by reducing the dimensionality to feature vectors. [19]

Word2Vec algorithm uses the compression idea (autoencoder-like architecture) and tries to learn to output the same value that was received as input. The idea is to build a less-dimensional model that represents the input in one of the hidden layers, the least-dimensional layer in the middle of the neural network model. [19] [20]

The representation can be improved by using the context of the surrounding words that can be learnt using the following two basic approaches.

- **Continuous bag-of-words model** tries to predict the middle word based on the surrounding ones. Usually, words in the immediate neighbourhood are used. This model is called a bag-of-words because the order of the words is not important. The goal is to estimate the follow-

ing equation:

$$P(w_t|w_{t-L}, w_{t-L+1}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+L-1}, w_{t+L}) \tag{1.1}$$

- **Continuous skip-gram model** uses the inverse logic of the continuous bag-of-words model, it attempts to predict which words are used in the context (the immediate neighbourhood – the range can differ) of a single word. The goal is to estimate the following equation:

$$P(w_{t-L}, w_{t-L+1}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+L-1}, w_{t+L}|w_t) \tag{1.2}$$

One of the advantageous side effects of the Word2Vec representation of the features is the hidden relationship between words. Words with similar meanings are closer to each other in the latent-space representation of the feature space, meaning that its easier to recognise words with analogous sense. Furthermore, this means that relationships of similar words go in the same direction (see Equation 1.3 and Image 1.1 for an example). [19]

$$vec(Berlin) - vec(Germany) + vec(France) = vec(Paris) \tag{1.3}$$

1.1.6.2 BERT-like Architectures

Current state-of-the-art architectures still use neural networks, specifically transformers, in their architectures. The huge improvement came in year 2017 with the paper *Attention is All You Need*. [22] The paper introduced a transformer that eschews recurrent neural networks and instead uses only the attention mechanism to capture global dependencies between input and output. This architecture significantly improves computational speed because it enables much greater parallelism and reaches the state-of-the-art results in translation quality.

This architecture is still used in nowadays state-of-the-art models such as GPT-2 and GPT-3. Today's transformers can deal with much more sophisticated tasks than filling in the missing words. GPT-3 is capable of creating stories, poetry, articles, news reports. Only a small amount of input text can be used to generate large sections of quality text. It can also be used to perform automated conversational tasks responding to any person or computer. Interestingly, it can also generate text summaries and programming code. [22, 23]

The attention-like architectures in the transformers allow the use of models that need much lower computational power to outperform the previous state-of-the-art methods. Nevertheless, the paradigm describing that with more computational power and bigger models the quality of NLP processing models increases, still remains. [23]

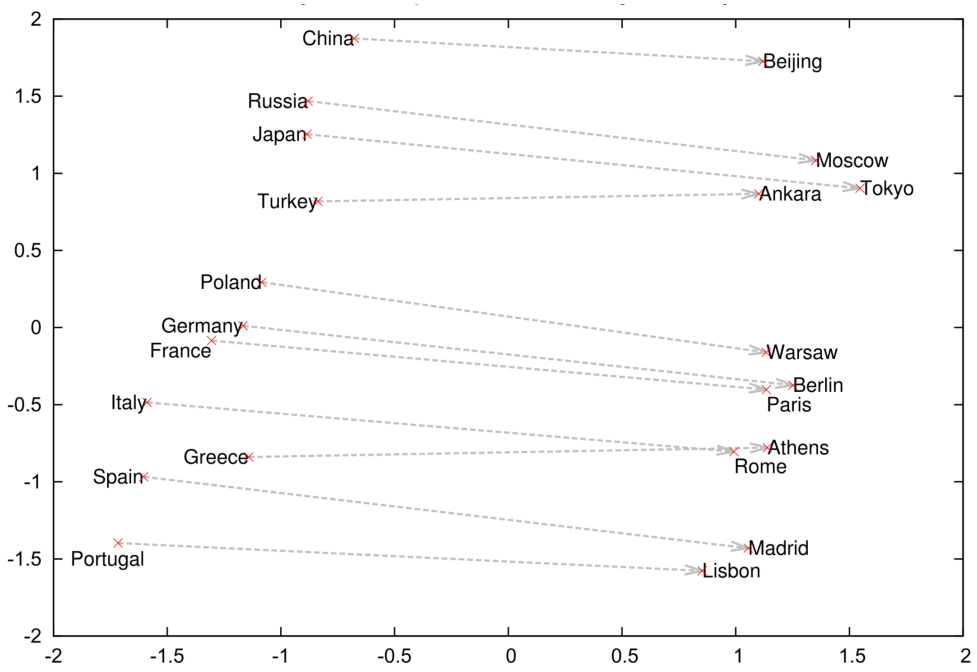


Figure 1.1: The PCA projection of a 1000-dimensional skip-gram vector of countries, their capitals, and the relationships between them. [21]

1.1.6.3 Image Processing

Another feature that can be used to represent an item in a RS is an image. Image recommendation is the same thing as the image classification task. These two tasks share similar goals. Image recommendation involves the underlying process of image classification, but in a much broader sense. Image recommendation needs the most descriptive and distinctive features that can be used to obtain matching items. [24]

1.1.6.4 Usage of Embeddings in Recommender Systems

As mentioned in previous sections, feature extraction methods on NLP tasks using both large sparse vectors or continuous dense vectors can discover more than just the similarity of user-defined keywords or categories. These embeddings can be used within RSs to create more sophisticated features that may contain more information about item/user similarities. That is why feature pre-processing plays a huge role in content-based methods and can help with improvement of not only content-based recommender systems.

1.1.7 Existing Solutions

In this part, we take a look at existing solutions that are available and can be used instead of implementing a recommender system from scratch.

- **Recombee** is an artificial intelligence-powered recommender system that uses deep learning, collaborative filtering methods, and content-based algorithms and provides real-time analytics.

  Recombee is not a traditional rule-based recommender system; instead, it uses an AI-driven solution that can reflect real-time changes, as well as the complexity of user behaviour online. The model enables its users to upload visual data and is capable of processing these images to extract visual similarities.

  By implementing not only collaborative filtering methods, Recombee can deal with the cold start problem by introducing user and item attributes. These attributes are used to discover groups of similar items.

  This recommender system allows users to create custom scenarios, which can each focus on different situations (landing page, personalised recommendation, etc.). It also allows for the creation of so-called business rules – filters and boosters. These methods help to specify various rules based on the item properties. For example, boosters can help with the promotion of new items, or items on sale, and filters can for example prevent children from seeing inaccessible content. [25]

- **Google's recommendation AI** software was introduced in 2019. It is a fully managed service designed for retail-orientated businesses. Its purpose is to increase click-through rates, conversions, and overall revenue. [26]

  Recommendation AI supposedly excels at scenarios with long-tail products and cold-start problem. It uses a so-called context-hungry deep learning model developed in connection with Google Brain and Google Research. Recommendation AI should be able to handle the bias that arises with popular or on-sale items; and be able to dynamically adapt to real-time customer behaviour. [27, 28]

  One of the biggest advantages is the connection with other Google services such as Google Analytics 360, BigQuery or Google Tag Manager. [27]

1.2 Job Recommendation

Job seeking is an activity that most people encounter in their lives. It can be difficult to get into the job market. There are many platforms that try to connect people seeking jobs with the companies that offer them (e.g. LinkedIn,

Jobs.cz). The usage of recommmender systems is quite evident in this specific use case. The RS can be used for both sides, by recommending available jobs to candidates and by recommending potential candidates to companies' that seek employees.

1.2.1 Specific Problems

The job RS differs from a variety of other RS. It is different in its main purpose and logic. If there is a job opening, it is usually filled by one or a few candidates, and it cannot be later recommended to any additional users.

Researchers working on LinkedIn's recommender system encountered some unique information retrieval, system, and modelling challenges while tuning their RS. [29]

The first issue is that personalised recommendations need to be computed in real-time, which becomes a challenge when it comes to scoring millions of different structured candidate job documents with each query without lowering the degree of data freshness and while maintaining reasonable latency limits. The query is composed of the member context and the interests expressed in the member profile.

The underlying retrieval systems often use content-based models, which can create another issue, which originates from their definition (more about this in Section 3.3.3.3). Recommendations are made mostly using the explicit member context and interests extracted from the user profile and may not consider the users' implicit interactions. Therefore, it is possible to face challenges in integrating different types of user-interaction signals into the relevance model.

Lastly, the recommendation problem differs in its core from the more traditional recommender system problems (e.g. book, movie, product recommendation). While all the mentioned problems share the same objective to maximise user engagement, there is a difference that typically the job offer can be filled by one or a few best candidates, whereas the movie can be consumed by innumerable users.

The goal is to provide a sufficient number of candidates, while not delivering too much to overwhelm the job posting. Offering too many job candidates would also result in the decreasing chance of getting the job even for perfectly qualified members. This may result in the decline of unsatisfied users after a certain number of unsuccessful interviews. [29]

1.2.2 User Job Profiles

There are many different pieces of information that the recommender system can use to enhance its recommendation capabilities. The user profile can consist of both basic information about the user and interaction records. Basic user information can consist of the user's background, education background,

previous work experience, demographic information, and many others. The interaction records consist of the user's behaviour history with the recommender system, such as detail views of job postings, searches, and browsing history.

In recent years, data such as previous job duration, social network behaviour history, and other external information sources have drawn interest in the job recommendation task.

1.2.3 Job Recommendation for Students

The task of recommending jobs to students is quite different from the recommendation to people with previous work experience. Students often do not have work experience, and it is harder for them to orient themselves in the job market. On the other hand, we have more personal information about the student. We can take into account their major and courses.

It is quite common for many students to try to get suggestions from their families, friends, mentors, or supervisors during their first job hunt. [30] Many students rely on their university as the primary source of information regarding job possibilities. [31]

Furthermore, the location where an individual's potential job is sought can be more diverse than in the case of other adults. The most preferable locations are usually around the student's hometown or school (university). [31]

The issue we want to address in this paper is that it is difficult for students and employers to go through all the possibilities and face the information overload. [32] This problem can be solved by creating a suitable system for both students and employers to filter the information and get recommendations. [33]

Many times, student data are stored in the school databases but are never utilised because it can be hard to generalise the information contained in the student data. The problem is that there are no processes that transform the data to non-confidential form and still keep their utility. [33]

1.2.4 Related Work

There has been some research done regarding job recommendation for students on the Beihang University, Beijing, China. [34, 35] They try to recommend jobs to students based on the jobs that their graduate students have acquired.

In [34] a student's similarity is calculated to other graduate students. After that, a top-n voting algorithm is used, which provides a recommendation. After which the employer's information is used for re-ranking (see 1.2).

The student's similarity is calculated on the basis of two attribute groups. Firstly, the individual background of the student (home town, gender, univer-

sity, department, courses taken, etc.) and second, his achievement attributes (e.g. grades, scholarships, grants, awards).

The re-ranking algorithm uses information using employer's attributes such as business category, location. These attributes are projected onto the recommendation through graduate students who are used for the original recommendation. The advantage of the re-ranking module is that it can be easily removed, changed, or added without changing the original job recommendation. [34]

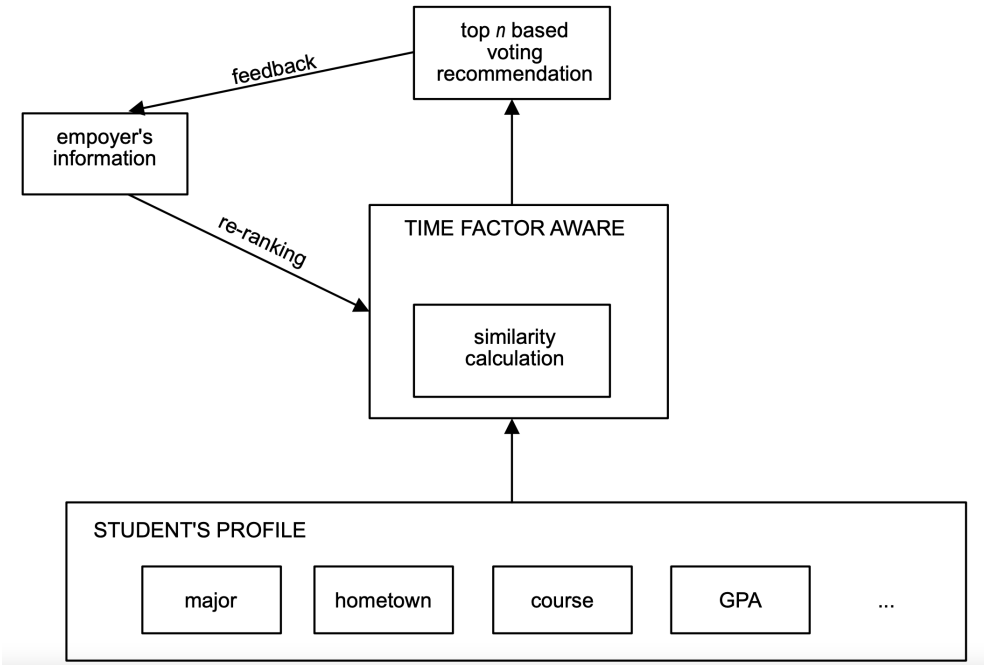


Figure 1.2: Beihang job recommendation system framework

In [34] they found that the similarity of the student's course plays an important role in the job recommendation task. They also found that recent job employment data plays a much bigger role in job recommendation than more historical information.

The other paper on Beihang University [35] focusses more on the importance of the personal attributes of the students and their influence on the quality of the recommendation. The authors emphasise the relevance of the regional economic index (REI) and the regional familiarity index (RFI).

REI is calculated based on the total population, the urban-rural population ratio, and the average consumption and income of a region. RFI represents the individual's familiarity with his hometown and university.

These two indices are combined with the graduate performance index

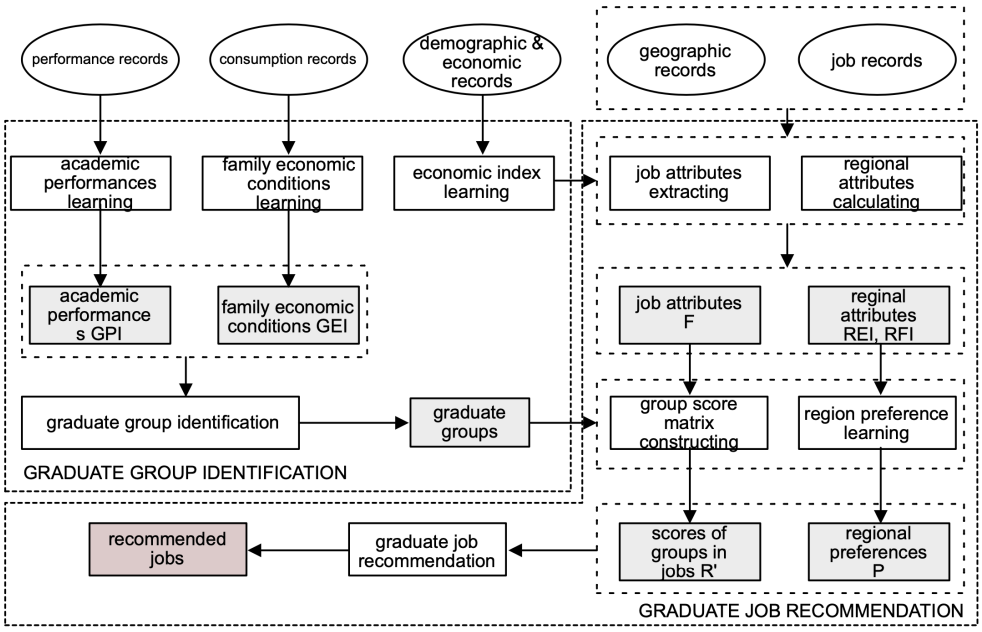


Figure 1.3: The other Beihang job recommendation system framework

(GPI) to construct a more complex recommendation system (see Figure 1.3). They try to solve the following three challenges:

- how to measure the abilities and conditions of a student,
- how to use collaborative filtering methods for students without historical data,
- how to integrate student's preferences for jobs into the collaboration filtering method.

They found that both job attributes and job locations have a positive impact on the hit ratio, but job preferences for job locations are more effective than job attributes (it is necessary to note that the research took place only in China and its regions).

1.3 Student Profiling

There has been quite a lot of work done on student profiling. The only problem is that it is usually done for the purpose of helping them in their school environment and not for the search for a job.

The most common approach is to classify students into clusters that are created using basic clustering methods such as k-means, apriori algorithm or decision trees.

The most common attributes considered for student clustering are:

- Student's grades – it can be further divided into exam marks, term grades, etc.
- Participation in learning environments – text mining and social network analysis techniques were used on the learning forums.
- Personal information – family information, financial support, diseases, work.
- Psychological traits – motivation, participation, cognitive processing strategies, etc.

All the mentioned papers on student profiling focus on student's performance in the school environment. They try to select either extraordinary students or students who need more focus from their teacher in order to succeed.

In conclusion, all the papers point out that it is advantageous to include more information than just student's academic performance. All the attributes mentioned above have their purpose in the student clustering.

1.3.1 Student Skill Mining

Unlike most related work, the paper [36] focusses its research specifically on connecting educational facilities with industry. They point out that in recent years companies have developed interest in research that takes place in universities and they try to build a system to connect the industrial needs with the academic world. This work is carried out at the Faculty of Information Technology at Czech Technical University in Prague.

In this paper, an ontology (skill tree) is built for each student that describes the mapping of student courses (subjects). The purpose of these ontologies is to create a well-organised student profile that can be more easily understandable by external partners. At first, skills are estimated on the basis of the evaluation from university courses, and data mining processes are used to create a skill tree, which is later transformed into a final more standardised set of skills.

Using this mapping, they create a student profile that is easy to interpret and matches potential industrial opportunities. They distinguish two groups of skills: *objective skills* – the skills that are computed using the successfully finished courses, and *subjective skills* – the skills the student enters in his profile at his/her own discretion. Skills aim to represent various levels

of abstraction and are simplified to cover the requirements of the industrial partners of the university. [36]

The paper not only distinguishes the courses that a certain student has completed, but also considers the student's grade. They purpose two different grade mappings. The first mapping takes the successful grades available to which a value is uniformly assigned (A: 1, B: 0.8, C: 0.6, D: 0.4, E: 0.2). The other mapping tries to deal with the issue of dealing with the dissimilar difficulty of various courses by applying a CUME_DIST (cumulative distribution) function to assign the values for each grade. [36]

1.4 Recommendation with Capacity Constraints

In many cases, candidate items are associated with a maximum capacity (e.g., the maximum number of pieces of clothing on stock). Despite the fact that in most recommendation scenarios a capacity constraint is present, nowadays recommendation methods are not designed to deal with or optimise their recommendation in regard to these constraints.

This issue is addressed in [37]. They purpose a *Recommendation with Capacity Constraints* framework that aims to optimise the precision of the recommendation and the usage of expected items with respect to capacity constraints.

They introduce new attributes for both items and users. First, *item capacities* indicate the maximum number of users who can use the given item simultaneously (e.g. visit a movie theatre at a certain time slot). Second, each user has a defined *user propensity* that indicates his probability of following the system recommendations.

They show that when they apply their newly purposed method to three state-of-the-art recommendation models based on the latent factor (probabilistic matrix factorisation, Bayesian personalised ranking and geographical matrix factorisation), they can achieve an improvement on the top-n recommendation quality of the respective unconstrained models.

Chapter 2

Data Analysis

In this chapter, we analyse the data with which we work in this work. Their structure, volume, and manner in which we process them.

2.1 Student Data

The students dataset contains the following columns and has 13,810 rows:

- Study ID (integer) – an identification number of a student's study (if a student failed the study programme and started over again, the study ID would change).
- Username (string) – student's unique identification string.
- Form of the study (category – string) represents whether the student studies remotely or full-time.
- Study programme (string) – name of the study program.
- Field of study (string) – name of the student's field of study.
- Field of study code (category – string) – code of the student's field of study.
- Study begin date (date) - the date on which the student joined the university.
- Study end date (date) - the date on which the student ended his studies at the university.

The university courses data set was composed of the following columns and had 1,122 rows:

- Course ID (integer) – an identification number of the university course.

- Code (string) – short representation of the course.
- Czech (English) name (string).
- Finish type (category – string) – each course can end with a credit, a classified credit, or an exam.
- Scope (string) – the number of lectures and practical tutorials.
- Czech (English) keywords (list divided by commas).
- Lectures' language (CS/EN).
- Czech (English) annotation (string).
- Czech (English) requirements (string).
- Czech (English) lectures' outline (string).
- Czech (English) tutorials' outline (string).
- Czech (English) literature (string).
- Czech (English) goals (string).
- Czech (English) content (string).
- Department name (string).

The student classification dataset contained the following columns with 271,847 rows:

- Credit received (character) – character representing whether the student received the credit.
- Course finished (integer) – the course has been finished by the student (1 for the finished course).
- Grade (character A–F) – student's final grade.
- Study ID (integer) – an identification number of a student's study.
- Course enrolment ID (integer).
- Scheduled course ID (integer) – identification number of the scheduled course.

2.2 Opportunities Data

On an abstract point of view, the opportunity data represent a very similar group of data to the courses. They require a student to have certain skills and help the student improve some of his skills or even acquire new ones. They can be defined by text content or a list of keywords.

In total, we have 166 opportunities that are composed of the following attributes:

- Opportunity ID (integer) – unique number associated with the opportunity.
- Type ID (category – integer) – a categorical value representing the type of opportunity (internship, job offer, study project, challenge, contract, research support).
- Name (string).
- Description (string) – description of the content of the opportunity.
- Keywords (list of strings) – keywords representing the opportunity.
- Location (string) – the location where the opportunity is to be performed.
- Wage (string) – number or string that describes a potential wage.
- Technical requirements (string or list).
- Formal requirements (string).
- Other requirements (any).
- Benefit (string) – benefits that the opportunity offers.
- Job start date (datetime)
- External link (http link) – link for a more detailed description or just the link for the company's website.
- Home office (string) – value that describes the possibility of working remotely.

The opportunity data were filled out by various companies, which is why the quality and structure of the opportunity specifications differs.

2.3 Data Processing

The Czech and English data (descriptions, annotations, etc.) are at a similar level of quality. Therefore, we work only with the English columns during the dataset processing. The English language was also chosen because in later stages of our work, we perform natural language processing tasks that are better documented in English.

We take into consideration only students who are currently studying. We process their current study as well as their past studies (e.g. we merge bachelor and master studies together with certain weights) as long as they were on the same faculty.

2.3.1 Availability

The recommendation system is carried out under the supervision of Unico and its Experts.ai site. That is why student data is not available to the company to use them without their consent. We have to implement a subscription policy in which the user's consent is explicitly provided and he/she has the right to unsubscribe at any moment.

Chapter 3

Realisation

In this chapter, we explain the structure of our recommendation module in terms of how the recommendation is implemented and structured. We dive into different architectures and ideas that were created and later tested.

3.1 Recommendation Structure

In the Chapter 1, we discuss many possible realisation possibilities. We describe some universal methods of recommender systems as well as some very specific implementations of job recommendation for students.

We choose to implement the general structure idea from 1.2.4 and divide the entire recommendation process into two parts.

The first part does not take any student data into consideration; it works solely on the basis of past interactions and possibly item similarity.

The second part is implemented as a re-ranking module (see Figure 3.1) which receives top recommendation from the first part of the recommender system (e.g. 20 items) and re-ranks those items based on the student profile. The position from the interaction-based RS is still taken in the account.

We make this decision due to the scalability of the whole RS. If the system is implemented at different universities in the future, the part of the interaction-based RS can be prototyped very quickly without dealing with legal responsibilities regarding the processing of student data related to the European GDPR[1].

The RS can work entirely without student data, and the student profile re-ranking module can be added later, when all the legal responsibilities are solved.

We also take inspiration in 1.3.1 in the second part of the recommendation system (the re-ranking module). We implement the same idea of acquiring skills from the courses that a student has successfully completed. We adopt

[1] https://gdpr-info.eu/

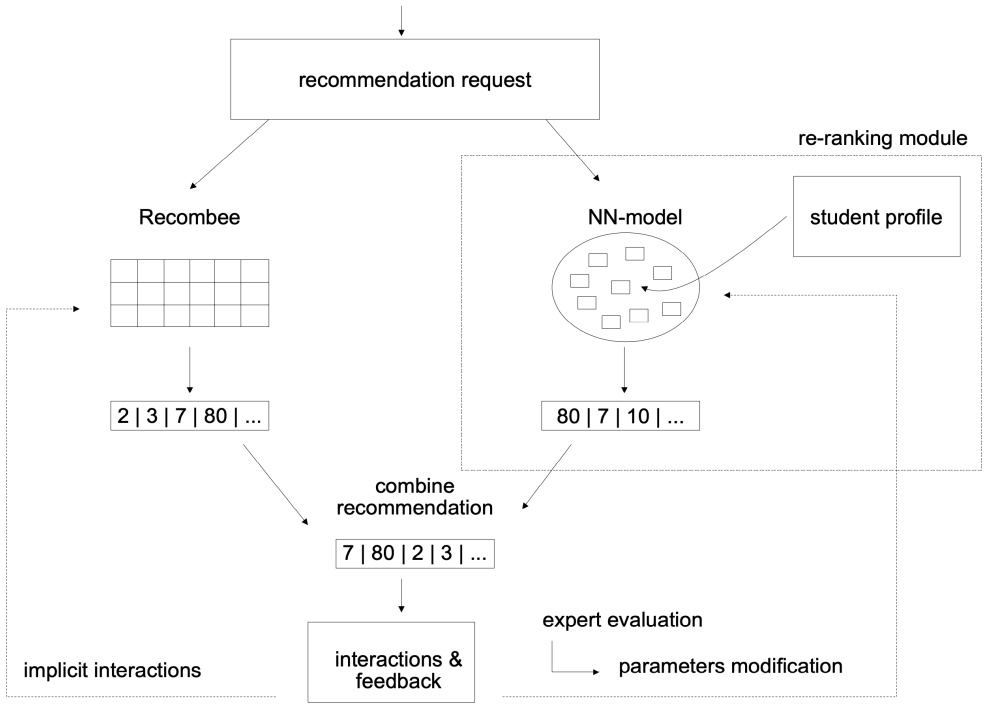


Figure 3.1: The process of our global recommendation schema. The left part displays the necessary part of the RS: the interaction-based RS without the student profiles implemented using Recombee recommendation service. The right part (re-ranking module) describes the recommendation of the basis of the student profiles (for a more detailed description, see Figure 3.2). The right part is removable, and the system works without it.

only the *objective skills* part of the system. By doing that, we try to address the problem with outdated information, where a student places some data in his profile and never updates it. This idea was consulted and approved by the authors of the referred paper who came across the same problem in the past.

We also considered the possibility of giving space to students to specify their preferences about the opportunities they might be interested in (internship, part-time, etc.). We rejected the idea based on the same idea: the student preferences can evolve very quickly in comparison to their desire to constantly update their profile. Furthermore, these preferences can be partially replaced by implementing active filters that the student can use to see only specific desired types of opportunity.

3.2 Interaction RS

The goal of this whole work is to focus more on the creation and utility of the student profile part of the RS, which is why we decided to implement a pre-prepared solution, specifically the Recombee service (see Section 1.1.7) as the interaction part of our RS architecture.

This allows us to use a tuned system that does not require a further improvement (from the algorithmic part of the work, it only requires some data tuning to perform better – if the item attributes are well-set, the recommendation quality should increase).

Although we may have some student profiles, we do not add any student data to the Recombee platform. We want to keep the two modules separated. Adding student data to the Recombee platform would also complicate the matching of user tokens through different devices the user can use (e.g. mobile and desktop).

3.3 Re-ranking Module

The general idea behind all different implementations is to create some kind of student embedding that can then be matched to the embeddings of the available opportunities. There are two ideas of embedding creation described in the following implementation stages.

The first idea is to calculate keywords that individually represent each student and each opportunity. The main issue related to this idea is the difference in language. Some opportunities have been written in Czech, and others have been written in English. To handle this, a translation is required, which can create some additional noise in the opportunity description.

The second idea is to create neural embeddings. These embeddings can potentially deal with the language problem more elegantly. On the other hand, it is much more difficult to interpret these embeddings, even though they might be the state-of-the-art method when it comes to NLP tasks. [38]

3.3.1 Data Preparation

The creation of embeddings always comes with an NLP task. It is crucial to have adequate data when creating embeddings, both neural and keyword-based. For that reason, we need to prepare the data from which we extract the embeddings in the best possible form.

We do not match opportunities directly with students. All the information we have about the students is based on their previous study, and we do not possess any personal information that is not related to the university environment. Therefore, we need to construct an algorithm that can transform the finished courses and courses into a student profile. The first thing we need

to do is create a student profile that can be matched with the available opportunities. Essentially, we need to create embeddings for both students and opportunities.

3.3.1.1 Courses (subjects)

The first thing we need to do is process the university courses which can later be used to build a student profile. There are many columns that must be considered (see Section 2) – name, annotation, keywords, requirements, goals, content, lectures', and tutorials' outlines. The quality of these columns differs over different courses. To avoid losing important information, we consider all the columns mentioned above by merging all the text together. The text is then used to create the embeddings of the course by processing it as an NLP task.

3.3.1.2 Opportunities

The opportunities suffer from a very similar problem – there are many columns that are filled manually by different people. This creates an issue because the texts are not standardised. Some people fill in different information for different columns, which is why all relevant columns need to be included in the embedding creation process (see Section 2.2). We consider the following columns: name, description, keywords, technical requirements, and other requirements; in the same manner as in the course processing.

Another issue that needs to be addressed regarding opportunities is the difference in the language in which they are written. There have been 72 opportunities which were written only in Czech, but we need to translate them to English so that we could use keyword extracting algorithms.

In order not to translate all the opportunities manually, we decide to use a DeepL[2] translation programme (its advanced plan). [39] After the translation, we review the translations and edit the texts if necessary.

3.3.2 Keywords Embeddings

In this part, we describe the methods we use to generate student and opportunity embeddings. As the first method, we choose to create embeddings based on keywords extracted (or manually added) from the available data we have. The general idea of the keyword-based student profiling is depicted in the schema in Figure 3.2.

[2]https://www.deepl.com/translator

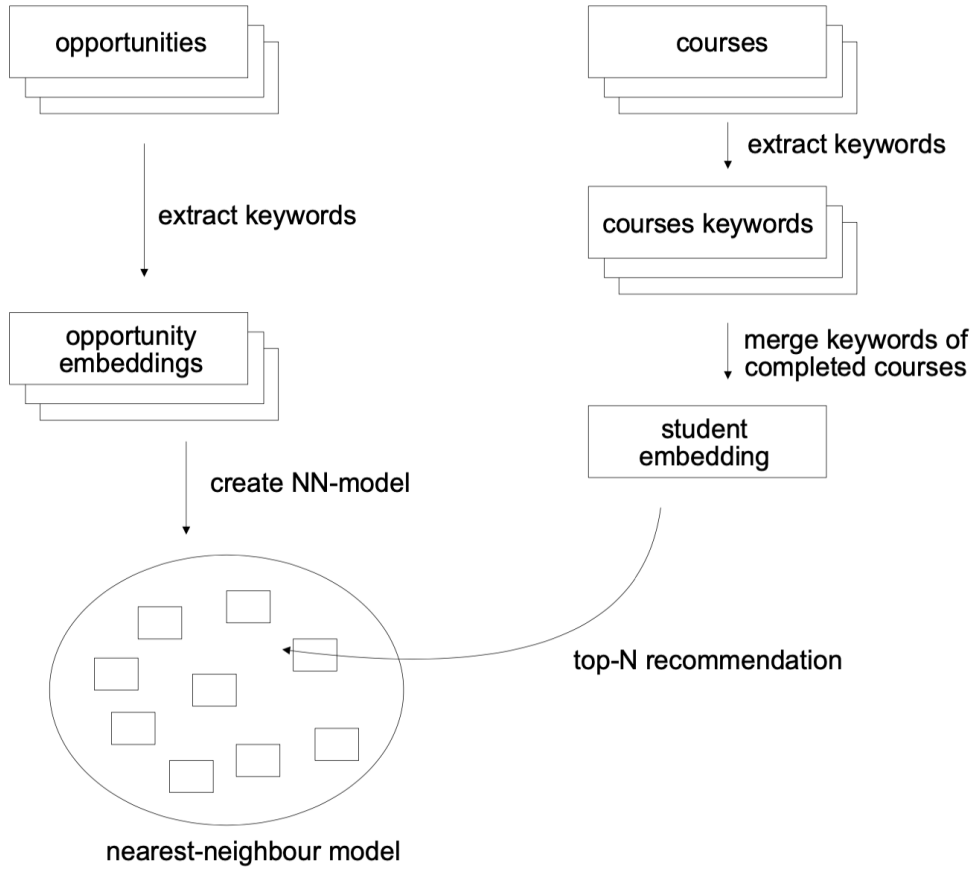


Figure 3.2: The process of the student profiling by extracting keywords from completed courses and available opportunities. The keywords are extracted from all courses and opportunities. Keywords (potentially with weights) of the opportunities represent opportunity embedding, which are used to create a nearest-neighbour model used for recommendation. The student profile (embedding) is composed of keywords from successfully completed courses; this profile represents his embedding. Student embedding is used to determine the closest opportunity embeddings (usually using the cosine distance).

3.3.2.1 Student Embeddings

The only data available about the students are based on the courses (and grades) they studied. Therefore, the first step is to generate keywords for courses.

After having keywords for all courses, we can take the keywords of each course that a certain student had finished and add those keywords to the student's profile. We want to include not only the keyword, but also the

grade of the student. Therefore, each keyword has a weight assigned based on the student's grade he received from the course (A:5, B:4, C:3, D:2, E:1, F:0). The keyword weight for courses that do not have a grade is set to the maximum value of 5. Keywords of certain (most) of the course are intertwined, so we solve this by adding up the calculated weights in the student profiles.

The student profile grows with each course he had successfully completed. We do not take into account the courses in which the student fails. For students who have more than one study on the university (e.g. the student studies master degree and in the past finished his bachelor's study on the same faculty), we modify the weights of the courses from the past studies by a multiplication coefficient of 0.2. We do this to emphasise the courses taken in recent semesters, while preserving the information from the past. When we kept the coefficient equal to 1 (without modification in the weights), the obligatory bachelor courses had a tendency to have a stronger effect than the more specialised courses. In addition, the overall student embeddings were much more alike, and it was more difficult to distinguish student specialisations.

3.3.2.2 Opportunity Embeddings

The opportunity embeddings are calculated in the exact same manner as the course embeddings – we mostly operate the same way with courses and opportunities. The only difference is that we add the keywords the company manually filled in to the calculated ones. If we use any weight for the keywords, we always add the maximum value for those added manually by the company.

3.3.2.3 Matching Algorithm

After having all student and opportunity embeddings, we can move on to the matching part of the algorithm. We cluster the opportunities using a *k-nearest-neighbour* method. We use the `NearestNeighbors` model from the Python `sklearn`[3] library. We use the cosine distance metric to calculate the distance between embeddings.

After that, we calculate the n closest opportunities to the student's embedding. In the most basic setting, the top-n recommendation is used and the closest opportunities are recommended to the student.

3.3.2.4 Explainability

During the explanation process, the algorithm is basically reversed. We find intersecting keywords (skills) for the student embedding and the opportunity embedding. Then we look up which courses the skills were gained from. In

[3]https://scikit-learn.org/stable/

this way, we can explain to the student why a certain opportunity was recommended to him, we show him the intersecting keywords and the courses from which the skills were acquired.

3.3.3 Specific Implementations

In this section, we describe some specific implementations of the previously described methods. We describe a method based on tf-idf, a method based on custom keyword selection, and their combined method. We focus on their issues and strengths.

3.3.3.1 Classic Tf-idf

The first algorithm that we go through is the basic term frequency-inverse document frequency algorithm (*tf-idf*) to extract keywords from courses and opportunities. We implement it using the `TfidfVectorizer` from the Python `sklearn` library. We use English stop words from the `NLTK` library. The vectorizer is used on a corpus containing all available courses (subjects) and all available opportunities merged in a single list.

We implement our own tokenizer that keeps only letters (not numbers). It then extracts the words using the `TreebankWordTokenizer` that are later lemmatised using the `WordNetLemmatizer`. Both are from the `NLTK`[4] library (see code snippet 3.3).

```
def preprocessing(doc):
    doc = re.sub(r"[^a-zA-Z]", " ", doc.lower())
    words = word_tokenize(doc)
    words_lemmed = [WordNetLemmatizer().lemmatize(w)
            for w in words if w not in stop_words]
    return words_lemmed
```

Figure 3.3: Tf-idf document preprocessing function

The tf-idf function is configured to return ngrams in the range from 1 word to 3 words. We keep only the top 2000 features that have document frequency lesser than 19 %. The 19 % limit was decided to remove corpus-specific stop words such as *student, computer, learn*. Overall, 75 corpus-specific stop words are removed using this threshold (see Appendix C for all removed stop words). We do not set the threshold any lower because we do not want to remove words such as "c" in the C programming language.

These features are manually checked later and more words are removed. We remove all ngrams containing the words: *student, course, able*; because

[4] https://www.nltk.org/

Table 3.1: The table of diversity results of the classic tf-idf method shows the number of distinct opportunities recommended and the number of times that the best opportunity was recommended while performing the *top-n* recommendation for all students.

n-neigbours	1	3	5	10	20
Distinct opportunities	71	106	124	153	160
Top opportunity	661	1011	1234	1457	1731

they do not represent any skill, which is the main point of this whole process. We then go through all the generated ngrams and remove even more of them such as *introduction, assignment, study* (see Appendix C for all removed ngrams). Altogether, we keep 1822 ngrams that represent skills that can be extracted from the school courses and the job opportunities.

```
stop_words = stopwords.words('english')
vectorizer = TfidfVectorizer(tokenizer=preprocessing,
        stop_words=stop_words, ngram_range=(1, 3),
        max_df=0.19, max_features=2000, sublinear_tf=True)
kw_matrix = vectorizer.fit_transform(documents.values())
```

Figure 3.4: Tf-idf Python implementation

Now, we calculate the skills of the student using the algorithm described in 3.3.2.1. The only difference is that we multiply the weight of the course (calculated from the student's grade) by the value that was earlier calculated by the tf-idf algorithm.

After that, we compose a *k-nearest neighbours* model described in Section 3.3.2.3. Explainability is done in the same way as in Section 3.3.2.4.

A minor problem we encounter with explainability while using this method is that keywords are often quite meaningless, for example: *machine, specific, idea, and learning*. Although they may be crucial for matching an opportunity with a student (for example, a *machine learning* opportunity), they are not very useful when it comes to explaining why a certain opportunity was recommended.

In Table 3.1 we show the diversity results of this method, we point out two metrics: the number of different opportunities recommended to all students when using the top-n recommendation, and the number of times the top opportunity is recommended. We can see (even with comparisons in Table 3.7 and Table 3.6) that the tf-idf method performs quite well (second to third best) in terms of diversity compared to other models.

```
'Junior Data Scientist'
# Matching keywords:
{'html', 'layer', 'task', 'database', 'big data', 'real',
'interpretation', 'activity', 'description', 'output',
'production', 'different', 'text', 'web', 'machine', 'format',
'support', 'creation', 'parsing', 'big', 'business', 'various',
'classification', 'science', 'mode', 'code', 'monitoring'}

# Matching courses with their keywords
'bi-emp': {'business', 'classification', 'support', 'output',
           'production'},
'bi-aag': {'text', 'classification', 'task', 'machine',
           'creation', 'output', 'parsing'},
'bi-big': {'big', 'big data'},
'bi-ppa': {'machine', 'interpretation'},
'bi-zdm': {'database'},
'fi-hte': {'science', 'activity', 'business'},
'bi-vwm': {'layer', 'various', 'text', 'web', 'database',
           'mode'},
'bi-prr': {'description', 'monitoring', 'real'},
'bi-xml': {'html', 'text', 'web', 'database', 'format',
           'different'},
'bi-kot': {'code'}
```

Figure 3.5: Explainability of an opportunity recommendation with the basic tf-idf method. *Matching keywords* represent the intersecting keywords of the student and the opportunity. Below that we display matched courses and the intersecting student-opportunity keywords with those courses.

3.3.3.2 Custom Keyword Method

To address the issue of explainability mentioned in the previous Section 3.3.2.4. We decided to use a limited set of keywords that will perform more intuitively during the explanation. In this section, we take the general idea from paper *Reducing cold start problems in educational recommender systems* [33] and recreate some of the parts. The work was done on the same faculty and was performed with the same (but older) data that we use in this work; therefore, the solution is tailored for our purpose and we can replicate the process of creating skill clouds for students. In the mentioned work, the skill clouds are used to recommend the students to experts who may come into contact with them. We can adjust the main idea for the recommendation of opportunities.

In the early stages, we proceed similarly to Section 3.3.3.1. We take the merged text described in Section 3.3.1 and remove all the special characters,

tokenise the text, remove stop words, and finally lemmatise the remaining words. After that we create ngrams, again ranging from 1 word to 3 words, and calculate their overall frequency in the whole corpus (school courses together with job opportunities).

After that, we take all the generated ngrams (over 130,000 unique ngrams) and match them to three different datasets of complex skills:

- Linkedin skills – we use the *Linkedin* dataset[5] of 50,000 professional skills that was published in 2019 and is composed of soft and hard skills.
- India Skills dataset – the skills are taken from the company from India, *It's Your Skill*. We use only free API capabilities to acquire about 5,700 skills.
- ACM classification skills – in paper [33] they use 400 skill categories (all from the IT domain) from which they build their ontology trees. The 400 skills were inspired by the *ACM digital library* categories[6].

We merge all the datasets together and match them with all the generated ngrams of the courses and opportunities. This is how we acquire about 3,700 distinct skills. After that, we go through all the skills and remove the stop skills that are not useful for our task (Czech, assignment, etc.). We create a skill mapper, which selects which skills are to be mapped to other skills (e.g. recommender to recommender system, recommending to recommender system). This process reduces the skill dataset to 1913 distinct skills. The skill mapper is used on both the skills dataset as well as on the ngram sets that were previously generated from the courses and opportunities.

Now we have a set of skills for each course (and for each job opportunity), and we can calculate the embeddings of the students using the algorithm mentioned in Section 3.3.2.3.

Finally, we again compose a *k-nearest neighbours* model described in Section 3.3.2.3. Explainability is done in the same way as in Section 3.3.2.4. The advantage of explainability with this method is that it is much more relatable than with the tf-idf method 3.4 because we have complete control over the ngrams that are being matched and explained later.

The problem that occurs using this custom method is that the recommendations were not diverse enough. When recommending the top five opportunities to all current students who had at least one skill (total number of 1806 students), the recommender algorithm recommended only 85 of the 166 opportunities (see Table 3.2). This diversity is significantly lower than the diversity of tf-idf, which can recommend 124 out of 166 opportunities (Table 3.1) in the same configuration (for a more detailed comparison, see Section 3.3.3.4).

[5] https://www.linkedin.com/business/learning/blog/top-skills-and-courses/the-skills-companies-need-most-in-2019-and-how-to-learn-them

[6] https://www.acm.org/publications/class-2012

The best opportunity is also recommended for 1598 out of 1806 . The reason for this behaviour is that the opportunity has very general keywords (information technology, operating system, etc.). These general skills are acquired in the mandatory courses each student has to go through during his/her studies; they can also be acquired by many different courses, resulting in a huge weight in the profile in the majority of the students.

This creates an issue where the more general keywords that are located in most of the courses (opportunities) significantly outweigh more specific keywords that may carry more interesting information about the course (opportunity).

```
'Junior Data Scientist'
# Matching keywords
{'web', 'program analysis', 'recommender systems', 'programming',
'classification', 'parsing', 'interpretation', 'big data',
'databases', 'code', 'business process'}

# Matching courses with their keywords
'bi-emp': {'recommender systems', 'classification',
           'business process', 'program analysis'},
'bi-aag': {'programming', 'classification', 'parsing',
           'program analysis'},
'bi-big': {'big data', 'databases'},
'bi-ppa': {'interpretation', 'programming'},
'bi-zdm': {'programming', 'databases'},
'fi-hte': {'business process'},
'bi-vwm': {'web', 'program analysis', 'recommender systems',
           'programming', 'databases'},
'bi-prr': {'program analysis'},
'bi-xml': {'programming', 'databases', 'web'},
'bi-kot': {'programming', 'code'}
```

Figure 3.6: Explainability of an opportunity recommendation with the custom keyword method. *Matching keywords* represent the intersecting keywords of the student and the opportunity. Below that we display matched courses and the intersecting student-opportunity keywords with those courses.

3.3.3.3 Combined Solution

After analysing the two previous solutions, we propose a combined solution that is supposed to address their weaknesses. The purpose of this solution is to maintain the limited set of available keywords while addressing the problem of diversity.

Table 3.2: The table of diversity results of the custom keyword method shows the number of distinct opportunities recommended and the number of times the best opportunity was recommended while performing the *top-n* recommendation for all students.

n-neigbours	1	3	5	10	20
Distinct opportunities	49	78	85	108	132
Top opportunity	739	1464	1598	1682	1756

Table 3.3: The table of diversity results of the combined method shows the number of distinct opportunities recommended and the number of times the best opportunity was recommended while performing the *top-n* recommendation for all students.

n-neighbours	1	3	5	10	20
Distinct opportunities	62	105	121	142	152
Top opportunity	457	725	791	1036	1270

We start by implementing a tf-idf vectorizer with all the detected features (over 130 thousand features) over ngrams of lengths ranging from 1 to 3 words. Tokenization, lemmatisation, and elimination of stop words are performed in the same manner as in the tf-idf method (see Section 3.3.3.1).

After generating this large matrix, we update this matrix using the defined keywords for each opportunity (add one to the appropriate cells). Later, we map all the generated skills using the same logic as in Section 3.3.3.2.

We basically receive the same set of skills as in the custom keyword method (see Section 3.3.3.2), but with the difference that these computed skills have the appropriate weight based on their tf-idf relevance score. After that, we replicate the work from the previous section with the only difference in the calculation of the student profile. Here, we multiply the students' grades by the relevance of the keyword before appending them to their profiles.

In summary, we observe a diversity similar to that of the pure tf-idf method while maintaining the explainability of the custom keyword method. Again, using the top five nearest-neighbour recommendations, we recommend 121 of 166 opportunities – Table 3.3 (for a more detailed comparison, see the comparison Section 3.3.3.4).

3.3.3.4 Comparison

In this section, we compare the results of the three methods mentioned previously. Mostly, we compare their ability to recommend a more diverse set of opportunities, not only to recommend the best opportunity to every student.

In Tables 3.1, 3.2, 3.3 we compare two metrics for the changing number of recommendations for each student. We measure the number of distinct opportunities that are recommended to all students separately and the number of times the top opportunity is displayed in the top-n recommendation for all students. As can be seen, the pure tf-idf method provides the most diverse results when it comes to the distinct opportunities metric and is closely followed by the combined method. However, the combined method provides almost the same results while beating the pure tf-idf method in the top opportunity metric and while preserving the more relatable explainability.

Taken together, the combined method seems to provide overall the most reasonable results. It maintains very good diversity, and the explainability is much more explanatory and understandable to the user.

3.3.4 Neural Embeddings

In this section, we focus on experimenting with recommendations based on neural embedding matching. As mentioned in Section 1.1.6, neural embeddings have improved significantly in the last few years and nowadays are a state-of-the-art solution when it comes to document recommendation, information retrieval, and NLP tasks. That is why we want to experiment with this solution as well, even though the explainability of this solution is hardly interpretive. [18, 17]

We have to create an algorithm where neural embeddings can be used. We create a neural embedding for each course and for each job opportunity we have. The neural embeddings are generated using an `OpenAI`[7] library and its API. This library is not completely free, but there is a limit of $ 18 that is sufficient for our purposes.

We query for more than 700 embeddings (representing each opportunity and each course; the courses with the same warp and the same description were queried only once). `OpenAI` offers four types of embedding lengths:

- Ada – 1024 dimensions,
- Babbage – 2048 dimensions,
- Curie – 4096 dimensions,
- Davinci – 12288 dimensions.

We use the Babbage embedding with 2048 dimensions for our purposes. We would use more-dimensional ones, but they are more expensive, and for the sake of the experiment, 2048 dimensions should be enough. `OpenAI` also offers more engines that differ in their purpose:

[7] https://openai.com/

- *Similarity embeddings* – good at capturing similarity between two or more pieces of text.
- *Text search embeddings* – these models focus on measuring the relevance of long documents to a short search query (the embeddings for documents and queries differ).
- *Code search embeddings* – similar logic to text search embeddings. There are two types of engine, one for embedding code and the other one for embedding code snippets which are to be retrieved.

Interestingly, when we generate the embeddings based on the description, name, etc., the neural embeddings we create are very similar to each other. This happens because all the courses and opportunities are IT related, meaning they all share a similar part of the latent space.

3.3.4.1 Course-opportunity Matrix

After having all the embeddings, we compose a *k-nearest-neighbour* model using all courses (their neural embeddings). In the model space, we find the k courses closest to each opportunity. We use the dot product as a metric for our *kNN* model (the *L2* norms of all generated neural embeddings equal 1 meaning that the dot product equals the cosine distance). We actually use $(1 - dot(E_a, E_b))$ because we want the closest courses to be near each other. The $dot(E_a, E_b)$ denotes the dot product of two arbitrary course embeddings (E_a, E_b)

Using the calculated distances for each course $i \in \{1 \dots m\}$ and each opportunity $j \in \{1 \dots n\}$ where m represents the number of courses and n represents the number of opportunities, we create a course-opportunity matrix $OPP \in \mathbb{R}^{m,n}$:

$$OPP_{i,j} = dot(E_i, E_j) \tag{3.1}$$

where E_i represents the embedding of the i-th course and E_j represents the embedding of the j-th opportunity. For each opportunity (each column of the OPP) we keep only k greatest values – only k courses represent each opportunity.

3.3.4.2 Student-course Matrix

For each student $h \in \{1 \dots l\}$ and each course $i \in \{1 \dots m\}$ where l represents the number of students and m represents the number of courses, we create a student-course matrix $STU \in \mathbb{R}^{l,m}$:

$$STU_{h,i} = grade(Student_h, Course_i) \tag{3.2}$$

where $grade(Student_h, Course_i)$ represents an arbitrary function that maps the student grades in specific courses to a numerical value. We derive the grade value of the student using the following list: ($A = 5$, $B = 4$, $C = 3$, $D = 2$, $E = 1$, $Z = 5$). The grade Z means the course ended only with credit (without any additional grade). As an example on another function mapping student's grade in a course to a numerical value could be a percentile function.

3.3.4.3 Recommendation

The recommendation values are computed by multiplying the matrices mentioned above (3.1 and 3.2) in the following manner (see Equation 3.3) and resulting in the recommendation matrix $REC \in \mathbb{R}^{l,n}$ where l represents the number of students and n represents the number of opportunities.

$$REC = STU \cdot OPP \tag{3.3}$$

The advantage of this matrix is that we can extract the best opportunities for each student as well as the best candidates for each opportunity.

$$recommend\_stu(student_i, k) = argmax(REC_{i:}, k) \tag{3.4}$$

$$recommend\_opp(opportunity_j, k) = argmax(REC_{:j}, k) \tag{3.5}$$

The opportunity recommendations for the student i are placed in the row i of the REC matrix (see Equation 3.4). The best candidates for opportunity j are placed in the column j of the REC matrix (see Equation 3.5). The function $argmax(list, k)$ returns the indices of k highest values in the $list$ (row or column).

3.3.4.4 Results of Neural Embedding Recommendation

We create a few experiments that focus mainly on the diversity of courses used for recommendation and the diversity of opportunity recommendation for students.

The parameter on which we focus is k used during the construction of the *k-nearest-neighbours* model of courses. We consider the following values of k: 5, 10, 20, 40, 100, 200, 420. In total, there are different 420 courses (courses that have the same description and name are taken as the same course), so for k equal to 420 the OPP matrix is full of non-zero values (unless the zero is calculated).

As we can see in Table 3.4 the overall *opportunity diversity* is high for almost all cases compared to the keyword methods (see Tables 3.3, 3.2, 3.1). A surprising observation is that we have to set $k = 100$ so that most courses are taken into account during the recommendation.

We try to identify some kind of similarity among the least considered courses, but cannot find any common trait. There are all possibilities of

Table 3.4: Diversity of the number of courses considered and the final recommendation of opportunities. *Considered courses* – the number of non-zero rows in the OPP matrix. *Top course* – maximal number of non-zero values of all columns in OPP matrix. *Opportunity diversity* – the number of distinct recommended opportunities while performing 5 best opportunities recommendation for all students. *Top opportunity* – the number of times the top opportunity is recommended.

k	5	10	20	40	100	200	420
Considered courses	144	198	257	325	390	418	420
Top course	33	50	80	104	141	146	146
Opportunity diversity	133	142	143	137	141	134	36
Top opportunity	1064	1021	809	972	795	813	1354

Table 3.5: The table of diversity results of the neural embeddings method shows the number of distinct opportunities recommended and the number of times the best opportunity was recommended while performing the *top-n* recommendation for all students.

n-neighbours	1	3	5	10	20
Distinct opportunities	104	143	158	165	166 (max)
Top opportunity	279	537	577	960	1221

courses: theoretical and practical courses, undergraduate and graduate courses, optional and compulsory courses.

An issue that is very common in association with neural embeddings (and almost all machine learning methods based on neural networks) is the interpretation and explanation of the decision-making process. In our case, the only thing we can certainly explain is the "middleman" in the form of the course on whose basis the recommendation is made. The reason why some courses are closer to certain opportunities in the neural embedding space is hard to interpret for us and remains a bit of a mystery. [40]

The explainability can be done using classical methods such as *tf-idf* or the *custom keyword method* mentioned in Section 3.3.3.2 where we extract describing keywords. This method has nothing in common with the neural embeddings, but at least it gives a user some distant reason why he/she might like the given result.

As we can see from Table 3.5 the number of distinct opportunities is the highest compared to classical methods (Tables 3.3, 3.2, 3.1). And when we recommend 20 opportunities to each student, we achieve the recommendation of all currently available opportunities.

Table 3.6: Comparison of distinct opportunities of all methods: we compare the number of distinct opportunities that were recommended to students when recommending the top n opportunities to each student.

n-neighbours	1	3	5	10	20
Custom keywords	49	78	85	108	132
Basic tf-idf	71	106	124	153	160
Custom keywords with tf-idf	62	105	121	142	152
Neural embeddings	104	143	158	165	166

Table 3.7: Comparison of top opportunity recommendation of all methods: we compare the number of times the most-fitted opportunity was recommended during a top n opportunity recommendation for each student.

n-neighbours	1	3	5	10	20
Custom keywords	739	1464	1598	1682	1756
Basic tf-idf	661	1011	1234	1457	1731
Custom keywords with tf-idf	457	725	791	1036	1270
Neural embeddings	279	537	577	960	1221

3.3.5 Overall Comparison

In Tables 3.6, 3.7 and Figures 3.7, 3.8 we can compare the diversity in the recommendations of all purposed methods. In Table 3.6 and Figure 3.7, we compare the number of distinct opportunities that the method recommends to students while performing a top-n recommendation for each of them. Diversity increases with a higher number of distinct opportunities. In Table 3.7 and Figure 3.8, we compare the number of times the most frequent opportunity is recommended while making a top n recommendations for each student. In this case, the diversity increases with decreasing numbers.

As we can see, the custom keyword method with tf-idf and the basic tf-idf method perform the best in terms of diversity of all keyword-based methods. The custom keyword method with tf-idf (the combined method) has the upper hand in explainability. Overall, the neural embeddings are capable of the most diverse recommendation in terms of the number of distinct opportunities, in terms of top opportunity recommendations, it is practically equal to the custom keyword method with tf-idf.

The measured diversity can tell us only the capabilities of a certain method to separate students and opportunities, it does not give us the information about the quality of given recommendations.

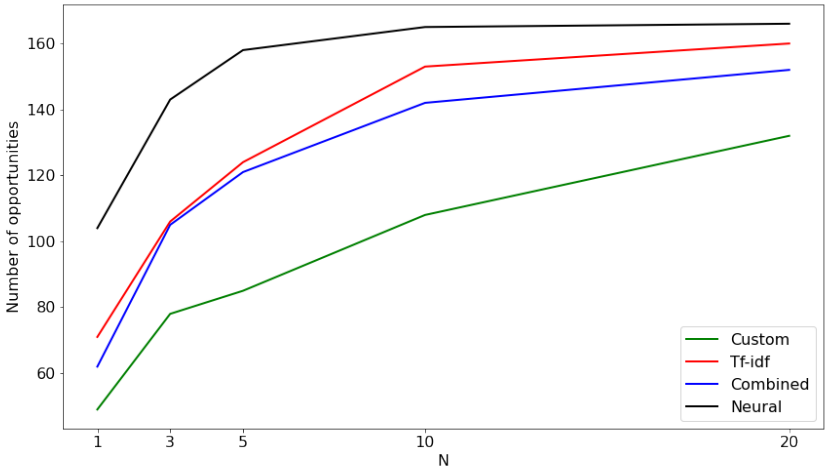


Figure 3.7: Dependence of the different opportunity diversity on N where N represents the number of top.n recommendations for each student. The top.n recommendation was made for 1806 students with a total of 166 opportunities.

3.3.6 Summary and Model Selection

If we focus only on the diversity capabilities of the purposed methods, it would seem appropriate to select the neural embedding method as the best. However, diversity does not necessarily mean the best recommendation. The neural embedding method does not take into account the quality in which individual descriptions are written. One of the best opportunities that was recommended using this method was an opportunity without a description. It is really difficult to interpret such a decision.

Ultimately, we decide to test and analyse the **custom keyword method with tf-idf** due to its high qualities in the case of diversity of recommendations and due to its best explanation ability.

For even a better diversity of recommendations, we can use the algorithm described in Section 3.4 which is derived from the methods purposed in the paper [37].

3.4 Job Recommendation Constraints

In Table 3.7, we compare the number of times a certain opportunity is recommended to students during a top-n recommendation. As we can see, the best opportunity is recommended a lot of times for each scenario. This is partially caused by the students of the first year because they do not have the space

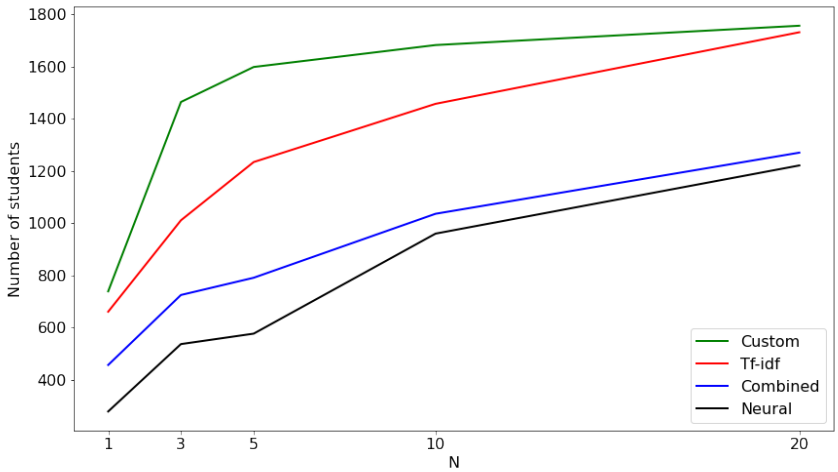


Figure 3.8: The dependence of the number of times the top opportunity was recommended on N where N represents the number of top-n recommendations for each student. The top-n recommendation was made for 1806 students with a total of 166 opportunities.

to select any optional courses. In general, they mostly all study the same courses, resulting in very similar profiles.

Nevertheless, we need to address this problem because the job recommendation is different from the recommendation of, for example, movies, because the opportunity cannot satisfy all users who apply for it. The open position can be filled by only one or a few students (see Section 1.2.1 for a more detailed description of the problem). This is the reason why we address the issue of recommending the same opportunity to a large number of students.

In the theoretical part Section 1.4, we refer to a solution purposed in [37]. This work aims at creating a general solution that considers multiple adjustable results on an item/user basis (item capacities and user propensities). In our work, we deal with a specific subproblem of the work of [37]. The item capacities are practically the same for most of the opportunities, and they do not change over time – once the position is filled, it can usually be removed from the RS (if the position is reopened, we can consider it as a new position in regards to the item capacity). User propensities are also somewhat specific, because the goal is usually to find one single opportunity for every student. In our case, it might be more appropriate to derive the user's propensity based on his actions; for example, a subscribed student might be more likely to make a contact with the opportunity than an unsubscribed one.

We implement a solution that aims to distribute the opportunity recommendations by setting a maximum number of students to whom the opportunity can be recommended on the first page (we do not limit the visibility of the opportunity on further pages or on the first page during user custom filtering).

We do that by implementing some kind of *round robin* system for recommendation. At first, we shuffle all students for whom the recommendation is done, so not to create alphabetical or any other disadvantages. After that, we iterate the list and add the top three recommendations to all students sequentially. The only limitation is that each opportunity can be recommended only a certain number of times (the limit differs based on the number of students for whom the recommendations are made and the number of all opportunities). The process is repeated two more times (for three and four more recommendations, respectively) while maintaining and updating the same list. If an opportunity reaches a maximum number of recommendations in the first round of *robin round*, it cannot be recommended again in subsequent rounds.

3.5 Explanation

In this section, we summarise the explanation capabilities of individual methods that are mentioned in the previous sections. We compare the explanation of the following methods: *basic tf-idf*, *custom keyword method with tf-idf*, *neural embedding method*. We do not mention *custom keyword method* separately because it is similar to *custom keyword method with tf-idf* in terms of explanation (the only difference is in matching keywords).

In Figures 3.9 and 3.10 we display the recommendation of the same opportunity to the same student using the *custom keyword method with tf-idf* and the *basic tf-idf method*. As can be seen, there are much fewer keywords matched in Figure 3.9 (*custom keyword method with tf-idf*) compared to Figure 3.10 (*basic tf-idf*), but all the keywords matched are much more complex, such as *machine learning, web mining, data analysis*. Some of the keywords in tf-idf methods are also complex, such as *machine learning, data mining*, but there are many almost meaningless keywords such as *direct, look, way, create*.

We tried to reduce the number of non-skill keywords in tf-idf (see the appendix C), but there are still a lot of them. We could remove even more, but in that way we would be nearing the custom keyword method, which would result in the loss of the simplicity of the basic tf-idf method.

We describe a very basic explanation outline for the neural embedding method in the previous Section 3.3.4, where the explanation is done using the calculated keywords from either the basic tf-idf method or the custom keyword method. This is a suboptimal method, because the explanation is based on a completely different process, and in some cases there might be no matching keywords.

```
'Python Developer junior'
# Matching keywords
{'machine learning', 'qt', 'devops', 'designing', 'spark',
'web', 'bootstrap', 'python', 'data analysis', 'web mining',
'automation', 'agile', 'data mining'}

# Matching courses with their keywords
'ni-mvi': {'machine learning', 'web mining', 'data mining'},
'ni-pdd': {'web mining', 'data mining', 'web'},
'ni-sz1': {'machine learning'}, 'ni-pon': {'machine learning'},
'ni-adm': {'machine learning', 'web mining', 'data mining'},
'ni-bml': {'machine learning', 'bootstrap'},
'ni-sz2': {'machine learning'}, 'ni-scr': {'bootstrap'},
'ni-pdb': {'spark'},
'bi-dbs': {'web'},
'bi-pa2': {'qt'},
'bi-tjv': {'devops', 'web'},
'bi-bez': {'designing'},
'bi-pyt': {'python'},
'bi-zum': {'machine learning', 'web mining', 'data mining'},
'bi-vwm': {'web mining', 'data mining', 'web'},
'bi-si1.2': {'agile'},
'bi-vzd': {'data analysis', 'web mining', 'data mining',
'machine learning'},
'bi-svz': {'automation'}
```

Figure 3.9: Explainability of an opportunity recommendation with the custom keyword method. *Matching keywords* represent the intersecting keywords of the student and the opportunity. Below that we display matched courses and the intersecting student-opportunity keywords with those courses.

In general, we evaluate that the *custom keyword method with tf-idf* is capable of providing the best explanation for its recommendations. The matched keywords are understandable and easily interpretable for students.

```
'Python Developer junior'
# Matching keywords
{'numerical', 'bootstrap', 'end', 'least', 'mining',
'data mining', 'agile', 'power', 'web', 'may', 'lead', 'create',
'designing', 'web application', 'technical', 'exact', 'idea',
'machine', 'perspective', 'u', 'machine learning', 'direct',
'graphical', 'data analysis', 'python', 'learning', 'automation',
'look', 'way', 'input'}

# Matching courses with their keywords
'ni-umi': {'lead'},
'ni-mpi': {'numerical', 'machine'},
'ni-mvi': {'learning', 'machine learning', 'mining',
'data mining', 'machine'},
'ni-pdd': {'mining', 'data mining', 'web'},
'ni-sz1': {'learning', 'machine', 'machine learning'},
'ni-pon': {'numerical', 'learning', 'machine learning', 'least',
'machine'},
'ni-pdp': {'technical'},
'ni-adm': {'learning', 'machine learning', 'mining',
'data mining', 'machine'},
'ni-ccc': {'create', 'technical', 'idea', 'way', 'input'},
'ni-scr': {'bootstrap'},
'ni-kop': {'exact'},
'ni-pdb': {'machine'},
'bi-cao': {'power', 'look'}, 'bi-pa1': {'input'},
'bi-zma': {'way'},
'bi-ps1': {'way', 'input'},
'bi-uli': {'learning', 'machine', 'input'},
'bi-sap': {'machine', 'input'},
'bi-dbs': {'way', 'direct'},
'bi-lin': {'u', 'way'},
'bi-pa2': {'may'},
'bi-aag': {'machine'},
'bi-zdm': {'power'},
'bi-tjv': {'end', 'web', 'create'},
 ...
```

Figure 3.10: Explainability of an opportunity recommendation with the basic tf-idf method. *Matching keywords* represent the intersecting keywords of the student and the opportunity. Below that we display matched courses and the intersecting student-opportunity keywords with those courses.

Chapter 4

Analysis

In this chapter, we focus on analysing the performance of our recommender systems. We perform two stages of A/B testing with different testing groups, and we describe the established results.

4.1 Testing Environment

The testing was carried out under the company Unico and its websites. The testing was carried out mainly on the website of the Faculty of Information Technology of the Czech Technical University in Prague. The first part of the testing was started during the COFIT[8] student job fair that lasted for more than three weeks.

The widget on the website with the opportunity recommendation was placed as an HTML `iframe` on the website of the faculty[9] (see Figure 4.1) and it was placed in the faculty Microsoft Teams platform (see Figures 4.2 and 4.3).

4.1.1 Measured Interactions

We measure the following interactions: detail views, purchases (apply for a job using the apply button), and number of recommendations. We also collected interaction with the website such as text search, filtering using predefined filters and subscriptions.

The interactions were captured using the Recombee platform and the Google Analytics tool[10]. An issue that occurred using this configuration (the implementation as a website widget) was that the interaction capturing tools needed to use the third-party cookies that can be disabled by some users (e.g. the Safari web browser disables the third-party cookies by default). This

[8] https://fit.cvut.cz/cs/spoluprace/pro-studenty/veletrh-cofit
[9] https://fit.cvut.cz/cs/spoluprace/pro-studenty/nabidky-prace/partneri-a-sponzori
[10] https://analytics.google.com/analytics/web/

created a minor inconvenience, which resulted in the capture of fewer interactions.

4.2 Testing

We randomly create groups of users who receive different types of recommendation. In the first part of the testing, we create two groups and test the significance of the interaction-only recommendation compared to no recommendation at all.

In the second part of the testing, we measure the importance of the personalised recommendation based on the student profiles derived from their past studies. There is a smaller number of students in the second part of the testing because we need an explicit consent from users so that we can legally use their profile for the use case of personalised recommendation.

It must be mentioned that in the evaluation of the A/B tests all user interactions performed during application testing by non-student users were deleted.

4.3 A/B Testing - Part One

In this section, we describe the setting and results of the first part of the A/B testing. At first, we created 2 groups of users:

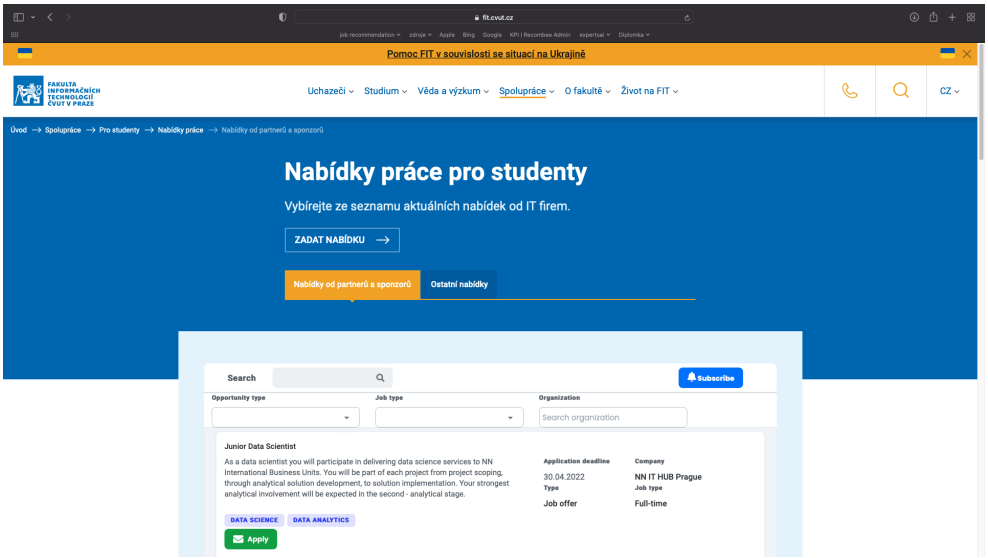


Figure 4.1: The recommendation system placement on the faculty's website

- **Group A** received a personalised recommendation solely based on their previous interactions and the relationships between the opportunities (the attributes of the items were used without the use of the attributes of the user).

- **Group B** received a random list of available opportunities extracted from the database without any logic.

Recommendations for group A were made using the Recombee platform. We should bear in mind that before the testing had started, there were no item-user interactions recorded on the platform, and in the early stages the recommender was suffering from the cold start problem. All interactions made by users in group B were also sent to the recommender to increase its performance and mitigate the effect of the cold start problem as quickly as possible.

The recommender system received the following interactions: detail views, purchases, custom searches, and keyword searches. By keyword search, we mean the interaction in which a user clicks on the prepared keywords and filters the opportunities.

4.3.1 Results

We evaluate multiple types of interaction (detail views, purchases, filters, searches), but detail views and purchases are considered the most important.

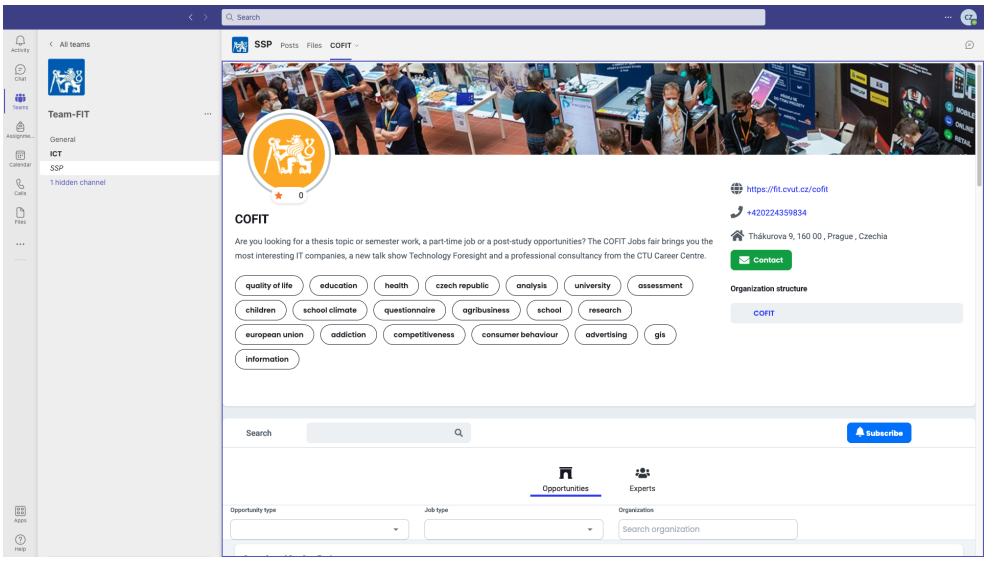


Figure 4.2: The landing page of the widget of the RS system in Microsoft Teams.

In Table 4.1, it can be seen that there is a notable difference in the number of total (and average) detail view interactions. The number of users who performed at least one detail view is practically the same 66 to 69, but the average number of detail views per user changes from 4.1 (group A – with recommendations) to 2.8 (group B – without recommendation). In Figure 4.4 we can observe the improvement process of the interaction RS. In the early stages we can observe the cold start problem, which is gradually solved by acquiring more interactions over time. The cumulative sum ratio of **group A** is calculated using Equation 4.1, where $I_{A,t}$ represent the number of interactions of **group A** from the beginning to time t.

$$csr(A,t) = \frac{I_{A,t}}{I_{A,t} + I_{B,t}} \tag{4.1}$$

In Table 4.3, we measure the statistics of the position of the opportunities with which they interact. It seems that the mean and median are the same for both tested groups.

The most significant conversion is the purchase interaction, which in our case is harder to interpret, considering the very low number of overall interactions of this type (see Table 4.2). In total, we measured only 24 purchase interactions performed by 9 users, of which 15 interactions were performed by 6 users in group B (without recommendation). This would seem in favour of the no-recommendation scenario, but the number of interactions is so low that we do not dare to come to any substantiated conclusions.

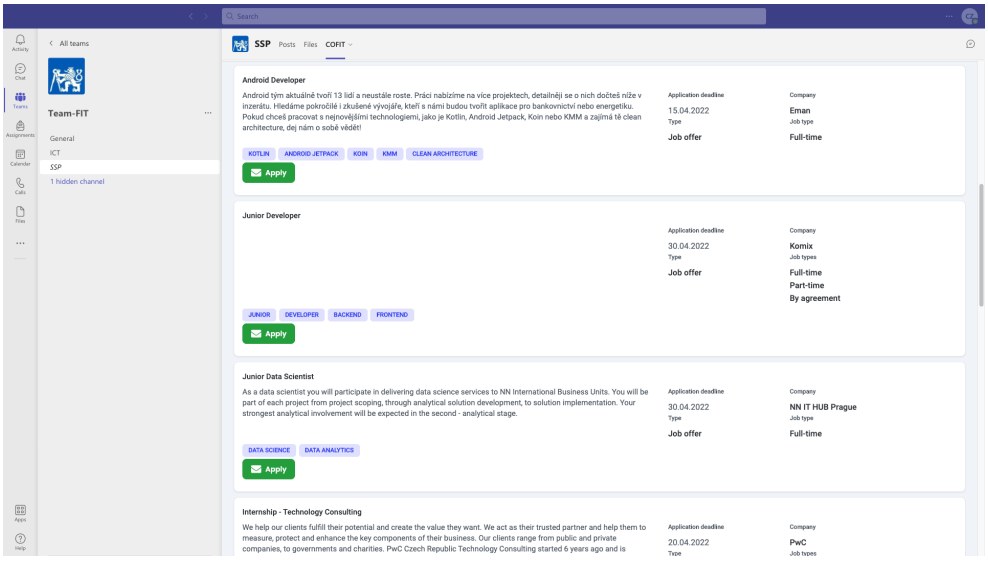


Figure 4.3: Recommendations example of the widget of the RS system in Microsoft Teams.

Table 4.1: Detail view statistics from the first A/B testing: *count* – number of measured interactions; *users* – number of users who performed the interaction (users were randomly distributed 50/50); *average* – average number of interactions per user; *Q25, Q50, Q75* – quantiles of the number of interactions.

	count	users	average	Q25	Q50	Q75
A	269	66	4.1	1	2	4.75
B	195	69	2.8	1	1	4

Table 4.2: Purchase statistics from the first A/B testing: *count* – number of measured interactions; *users* – number of users who performed the interaction (users were randomly distributed 50/50); *average* – average number of interactions per user;

	count	users	average
A	9	3	3
B	15	6	2.5

Table 4.3: Position of detail view statistics from the first A/B testing: *count* – number of interactions; *mean* – mean position of clicked interaction; *std* – standard deviation of the interaction position; *max* – the highest position of the interaction made; *median* – median of the position.

	count	mean	std	max	median
A	269	4.9	3.1	10	4.0
B	195	4.5	3.1	10	4.0

Table 4.4: Performed searches in the first part of the A/B testing. One full text search can mean more sub-searches if the user types slowly. *Count* – number of interactions, *users* – number of users who performed the interaction. *Q25, Q50, Q75* – quantiles of the number of searches.

	count	users	Q25	Q50	Q75
A	209	44	1	3	6
B	257	50	2	3.5	7

Table 4.5: Filter interaction statistics from the first A/B testing: *count* – number of interactions, *users* – number of users who performed the interaction. *Q25, Q50, Q75* – quantiles of the number of searches.

	count	users	Q25	Q50	Q75
A	828	88	2	5	11
B	662	89	2	5	11

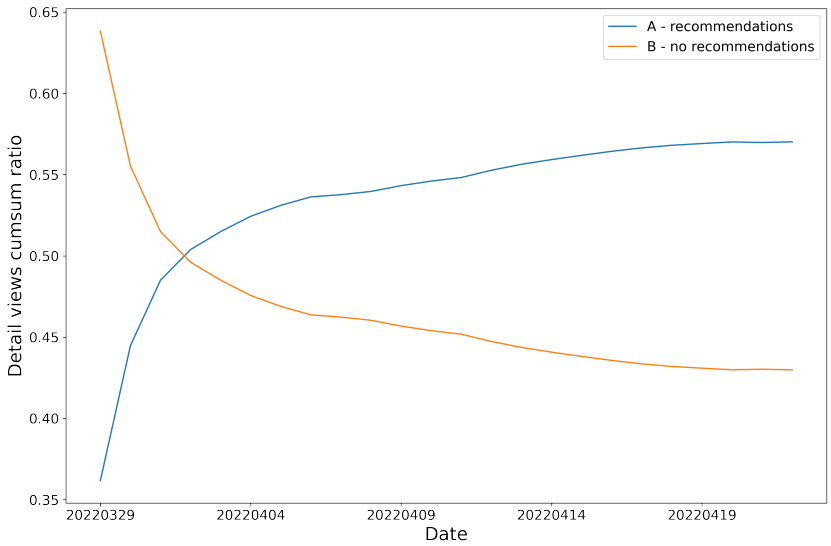


Figure 4.4: Cumulative sum ratio of detail views in the time span from 29.3.2022 to 23.4.2022. The cumulative sum for **group A** is calculated using the following Equation 4.1

4.4 A/B Testing – Part Two

The second part of the users A/B testing focusses on personalised recommendations based on the student profile. As mentioned in Section 2.3.1 the recommendations are carried out under the supervision of Unico, and we have to have the student's consent to be able to use his/her data for personalised recommendation purposes. Therefore, we can work only with students who have subscribed to personalised recommendations on the widget.

Personalised recommendation is performed offline, saved in the database, and later sent to the user via email. Each user has the possibility to unsubscribe from any email received.

We assume a certain bias that is connected with the subscription. Therefore, to eliminate bias, users of all groups are taken from the set of subscribed students.

4.4.1 Groups

In this part of A/B testing, we create three separate groups to test which recommendation method performs the best.

- **Group C** receives a personalised recommendation solely based on their previous interactions and the relationships between the opportunities (only the item attributes are used). It is the same setting as in **Group A** in the previous A/B tests, the difference being that here we use only subscribed students.

- In **Group D**, students receive a personalised recommendation based only on their profile (courses completed). The first ten items are taken on the basis of this recommendation, and the rest are filled in randomly. The recommended items are the same for the duration of the email newsletter; the first 10 items cannot change until a new recommendation is made with a different setting or different opportunities is made.

- **Group E** receives the most complex recommendations. The recommendations are a combination of the methods of **group C** and **group D**. First, both the interaction-based RS and the personalised RS based on the student profile make recommendations. The recommendations are then rearranged on the basis of the combined position in both recommendation lists (the two positions are summed).

We presume the following behaviour of the individual groups. We anticipate that **group C** may suffer from the cold start problem. The cold start problem should affect new users only, because the historical interactions are kept in the Recombee platform, hence the RS itself should not suffer from the mentioned problem.

The **group D** should be able to deal with the cold start problem easily, however, the system is not capable of adaptation to the user's behaviour once he makes a few interactions.

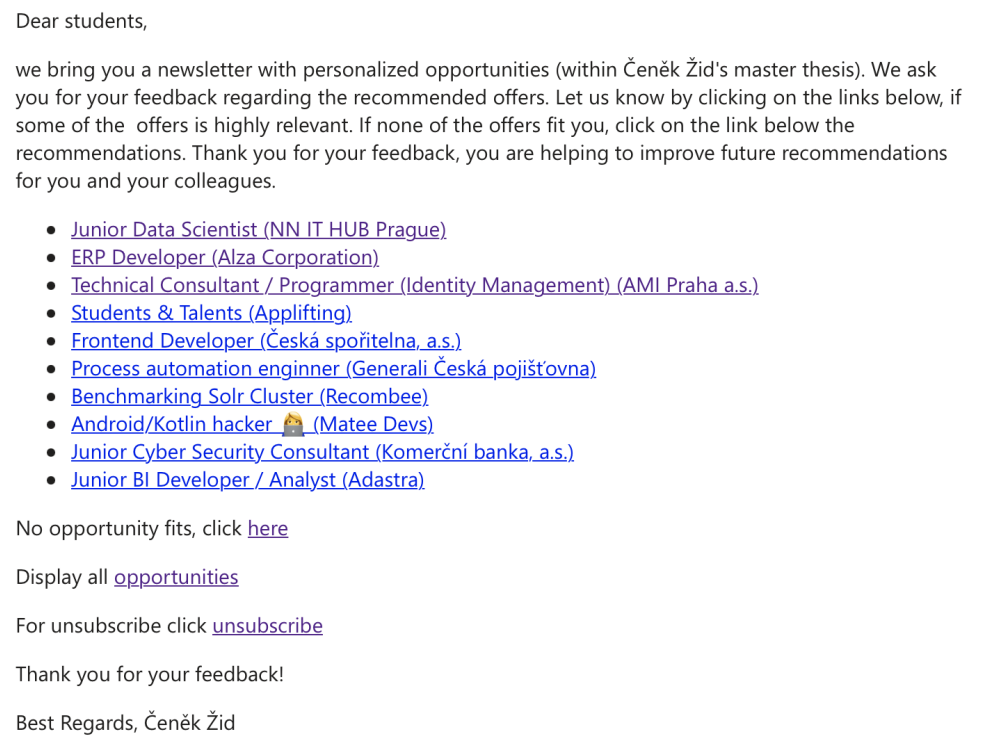
The **group E** should be able to handle both above-mentioned issues. In the early stages, it can recommend relevant items based on the student's profile, and later it can adapt based on the users' interactions (searches, detail views, etc.).

4.4.2 Detailed Parameters

Overall, we have 51 subscribed students. These students are randomly distributed into 3 groups equally, 17 students in each group. We perform an offline recommendation and send an email to students with the top 10 recommendations asking for their feedback. There is a link for each recommendation detail view, a link to the widget with more recommendations, a link to unsubscribe, and a link with the information that no links were relevant. We measure all the following clicks (see the email example in Figure 4.5).

Although we save the user identification token during subscription, most of the people had no previously measured interactions on the Recombee platform. We assume that this might be because the request for subscription was sent

via email to all students. We suspect that many students open their emails on the phones, whereas most of the interactions captured on the system were made on the desktop. The impact that it has on this experiment is that most of the recommendations for groups C are covered by the best sellers. It also has an impact on group E, where the recommendations are combined with best sellers and not with personalised recommendations.



Dear students,

we bring you a newsletter with personalized opportunities (within Čeněk Žid's master thesis). We ask you for your feedback regarding the recommended offers. Let us know by clicking on the links below, if some of the offers is highly relevant. If none of the offers fit you, click on the link below the recommendations. Thank you for your feedback, you are helping to improve future recommendations for you and your colleagues.

- Junior Data Scientist (NN IT HUB Prague)
- ERP Developer (Alza Corporation)
- Technical Consultant / Programmer (Identity Management) (AMI Praha a.s.)
- Students & Talents (Applifting)
- Frontend Developer (Česká spořitelna, a.s.)
- Process automation enginner (Generali Česká pojišťovna)
- Benchmarking Solr Cluster (Recombee)
- Android/Kotlin hacker 🤷 (Matee Devs)
- Junior Cyber Security Consultant (Komerční banka, a.s.)
- Junior BI Developer / Analyst (Adastra)

No opportunity fits, click here

Display all opportunities

For unsubscribe click unsubscribe

Thank you for your feedback!

Best Regards, Čeněk Žid

Figure 4.5: Newsletter example

4.4.3 Results

From the 51 subscribers, 32 students opened the email and 28 of them clicked on at least one link. The numbers were measured using the Google Analytics platform and the XCAMPAIGN mailing service[11].

No student clicked on the link saying they were not satisfied with any recommendation, and no student performed an unsubscribe action. We evaluate this positively along with the number of clicks and the number of emails opened.

We can see in Table 4.6 that we have captured the most clicks interaction in groups D and E (respectively, 37 by 11 users and 33 by 10 users). The overall

[11] https://www.xcampaign.info/switzerland-en/

Table 4.6: Second A/B testing statistics for groups C, D, E. We measure the following statistics: *Users* – number of users, *Click* – number of clicks on the links, *Clicks median* – median of user clicks, *Clicks/user* – average number of clicks per user, *Mean I* – the average index of the opportunities clicked, *Mean I* – the mean index of the opportunities clicked.

Group	Users	Clicks	Clicks median	Clicks/user	Mean I	Median I
C	7	18	3	2.6	3.9	**2**
D	11	37	3	**3.4**	**3.5**	3
E	10	33	3	3.3	4.4	4

best click-per-user ratio has group D with 3.4 clicks/user tightly followed by group E with 3.3 clicks/user. These values are higher compared to group C with only 2.6 clicks/user.

It must be noted that 6 out of 7 users that performed interactions in group C had no previous interactions captured, therefore they received the best seller recommendation list with no further profiling.

The best seller recommendation (group C) has the best click position median of 2 (the indices are calculated from 0). The worst click position is measured in group E (combined recommendation) with a mean 4.4 and a median of 4.

We conclude that the best-seller recommendation works because most of the users click on the higher-placed opportunities. The results of Table 4.6 suggest that the personalised recommendation based on the student profile works well and outperforms the best seller recommendation. This means that it is better to recommend based on the student profile compared to the best sellers for the new users (if the student data are available). This also helps to increase the diversity by not recommending the same set of items to all new users.

Overall, the most complex method with the combined recommendation in group E performs well (almost the best) in terms of the number of clicks, meaning that the re-ranking does not have a negative impact. This is an important result because this method can be used on the widget, where it is necessary to provide a changing list of recommendations.

Overall, we dare say that all the tested recommendation settings perform well and are capable of a relevant recommendation. The better performance of group D compared to group C indicates that it is useful to use the student profile for recommendation purposes. Although group E did not perform the best in the position of opportunities clicked, the important part is that it managed to recommend relevant opportunities to the students.

The position in which the opportunities are sorted in group E can be a matter of future testing and can be fine-tuned by hyperparameter adjustments. In this test, we only averaged the recommendation of both combined methods.

We would like to experiment with a more complex combination now that we see that the recommendation based on the student profile works. We could introduce a parameter that adjusts weights on both recommendation lists based on the number of interaction the user has performed (e.g., giving more weight to the student profile personalised recommendation in the early stages, when there are no interactions captured, and later mitigate its effect in favour of interaction-based recommendation).

4.5 AI Fairness

In the work, we deal with data processing of university and student data. The data should be protected and treated in a secure manner. In this thesis, we work with the data provided by the faculty, and for the sake of this work, we use the whole dataset with the anonymised data.

During the second part of the A/B testing that focused on personalised recommendations for students, we needed an explicit consent of all included students. The consent was acquired using a subscribe button on the widget website. After a subscription, we could use the data for recommendation purposes. Data were manipulated with according to European GDPR standards.

In terms of personalised recommendation using the student profile, we acquired some subscriptions using non-university emails which created a minor inconvenience because we could not use their student profile, because we did not know who the person was. The subscribed person could even be from a different university.

There may be a slight bias on the basis of the available data. We have only data provided by the faculty, meaning that for students enroled in the graduate programme who previously studied at different faculty/university, we have no data about their past studies. This gives them a disadvantage and they may receive suboptimal recommendations.

We try to recommend as many opportunities as possible. This may result in a recommendation of an opportunity to a non-optimal student, because we only work with students of one faculty and the optimal student can study on a completely different university.

4.6 Future Work

We have tested a few models, but essentially we chose the tf-idf model with custom keywords (ngrams). This model performs very well and is very easy to interpret. If we wanted to implement our algorithm in different faculties or even universities in the future, we would choose the basic tf-idf method in the current situation.

The tf-idf method performs well enough and is capable of easier fine-tuning with external experts who might not have an understanding of recommenda-

tion processes. It is even possible to transform the model later to a model with a limited set of skills in the same way as was done on our faculty.

In our work, we have focused more on the classic methods because it was easier to fine-tune them. For our second experiment, we had very few students; therefore, a lot of fine-tuning had been done before the experiment based only on the expert's judgment.

We would like to focus more on the capabilities of neural embeddings in the future because it is a state-of-the-art solution when dealing with NLP tasks. However, we would need to acquire a larger set of student profiles to properly tune the necessary parameters.

In our second experiment, we can see the differences in performance of all the tested methods. All of the methods perform well and are capable of recommending relevant opportunities. With more traffic in the widget, we would like to implement all three methods to work in parallel and see which methods the users tend to choose and tend to interact with.

Conclusion

We analysed various types of recommender systems in the theoretical part of our work. We described the possibilities of natural language processing and implemented some of them to further improve our goal, which was the creation of a recommender system for students that would recommend job opportunities for them.

The recommender system was successfully implemented in a scalable style. Our architecture can make quality recommendations even without the student data, but the capabilities of this system can be improved by implementing a re-ranking module that utilises student's university data. We also provided ideas about possible implementations in different faculties or universities.

We implemented and compared different types of student profiling based on the previously analysed methods. These methods were thoroughly compared based on their quality of recommendations, the quality of explanations, and the general capabilities of student profiling. Overall, the results suggested that the prime method is the tf-idf-based custom keyword method that was inspired by the paper *Reducing Cold Start Problems in Educational Recommender Systems.* [33]

In the final part of our work, we created two experiments that support the contributions of our work. The results of the first experiment indicated the improvement in recommendation quality that correlates with the use of an interaction-based recommender system compared to no recommendation at all. The final experiment confirmed the improvements with the use of the student profile and its positive impact on personalised recommendations.

We would like to experiment with a wider variety of parameters in the student profiling in the future, such as the weight adjustments for compulsory and optional courses or the weight adjustments depending on the time a course was studied. We would like to further examine the capabilities of neural-based embeddings and ideally create experiments on more student profiling methods on a larger set of students.

Bibliography

1. ROCCA, Baptiste. *Introduction to recommender systems* [online]. Towards Data Science, 2019 [visited on 2022-01-23]. Available from: `https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada`.
2. TOMÁŠ ŘEHOŘEK, Pavel Kordík. *Doporučovací systémy, úvod* [online]. Faculty of Information Technology, Czech Technical University in Prague, 2020 [visited on 2022-01-23]. Available from: `https://courses.fit.cvut.cz/NI-ADM/lectures/files/MI-ADM-10-en-slides.pdf`.
3. RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. Introduction to Recommender Systems Handbook. In: *Recommender Systems Handbook*. Ed. by RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha; KANTOR, Paul B. Boston, MA: Springer US, 2011, pp. 1–35. ISBN 978-0-387-85820-3. Available from DOI: `10.1007/978-0-387-85820-3_1`.
4. BALABANOVIĆ, Marko; SHOHAM, Yoav. Fab: Content-Based, Collaborative Recommendation. *Commun. ACM*. 1997, vol. 40, no. 3, pp. 66–72. ISSN 0001-0782. Available from DOI: `10.1145/245108.245124`.
5. PINELA, Carlos. *Content-Based Recommender Systems* [online]. Medium, 2017 [visited on 2022-03-27]. Available from: `https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235`.
6. WU, Jackson. *Knowledge-Based Recommender Systems: An Overview* [online]. Medium, 2019 [visited on 2022-03-27]. Available from: `https://medium.com/@jwu2/knowledge-based-recommender-systems-an-overview-536b63721dba`.
7. RUIJT, Corné de; BHULAI, Sandjai. *Job Recommender Systems: A Review*. arXiv, 2021. Available from DOI: `10.48550/ARXIV.2111.13576`.

8. VERMA, Yugesh. *A Guide to Building Hybrid Recommendation Systems for Beginners* [online]. Developers Corner, 2021 [visited on 2022-04-20]. Available from: https://analyticsindiamag.com/a-guide-to-building-hybrid-recommendation-systems-for-beginners/.

9. B.THORAT, Poonam; GOUDAR, R.; BARVE, Sunita. Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. *International Journal of Computer Applications*. 2015, vol. 110, pp. 31–36. Available from DOI: 10.5120/19308-0760.

10. BURKE, Robin; ROBIN. Hybrid Web Recommender Systems. In: 2007, vol. 4321. ISBN 978-3-540-72078-2. Available from DOI: 10.1007/978-3-540-72079-9_12.

11. CHIANG, Jeffrey. *7 Types of Hybrid Recommendation System* [online]. Medium, 2021 [visited on 2022-04-18]. Available from: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8.

12. SUBRAMANIAN, Dhilip. *Content-Based Recommendation System using Word Embeddings* [online]. Towards AI, 2020 [visited on 2022-04-18]. Available from: https://pub.towardsai.net/content-based-recommendation-system-using-word-embeddings-c1c15de1ef95.

13. KOEHRSEN, Will. *Building a Recommendation System Using Neural Network Embeddings* [online]. Towards Data Science, 2018 [visited on 2022-04-13]. Available from: https://towardsdatascience.com/building-a-recommendation-system-using-neural-network-embeddings-1ef92e5c80c9.

14. SIMHA, Anirudha. *Understanding TF-IDF for Machine Learning* [online]. Capital One, 2021 [visited on 2022-04-08]. Available from: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

15. QIU, Jiezhong; MA, Hao; LEVY, Omer; YIH, Scott Wen-tau; WANG, Sinong; TANG, Jie. Blockwise Self-Attention for Long Document Understanding. *CoRR*. 2019, vol. abs/1911.02972. Available from arXiv: 1911.02972.

16. YAO, Mariya. *10 Leading Language Models For NLP In 2021* [online]. Top Bots, 2021 [visited on 2022-04-18]. Available from: https://www.topbots.com/leading-nlp-language-models-2020/.

17. DIAZ, Fernando; MITRA, Bhaskar; CRASWELL, Nick. Query Expansion with Locally-Trained Word Embeddings. *CoRR*. 2016, vol. abs/1605-.07891. Available from arXiv: 1605.07891.

18. *word2vec* [online]. TensorFlow Core, 2021 [visited on 2022-04-20]. Available from: https://www.tensorflow.org/tutorials/text/word2vec.

19. MIROSLAV ČEPEK, Pavel Kordík. *Text and Graph Representations* [online]. Faculty of Information Technology, Czech Technical University in Prague, 2021 [visited on 2022-04-05]. Available from: https://courses.fit.cvut.cz/MI-MVI/media/lectures/06/NI-MVI-06-lecture.pd.

20. *Chapter 6. Document embeddings for rankings and recommendations* [online]. Manning, 2019 [visited on 2022-04-20]. Available from: https://livebook.manning.com/book/deep-learning-for-search/chapter-6/1.

21. COLYER, Adrian. *The amazing power of word vectors* [online]. The morning paper, 2016 [visited on 2022-03-31]. Available from: https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/.

22. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention Is All You Need. *CoRR*. 2017, vol. abs/1706.03762. Available from arXiv: 1706.03762.

23. SCHMELZER, Ronald. *GPT-3* [online]. Tech Target, 2021 [visited on 2022-04-16]. Available from: https://www.techtarget.com/searchenterpriseai/definition/GPT-3.

24. MUSTAFA, Zaki. *Image Based Product Recommendation System* [online]. Medium, 2021 [visited on 2022-04-16]. Available from: https://zakim.medium.com/image-based-product-recommendation-e1bfa29e508.

25. *Artificial Intelligence Powered Recommender as a Service* [online]. 2021 [visited on 2022-01-25]. Available from: https://www.recombee.com/.

26. WIGGERS, Kyle. *Google launches Recommendations AI ecommerce tool in public beta* [online]. The Machine, 2020 [visited on 2022-04-12]. Available from: https://venturebeat.com/2020/07/22/google-launches-recommendations-ai-in-public-beta/.

27. *Recommendations AI* [online]. Google Cloud, 2022 [visited on 2022-04-12]. Available from: https://cloud.google.com/recommendations.

28. *Google recommendations AI* [online]. Qubit, 2022 [visited on 2022-04-12]. Available from: https://www.qubit.com/google-recommendations-ai-capability.

29. KENTHAPADI, Krishnaram; LE, Benjamin; VENKATARAMAN, Ganesh. Personalized Job Recommendation System at LinkedIn: Practical Challenges and Lessons Learned. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. Como, Italy: Association for Computing Machinery, 2017, pp. 346–347. RecSys '17. ISBN 9781450346528. Available from DOI: 10.1145/3109859.3109921.

30. RENN, Robert; STEINBAUER, Robert; TAYLOR, Robert; DETWILER, Daniel. School-to-work transition: Mentor career support and student career planning, job search intentions, and self-defeating job search behavior. *Journal of Vocational Behavior*. 2014, vol. 85, pp. 422–432. Available from DOI: 10.1016/j.jvb.2014.09.004.

31. BREAUGH, James A. Employee recruitment: Current knowledge and important areas for future research. *Human Resource Management Review*. 2008, vol. 18, no. 3, pp. 103–118. ISSN 1053-4822. Available from DOI: https://doi.org/10.1016/j.hrmr.2008.07.003. Critical Issues in Human Resource Management Theory and Research.

32. PETERSON, Anna. On the Prowl: How to Hunt and Score Your First Job. *Educational Horizons*. 2014, vol. 92, no. 3, pp. 13–15. Available from DOI: 10.1177/0013175X1409200305.

33. KUZNETSOV, Stanislav; KORDÍK, Pavel; ŘEHOŘEK, Tomáš; DVOŘÁK, Josef; KROHA, Petr. Reducing cold start problems in educational recommender systems. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 3143–3149. Available from DOI: 10.1109/IJCNN.2016.7727600.

34. LIU, Rui; RONG, Wenge; OUYANG, Yuanxin; XIONG, Zhang. A hierarchical similarity based job recommendation service framework for university students. *Frontiers of Computer Science*. 2016, vol. 11. Available from DOI: 10.1007/s11704-016-5570-y.

35. ZHOU, Qing; LIAO, Fenglu; CHEN, Chao; GE, Liang. Job recommendation algorithm for graduates based on personalized preference. *CCF Transactions on Pervasive Computing and Interaction*. 2019, vol. 1. Available from DOI: 10.1007/s42486-019-00022-1.

36. KORDÍK, Pavel; KUZNETSOV, Stanislav. Mining Skills from Educational Data for Project Recommendations. In: HERRERO, Álvaro; BARUQUE, Bruno; SEDANO, Javier; QUINTIÁN, Héctor; CORCHADO, Emilio (eds.). *International Joint Conference*. Cham: Springer International Publishing, 2015, pp. 617–627. ISBN 978-3-319-19713-5.

37. CHRISTAKOPOULOU, Konstantina; KAWALE, Jaya; BANERJEE, -Arindam. *Recommendation under Capacity Constraints*. arXiv, 2017. Available from DOI: 10.48550/ARXIV.1701.05228.

38. SUBRAMANIAN, Anant; PRUTHI, Danish; JHAMTANI, Harsh; BERG-KIRKPATRICK, Taylor; HOVY, Eduard H. SPINE: SParse Interpretable Neural Embeddings. *CoRR*. 2017, vol. abs/1711.08792. Available from arXiv: 1711.08792.

39. *DeepL Pro* [online]. DeepL, 2022 [visited on 2022-04-08]. Available from: https://www.deepl.com/pro?cta=header-prices.

40. KOEPPE, Arnd; BAMER, Franz; SELZER, Michael; NESTLER, Britta; MARKERT, Bernd. Explainable Artificial Intelligence for Mechanics: Physics-Explaining Neural Networks for Constitutive Models. *Frontiers in Materials*. 2022, vol. 8. ISSN 2296-8016. Available from DOI: 10.3389/fmats.2021.824958.

Appendix A

Acronyms

AI Artificial intelligence

BERT Bidirectional encoder representations from transformers

GRPR General data protection regulation

IT Information technology

KNN K-Nearest neighbours

NLP Natural language processing

NLTK Natural language toolkit

NN Nearest neighbours

PCA Principal component analysis

RS Recommender system

REI Regional economic index

RFI Regional familiarity index

TF-IDF Term frequency-inverse document frequency

Appendix B

Contents of Enclosed CD

- `README.md` ..... content description
- `matching` ..... jupyter notebooks
  - `skills.ipynb` ..... custom skills extraction
  - `skill_computation.ipynb` ..... custom student skills computation
  - `matching.ipynb` ..... custom keyword matching
  - `tf_idf_matching.ipynb` ..... tf-idf algorithm
  - `tf_idf_matching_preselected_skills.ipynb` ... combined algorithm
  - `embedding_matching.ipynb` ..... neural embeddings matching
  - `final_recommendation.ipynb` ..... A/B testing split
  - `embeddings` ..... neural embeddings extraction
    - `embeddings.ipynb` .. jupyter with extraction of neural embeddings
  - `src` ..... folder with python modules
- `google_analytics` ..... Google Analytics data processing
  - `ga_processing.ipynb` ..... analysis of the captured data
- `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `Zidcenek_mater_thesis.pdf` ..... the thesis text in PDF format

Appendix C

Removed Stop Words

Automatically removed stop words: basic knowledge, use, course, application, skill, principle, goal, basic, used, project, also, area, method, communication, presentation, learn, tool, structure, development, gain, able, algorithm, problem, architecture, data, function, student learn, nan nan, overview, level, c, understand, nan, knowledge, system, solution, process, implementation, fundamental, type, example, theory, practical, network, computer, time, theoretical, analysis, programming, modern, get, management, module, security, language, part, aim, design, software, nanstudents, information, test, based, technique, using, advanced, processing, control, introduction, technology, experience, environment, work, student, model.

Manually removed stop words: introduction, assignment, successful, searching, czech, consultation, gain, foundation, regular, component, learn, aim, sequence, issue, cover, supply, terminal, related, emphasis, area, point, consultation, revision, construct, class, grade, new, bi, bie, given, goal, ii, mi, understand, main, provide, provides, study, topics, topic, basic, nan, apply, work, consultation, etc, give, many, upon, acquire, ability, small, among, know, taught, c c, well, due, want, useful, republic, good, czech republic, company, best, bachelor, strong, seminar, value, take, non, e g, mostly, life, however, last, working, even, university, need, definition, including, example, field, simple, like.