Project Advanced Web Technologies Final Report: Learning Technologies - Competence Extraction via ML / NLP

Inga Strelnikova
Faculty IV
TU Berlin
Berlin, Germany
inga.strelnikova@campus.tu-berlin.de

Jialun Jiang
Faculty IV
TU Berlin
Berlin, Germany
jialun.jiang@campus.tu-berlin.de

Stefan Vachenauer
Faculty IV
TU Berlin
Berlin, Germany
stefan.vachenauer@campus.tu-berlin.de

***Abstract*—Comparing competencies across national borders presents its obstacles. The EU-ESCO database of the European Union is intended to remedy this by providing standardized descriptions of competencies. These competencies can be acquired by completing courses. There are also already standards for their course descriptions (DIN PAS1045). But how can a potential student know which competencies are covered by which course and vice versa? In a first step, Natural Language Processing and Machine Learning are used to extract the competencies from the course descriptions. Then, a graphical database is built to link the extracted competencies with the matching course descriptions. Using a Restful API, the potential student should then be able to make the necessary queries and get the required information.**

***Index Terms*—Learning Technologies, Competence Extraction, Machine Learning (ML), Natural Language Processing (NLP)**

I. Introduction

The world gets more digitized, and computer-assisted every day. This development also does not stop in the way we learn new competencies. Every student would like to maximize the learning output by completing a course. Therefore, it's necessary to use information and communication technology (ICT) to reach this goal. In the world of education, this is called technology enhanced learning (TEL) [1].

The way employers recruit new employees is changing too. More and more digital tools are used and the frequency of job changes is increasing due to higher geographic and occupational agility. To handle this development and make competencies comparable and standardized, the European Union implemented a database called ESCO. This database should help people with various questions like which skills are needed for a specific occupation [2]? For international comparison, it's also necessary to have a standard for the description of courses. This is regulated by the DIN PAS 1045 standard. This standard defines the content and description minimum and data formats for exchange [3]. There are several databases with course descriptions based on this standard like the *Weiterbildungsdatenbank Berlin Brandenburg (WDBB)*.

Despite these standards, these databases have no connection to each other. This project aims to create a link between both databases and to find out which competencies from the ESCO database are covered by courses from the textitWeiterbildungsdatenbank Berlin Brandenburg and vice versa. For this purpose, the course descriptions will be analyzed with the help of Natural Language Processing and Machine Learning and matching competencies will be assigned. This will then be stored in a graphical database and made available to the student via a rest API. The following questions should be answerable for the student:

- What courses cover the competency with this name?
- What competencies does the course with this name cover?

II. Related Work

Before the project starts, it is necessary to research the existing literature and methods in the field, [4] gives us the big picture of this topic, introducing three types of potential methods for competence extraction: simple statistics approach, linguistics approach, and Machine Learning approach. In a simple statistics approach, we don't need the training data, and it uses statistical methods: word frequency, TF*IDF, word co-occurrence, NGram statistical information, etc. to extract key information from the text. A linguistics approach uses the linguistic features of text, lexical analysis, syntactic analysis discourse analysis, and so on with the text. In the Machine Learning approach, keywords extraction is seen as supervised learning, where a model is trained to classify the information that needs to be extracted and the information that does not need to be extracted using ML algorithms like Naive Bayes, Support Vector Machine, etc. [5] shows us how to extract competences from research publications, which is very similar to the topic of this project. It uses Named Entity Recognition provided by General Architecture for Text Engineering (GATE) system. Similar to [5], [6] introduces an Entity Recognition-based approach, but it uses a focus on the recognition of entities in highly specialized domains, called Ontology-based Entity Recognition, which we experimented with and found to be a good fit for our project. In addition, some very cutting-edge approaches are also reviewed, such as the Universal Sentence Encoder method developed by Google

[7], based on the deep neural network of the transformer architecture, which learns that the 512-dimensional embedding of a sentence containing some of the high-level features.

III. Approaches

In this section, we will introduce in detail the methods and techniques used in implementing competence extraction. It will show how to pre-process the data, what NLP algorithms to use, how to interact with the graph database, and how users can use the service through the RESTful API.

A. Architecture

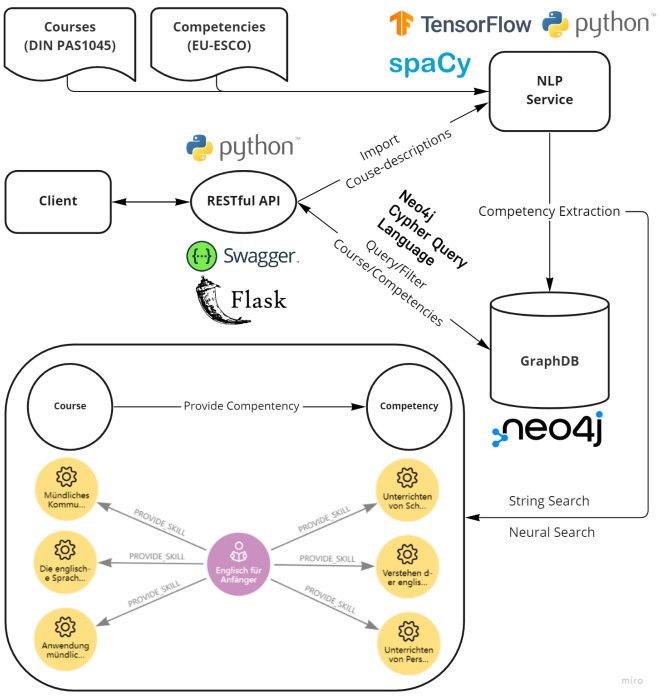

Fig. 1. Project Architecture

In illustration 1 the architecture of our project is displayed, which shows the big picture of our technology stack, the components of our project, and how they interact. Our main programming language is python. In order to get course descriptions, we use the *Weiterbildungsdatenbank Berlin Brandenburg* which meets the requirements for the DIN standard PAS1045. For competencies, we use the ESCO database of the European Union. For the user interaction, we use Swagger UI and the Restful API is implemented with Flask based on Python. The data storage is formed by a graph database from Neo4j for course descriptions and their related competencies. For general natural language processing, we have used the framework spaCy to process and analyze the text. Our machine learning approach for competence extraction is based on TensorFlow.

B. Data Preprocessing

In this section, we will introduce the approaches and tools we used in data pre-processing, in order to extract useful information from the dataset and provide it in a suitable format for the NLP algorithm in the next section III-C. Its source code is given in the form of a Jupyter Notebook and its relative address is `src/Preprocessing.ipynb`.

Two different algorithms introduced in III-C requires different format of data:

- Modified Ontology-based Entity Recognition: this algorithm is based on string matching, so we need to remove as much irrelevant information as possible. We defined a `process_ER` function to pre-process it. spaCy, an open-source software library for advanced natural language processing, and `de_core_news_lg`, spaCy's natural language processing pipeline for the German language is used to tokenize text, remove unwanted symbols, stop words, and pure digits. Please note that here we do not directly remove all combinations of symbols and letters or digits and letters, because some competences are such combinations, such as C# and 3D. Then we use spaCy's `lemma` function to perform lemmatization to the result from the previous step and convert the final result to lowercase.
- Universal Sentence Encoder: this algorithm directly converts the original sentences into embeddings, so we only need to segment the document into sentences in a suitable manner by using `process_NN`.

And two datasets are used in this project, one is the course dataset based on the DIN PAS1045 standard and the competence dataset under the ESCO framework:

- Course Dataset: the DIN standard PAS1045 was defined by the *Deutsches Institut für Normung* e.V. in 2004. The term PAS stands for Publicly Available Specification. It describes minimum standards of the content of education databases. PAS gives also recommendations for formats for the electronic exchange of this data based on the minimum standard Data Exchange for Training Information (DefTIS) [3].
  The raw file format of the course dataset is XML, the structure is very complex, and it contains a lot of information that is not needed in competence extraction, so we define an `import_course` function to import the XML file, extract the useful information and convert it to CSV format.
  The `import_course` function is using the XML file stored in the same repository in the `data` folder to create a tree using XML file. After that, we go through every course and only take the information necessary for the extraction later on, which is the course name, the course id, and the course description, creating a list from that information and a CSV file from the list after.
  The file path can be adjusted in the future for new XML files, as well as information that is needed for the extraction.
  In order to create the CSV file, it is needed to start the virtual environment first (on macOS it is the command `source bin/activate` from the main folder with

the code). Then it is needed to run the code once, with `python3 src/course_import.py` from the same folder.

After we extract useful information which are course id, course name and course description using `import_course` and `pandas` from raw course dataset, we combine the `course_name` column and the `course_description` column together as the `course_info` column, and we apply `process_ER` and `process_NN` to this new column, which will be provided to our two NLP algorithms from section III-C respectively.

Note that you will see many courses with the same name and description in the processed course dataset, but they actually have different course IDs, this is because for simplicity we only extracted `course_name` and `course_description`, but they may differ in other attributes like location and time, so they are still different courses. You may change the `import_course` function to import other attributes if you want.

- Competence Dataset: the term ESCO stands for European Skills, Competencies, Qualifications, and Occupations. It is the multi-language standard of the European Union. The goal is to define and classify occupations and competencies for the labor market of the European Union. This common standard should help to make the market more efficient and integrated. The dataset is free of charge [8].

  Compared with the complex XML format of the course dataset, we can directly use `pandas` to extract the concept URL, preferred label, and description from the competence dataset which is already CSV format.

  Then we apply `process_ER` to `preferred_label` column to get `labels_processed` which will be used to the Modified Ontology-based Entity Recognition algorithm.

  We also combine the `preferred_label` column and the `description` column together as the `info` column and without applying `process_NN` which will be used in the Universal Sentence Encoder algorithm, since one row of competence data has only a label and a short description, and we treat it as one sentence.

C. Competence Extraction (NLP)

This section is about the algorithms used for competence extraction, we used two types of algorithms. One is based on Named Entity Recognition and the other is based on Machine Learning. The source code for the competence extraction algorithms is available at `src/NLP.ipynb`.

1) Modified Ontology-based Entity Recognition: One of the most useful approaches in the early phase of the project was based on the Ontology-based Entity Recognition (OER) [6]. It can be seen as an improved version of Named Entity Recognition, and we made further modifications to the OER to meet the needs of the project, as you can see in illustration III-C1:

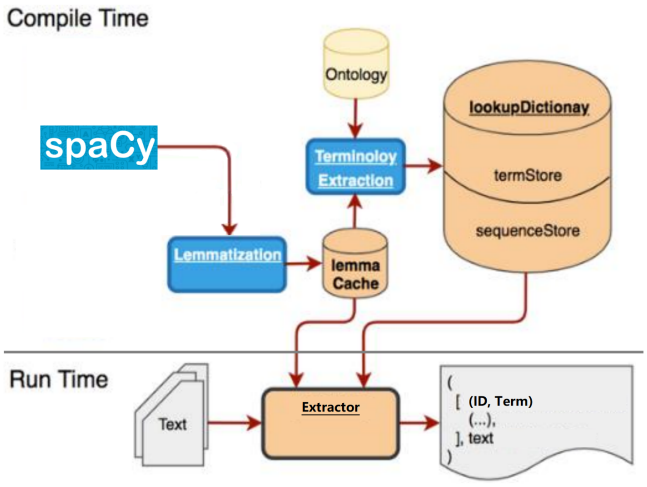

Fig. 2. Modified Ontology-based Entity Recognition Architecture

As mentioned in III-B, for the lemmatization step, the original paper used various dictionaries like WordNet, and we directly use spaCy's `lemma` function, which greatly simplifies this step. The ontology in our project is the set of preferred labels from the competence dataset and we use it to build the lookup dictionary. Note that we omitted the annotator part and replaced it with the extractor, which can be seen as a subset of the annotator since we only need to extract the competencies without annotating them in the original text. So the Modified Ontology-based Entity Recognition basically consists of two phases, one is the compilation phase and the other is the extraction phase.

- In the compilation phase, first, as mentioned above, we use spaCy's `lemma` function for lemmatization and form the lemmaCache. Then a so-called two-layered recognizer is used along with lemmaCache to build a lookupDictionary. The first recognizer layer is used to look up lemmatized terms based on the termStore. The second layer is used for the corresponding term sequences based on the sequenceStore. In our Python implementation, they are two dictionaries with the same names, namely: `termStore` and `sequenceStore`. `termStore` uses vocabularies from preferred skill labels as keys of the dictionary, and each vocabulary has its unique URI as the dictionary's value, `sequenceStore` are complete preferred skill labels, which use a tuple key, where each element in the tuple is the URI of the vocabulary of skill labels taken from `termStore`. We completed the construction of these two dictionaries by traversing the skill labels twice, That's all for the compilation phase.
- In the extraction phase, the tokenized text (`course_info_ER` mentioned in III-B) is scanned from the beginning until a word contained in the `termStore` is reached. Starting from this word a lookahead (the `check_candidates`) function in the source code) is performed searching for the longest sequence of words, which are contained in

the `termStore`. As soon as a subsequent term is not included in the `termStore`, the `check_candidates` method is used to find all sequences still contained in the `sequenceStore` by using URIs (the tuple key mentioned the compilation phase).

After completing the implementation of the Modified Ontology-based Entity Recognition algorithm, we test the algorithm with some sample courses and the results are shown in Figure 3.

	course_name	skill_label
0	Aktuelles Arbeitsrecht 2022	Arbeitsrecht
1	Ambulante Pflege - Rechtssicher Handeln und Ha...	Risikomanagement
2	Ambulante Pflege - Rechtssicher Handeln und Ha...	Datenschutz
3	Aufgaben des gesetzlichen Betreuers - Zur Refo...	planen
4	Aufgaben des gesetzlichen Betreuers - Zur Refo...	sich selbst darstellen
...	...	...
150	Die persönliche Nachfolgestrategie - die optim...	Kommunikation
151	Die persönliche Nachfolgestrategie - die optim...	planen
152	Die persönliche Nachfolgestrategie - die optim...	Reflexion
153	Die persönliche Nachfolgestrategie - die optim...	Geschäftsmodell
154	Die persönliche Nachfolgestrategie - die optim...	Kommunikation

Fig. 3. Modified Ontology-based Entity Recognition Results

As you can see that the Modified Ontology-based Entity Recognition algorithm can extract some short and critical competencies, but we think this result is not good enough because it has an obvious flaw, that is, it cannot extract complex competencies, so we turned into the Machine Learning.

2) Universal Sentence Encoder: When talking about machine learning algorithms, the first problem we face is how to convert text into features that can be used by machine learning algorithms. While working on this project, I was taking a machine learning course at TU Berlin [9] where I learned how machine learning works with textual data. Because word w are abstract symbols that do not have a numerical representation, they need to be embedded into one such representation: $e[w]$. There are mainly three types of embedding methods: one-hot encoding, kernel PCA embedding, and learned to embed. The one-hot embedding represents each word with indicator functions, it will be very high-dimensional in our project, since we work with two big datasets. Kernel PCA uses some structured kernel $k(w, w')$ to measure similarity between words. And learned embedding starts with a randomly initialized embedding $e[w] \in \mathbb{R}^d$ and learn the embedding on the supervised task directly or on some related unsupervised task: $e[w] \leftarrow e[w] - \gamma \cdot \partial\varepsilon/\partial e[w]$. We decide to use learned embedding because it can be trained with some powerful deep models to extract some high-level features that meet our needs for extracting complex capabilities.

After some investigation, we find the Universal Sentence Encoder model provided by TensorFlow [7], which uses state-of-the-art technology in Natural Language Processing (NLP) — the Transformer neural network. It was proposed in the most popular Machine Learning paper "Attention is All You Need" [10] in recent years and is increasingly the model of choice for NLP problems. The model constructs sentence embeddings using the encoding sub-graph of the transformer architecture [10]. This sub-graph uses attention to compute context-aware representations of words in a sentence that take into account both the ordering and identity of all the other words. The context-aware word representations are converted to a fixed-length sentence encoding vector by computing the element-wise sum of the representations at each word position. Finally, it outputs a 512-dimensional vector as the sentence embedding [7].

As mentioned in III-B, `skills_info` and `courses_info_NN` are feed to this model. First, we use this model to calculate a 512-dimensional vector for each row of `skill_info_processed` as the embedding and store it in the dictionary `skill_info_embedings`. Here an implementation trick is used, computing the entire `skills_info` directly would cause out of memory, so we use batch training, computing 500 rows of embeddings at a time in our implementation, on machines with different performances this number should be adjusted accordingly.

Once we have all skill embeddings, for each row of `courses_info_NN`, we calculate the inner product between each sentence to all skill embeddings, where the inner product describes the similarity between skill embedding and course sentence embedding, then we sort the inner product values and return the skills with the highest inner product values.

There are two hyperparameters that need to be found in this approach:

- `skill_threshold`: for similarities between sentences in courses info and skills info, if it has value above this threshold, it will be considered as course-related skills.
- `top_n`: the number of skills with the highest similarities returned among skills with a similarity above the threshold (for each sentence in a course title and description).

We use grid search to create results with different hyperparameters settings and the quality of the returned results is evaluated manually. In our case is we think the optimal hyperparameters settings is `skill_threshold, top_n = (0.4, 10)`.

Similar to III-C1, we find the same sample courses for testing, and the results are shown in Figure 3.

As you can see, this time we are able to extract complex competencies.

However, we are still not satisfied with the result, because if we match only a certain sentence in a piece of `courses_info_NN` to the relevant competencies, we will ignore the context of the `courses_info_NN`, so there are a lot of competencies that are misclassified to the course, for that we decide to use a coarser-grained embedding: we compute the entire piece of `courses_info_NN` as an embedding, by which we greatly improve the precision of our algorithm, but this also brings another problem: the number of competencies to be found is also greatly reduced. Obviously, this kind of matching to the entire piece of `courses_info_NN` is too

	course_name	skill_label
16	Aktuelles Arbeitsrecht 2022	kollektives Arbeitsrecht
17	Aktuelles Arbeitsrecht 2022	Arbeitserlaubnisse beantragen
18	Aktuelles Arbeitsrecht 2022	Arbeitsverträge aushandeln
19	Aktuelles Arbeitsrecht 2022	Individualarbeitsrecht
20	Aktuelles Arbeitsrecht 2022	Gesundheit und Sicherheit am Arbeitsplatz
...	...	...
416	Der optimale Bewerbungs- und Auswahlprozess	Workshops zur Arbeitssuche organisieren
417	Die persönliche Nachfolgestrategie - die optim...	Daten für unternehmenspolitische Entscheidunge...
418	Die persönliche Nachfolgestrategie - die optim...	Marketingplan für Schuhwerk umsetzen
419	Die persönliche Nachfolgestrategie - die optim...	Daten für unternehmenspolitische Entscheidunge...
420	Die persönliche Nachfolgestrategie - die optim...	Marketingplan für Schuhwerk umsetzen

Fig. 4. Universal Sentence Encoder Results

strict, so we think of a way to combine these two ways, we introduce a new hyperparameter:

- `course_threshold`: for similarities between `skills_info` and `courses_info`, if it has a value above this threshold, the skill will be not considered to be too far from the context of `course_info`, so it is introduced to filter out skills that are similar to sentences in the `course_info` but do not match it's context at all, and in our final result of the Universal Sentence Encoder algorithm, we set it to `0.2`.

The solution for our problem is finally to combine the results of the Modified Ontology-based Entity Recognition III-C1 with the Universal Sentence Encoder III-C2.

D. Database

In this section, we will explain the technical selection of the database and the implementation of interaction with the database. Its source code is also given in the form of a Jupyter Notebook and its relative address is `src/Neo4J.ipynb`

For the storage of course descriptions and related competencies, we used the fully managed graph database Neo4J Aura. This database is easy to set up, and administrate, globally available, and offers its own query language called Cypher. Additionally, Neo4j is a widely used graph database, It has very complete documentation and tutorials. Graph databases are using edges and vertices to store data. Edges and vertices are also referred to as relations and edges. Compared to relational models, where each data point can be stored individually in the same table and each association points to a connected table, a graphical model characterizes each data point as a node and their connections to others as a relationship [11]. The advantage of a graph database is therefore that speed of queries is no longer related to the amount of data, but to a concrete number of relations between the data. A graphical database like neo4j can be up to 1.000 times faster compared to a relational database [12].

For the implementation of interaction with the database, we use the object-oriented design and structure given by the Neo4J official documentation [13], where the class`APP` contains all functions that are required to interact with the database:

- `__init__`: used to establish a connection to the database, it is called automatically when the APP object is created, for this you need to enter your authentication information for this function.
- `close`: used to disconnect from the database.
- `create_skills`: used to create competence nodes from the competence dataset.
- `create_courses`: used to create course nodes from the course dataset.
- `create_relations`: used to create relationships between courses and skills based on results from two NLP algorithms
- `create_relation`: used to create a relationship between one course to one skill provided by the user.
- `find_skills`: used to find skills covered by a course through course id.
- `find_courses`: used to find courses containing a skill through competence URI.

Note that the aforementioned Cypher query language is used for all CRUD operations on the database, which is inspired by SQL and is intuitive, as shown in the figure below:

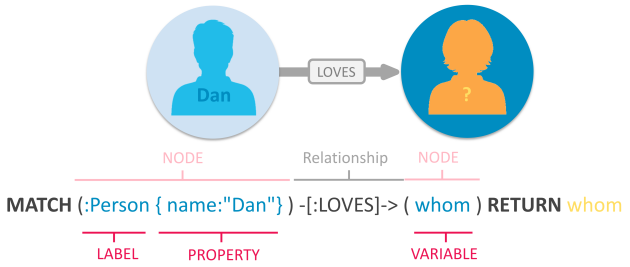

Fig. 5. Cypher Sample

We completed this part by referring to the introduction and tutorial at [14]. It is worth mentioning that in the initialization and filling of the database, we mainly use the `MERGE` operation, which can avoid the importing of duplicate course nodes or competency nodes, but this also leads to the data of more than 200,000 relationships. When uploading to this free instance, it takes a very long time.

Illustration 6 shows an excerpt of some nodes and how they are displayed in the Neo4J web application. You can see the nodes (yellow) and their relations (arrows) to one or more competencies (purple).

In our database initialization, we uploaded four files:

- `all_courses.csv`, including course id, course name, course description, by using `create_courses`
- `all_skills.csv`, including concept URI, preferred label, description, by using `create_skills`
- Using `create_relations` to upload extracted relationships between courses and competences:
  - `all_relations_ER.csv`: including course id, concept URI, extracted by Modified Ontology-based Entity Recognition.

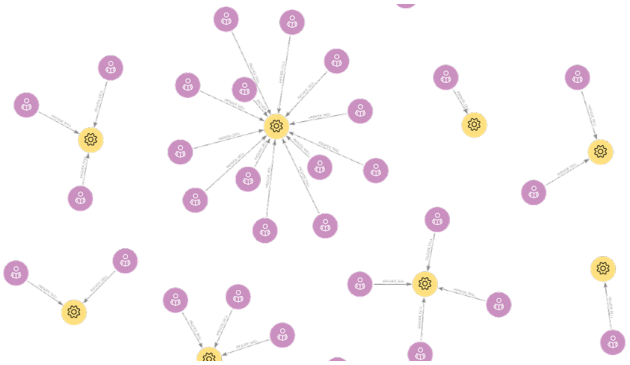

Fig. 6. Graph Database

– `all_relations_NN.csv`: including course id, concept URI, extracted by Universal Sentence Encoder.

Please note that loading CSV files from local are not possible due to the security settings of Aura Free instance, so we used online files.

E. RESTful API

One of the key components of our project is the Representational State Transfer (RESTful) Application Programming Interface (API). The source code is available at `src/app.py` This API is used for the client to request the courses and competencies depending on what they are searching for.

There are 4 endpoints for filtering or querying courses or competencies, which are:

- Searching for skills by course id: `/Filtering options/skills/<course_id>`
- Searching for skills by course name: `/Filtering options/skills_name/<course_name>`
- Searching for courses by competency URL (competency id): `/Filtering options/courses/<path:URI>`
- Searching for courses by competency label (competency name): `/Filtering options/courses_label/<label>`

The most reliable ones are the ones that use ids as those have to be unique. On the other hand, the filtering options that use a course name or a competency label (which is also a name), should be entered completely and exact to every character, even omitting any spaces. That way it is possible to match the competencies or courses in the database, otherwise, there won't be any results. Thus, this is one of the possible improvements for the future: to allow searching for courses or competency by only entering part of the name.

Because we have not focused on implementing a Frontend side for the user to interact with the API, we decided to use Swagger UI for a better presentation of the functionality of the 4 available endpoints. In the future, the endpoints can be connected to a Frontend if needed.

To search for the courses or competencies in the database, the RESTful API uses the functions mentioned in Section III-D. Use `python src/app.py` under the project folder to start it.

The next step is to open the browser on localhost and then there are 4 available endpoints as shown in Figure 7 and mentioned above. To search for a course, for instance, the user has to click on `/Filtering options/courses/{URI}` endpoint, click execute, enter the competency URI and then click execute. The results are then represented in a form of a JSON below with the course names that were found and the competency name from the competency URI that was entered by the user in case some courses are found. Otherwise, the JSON will only contain a message that nothing was found.

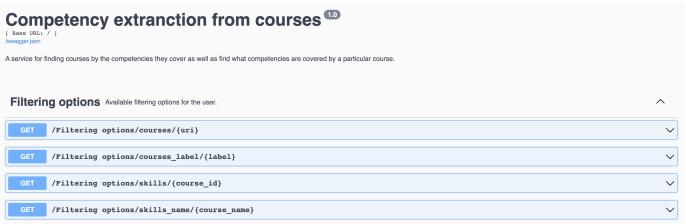

Fig. 7. RESTful API: endpoints

IV. Evaluation

Finally, we have a component for evaluation at `src/Evaluation.ipynb`.

A. Potential Methods

There needs to be a `sample_relations.csv` file, which should store a subset of "ground-truth" pairs of course id with its corresponding competence URIs, similar to the concept of "labels" in machine learning.

`result_files_NN` stores the results of the Universal Sentence Encoder algorithm (course id - competence URI pairs), it may have several files, corresponding to different hyperparameter settings. `result_files_ER` stores the results of the Modified Ontology-based Entity Recognition algorithm (course id - competence URI pairs), it usually only has one file because it has no hyperparameters. For each result in `result_files_NN`, we merge it with the result of `result_files_ER` and find the courses appearing in `sample_relations.csv`, compare the predicted competences, calculate accuracy:

$$\frac{\text{number of correctly predicted competences}}{\text{number of all correct competences}}$$

and precision:

$$\frac{\text{number of correctly predicted competences}}{\text{number of all predicted competences}}$$

.

For the test set, we plan to use the `..._control_re-encoded.xml` provided by the mentor at the end.

For the ”labels”, namely the subset of ”ground-truth” pairs of course id with its corresponding competence URIs mentioned above, we plan to use the results found by a native German speaker in our group.

B. Results

However, in actual evaluation, we found that such evaluation is not very meaningful, because it is difficult for us to rely on humans to find such ”ground-truth” pairs of course id with its corresponding competence URIs, that’s why we want to use the computer to do this job, and the competences dataset itself does not contain the entire competences.

For example, for the last course in the test set: *Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)*, in the beginning, the native German speaker in our group only found the competency *Geometrie*, but our algorithm found more competencies, such as *Bedarf an Baustoffen berechnen*, *Immobilien bewerten*, *den Zustand von Gebäuden untersuchen*, etc., our algorithm would be underestimated if we use the results found by the native German speaker for evaluation.

Another example is that HTML is often mentioned in web technology or programming courses, but this Competence is not included in the Competence data set so our algorithm cannot find it, which will also underestimate the performance of our algorithm, so we finally decided to just give our the results of the first and last course in the test set.

To sum up, we print the results of the first and last course in the test set in Figures: 8, 9, 10, 11:

0	Experte im Prozessmanagement	Arbeitsmarkt
1	Experte im Prozessmanagement	sich selbst darstellen
2	Experte im Prozessmanagement	andere führen

Fig. 8. Modified Ontology-based Entity Recognition Result of First Course

34	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Geometrie

Fig. 9. Modified Ontology-based Entity Recognition Result of Last Course

0	Experte im Prozessmanagement	in einem internationalen Umfeld arbeiten
1	Experte im Prozessmanagement	Verwaltung im Bildungsbereich
2	Experte im Prozessmanagement	Grundlagen der Pädagogik vermitteln
3	Experte im Prozessmanagement	Lehrveranstaltungen an der Universität abhalten
4	Experte im Prozessmanagement	Geschäftsprozesse verbessern
5	Experte im Prozessmanagement	Weiterbildungsbedarf ermitteln
6	Experte im Prozessmanagement	bisher unerkannte Bedürfnisse im Unternehmen e...
7	Experte im Prozessmanagement	betriebliche Weiterbildungsprogramme entwickeln
8	Experte im Prozessmanagement	Personal verwalten
9	Experte im Prozessmanagement	Leistungsdiagnostik
10	Experte im Prozessmanagement	Personal schulen

Fig. 10. Universal Sentence Encoder Result of First Course

V. Conclusion

The main tasks of the project are done, and our algorithms perform better than humans in certain situations, but there is still some space for improvement. The feature extraction could be improved further and we also need to find an approach to handle situations where keywords of the competencies do not directly appear in the course information. Further development could be a better query API and course recommender system based on an individual user profile. Finally, in the `README` file, or on our Github homepage: , you can find detailed information about installation and configuration of the project.

198	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Gelände für die Rohrleitungsverlegung begutachten
199	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Anlagenstandorte kontrollieren
200	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Werkstoffe zum Bau von Geräten berechnen
201	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Baustoffkosten veranschlagen
202	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Baustelle vorbereiten
203	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Hausbau planen
204	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Standort auf Bautätigkeit vorbereiten
205	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Bauweisen
206	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Immobilien bewerten
207	Fernlehrgang Fortbildung Aufmaßtechniker*in (ZAB)	Datenqualitätskriterien festlegen

Fig. 11. Universal Sentence Encoder Result of Last Course

References

[1] L. Price and A. Kirkwood, “Technology enhanced learning–where’s the evidence,” *Curriculum, technology & transformation for an unknown future. Proceedings ascilite Sydney*, pp. 772–782, 2010.

[2] “Why is ESCO needed?” https://esco.ec.europa.eu/de/node/107 (accessed Jul. 18, 2022).

[3] “PAS 1045,” https://de-academic.com/dic.nsf/dewiki/1067131 (accessed Jul. 18, 2022).

[4] J. Kaur and V. Gupta, “Effective approaches for extraction of keywords,” *International Journal of Computer Science Issues (IJCSI)*, vol. 7, no. 6, p. 144, 2010.

[5] B. Sateli, F. Löffler, B. König-Ries, and R. Witte, “Scholarlens: extracting competences from research publications for the automatic generation of semantic user profiles,” *PeerJ Computer Science*, vol. 3, p. e121, 2017.

[6] T. Hoppe, J. Al Qundus, and S. Peikert, “Ontology-based entity recognition and annotation,” in *Proceedings of the Conference on Digital Curation Technologies (Qurator 2020)*, F. FOKUS, Ed., vol. Vol 2535. Berlin: CEUR Workshop Proceedings, 2020, pp. 1–11, paper. [Online]. Available: http://ceur-ws.org/Vol-2535/paper_4.pdf

[7] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, “Universal sentence encoder,” *arXiv preprint arXiv:1803.11175*, 2018.

[8] “European Skills/Competences, Qualifications and Occupations (ESCO),” https://ec.europa.eu/social/main.jsp?catId=1326&langId=en (accessed Jul. 20, 2022).

[9] “Moses - Machine Learning 2,” https://moseskonto.tu-berlin.de/moses (accessed Jul. 31, 2022).

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[11] S. Paul, A. Mitra, and C. Koner, “A review on graph database and its representation,” in *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC)*. IEEE, 2019, pp. 1–5.

[12] “Diese Vorteile bieten Graphdatenbanken,” https://www.bigdata-insider.de/diese-vorteile-bieten-graphdatenbanken-a-615118 (accessed Jul. 18, 2022).

[13] “Python - Neo4J Aura,” https://neo4j.com/docs/aura/auradb/connecting-applications/python/ (accessed Jul. 31, 2022).

[14] “Getting Started with Cypher,” https://neo4j.com/developer/cypher/intro-cypher/ (accessed Jul. 31, 2022).