MJI
Media Jurnal Informatika
Vol. 17, No. 2, December 2025
p-issn : 2088-2114, e-issn : 2477-2542

Application of Named Entity Recognition (NER) in Job Vacancy Matching Using an Ontology-Based Approach (Case Study: Information Technology Sector)

Rizki Gunawan [a,1], Ade Hodijah [a,2,*], Muhammad Ikhsan Maulana Taqwim [a,3], Afyar Siti Ababil [a,4], Urip Teguh Setijohatmo [a,5], Sri Ratna Wulan [a,6], Muhammad Riza Alifi [a,7], Aprianti Nanda Sari [a,8], Hashri Hayati [a,9]

[a] Politeknik Negeri Bandung, Jl. Gegerkalong Hilir, Bandung 40559, Indonesia
[1] rizki.gunawan.tif22@polban.ac.id; [2] ade.hodijah@polban.ac.id*; [3] muhammad.ikhsan.tif22@polban.ac.id; [4] afyar.siti.tif22@polban.ac.id; [5] urip@jtk.polban.ac.id; [6] sri.ratna@polban.ac.id; [7] muhammad.riza@jtk.polban.ac.id; [8] aprianti.nanda@jtk.polban.ac.id; [9] hashri.hayati@jtk.polban.ac.id
* corresponding author

ARTICLE INFO

Article history
Received 2025/06/12
Revised 2025/12/26
Accepted 2025/12/27

Keywords
Job vacancy application
Named Entity Recognition
Taxonomy
Ontology

ABSTRACT

The dissemination of job vacancies through online platforms still faces limitations in understanding the semantic relationships between the skills possessed by job seekers and the qualifications required by a job position. This mismatch results in an inefficient search process and longer search times. This study aims to develop a semantic-based job vacancy recommendation application (talent matching) using a skill ontology approach. One of the main challenges in developing the ontology is the lack of standardized data structures in job vacancy postings, particularly in the job description section. To address this issue, Named Entity Recognition (NER) techniques are applied to automatically extract skill entities from job description texts. The extracted results are then classified into a taxonomy structure using SkillsGPT, thereby forming a hierarchical skill concept model semantically represented within the ontology using Protégé. The matching process between user skills and job qualifications is conducted through semantic similarity calculations employing the Sánchez Similarity method. Job vacancy data are collected via web scraping, while system development follows the Rational Unified Process (RUP) methodology and is evaluated using Black Box testing. Evaluation results demonstrate that the developed system is capable of providing semantically relevant job vacancy recommendations according to the user's skill profile. Therefore, this study contributes both theoretically and practically to the development of ontology-based recommendation systems, particularly in the automated modeling of skill taxonomies from unstructured data.

1. Introduction

Job search systems on many platforms still rely on keyword matching (string matching), which is unable to capture the semantic relationships (proximity) between job seekers' skills and the qualifications required in job vacancies [1]. Platforms such as Glints [2] and Kalibrr [3] are among the online platforms used by job seekers to search for job openings by reading each vacancy individually. This approach slows down the process of finding jobs that match their skills.

To address the challenges of limited keyword matching and the system’s inability to comprehend semantic relationships between skills, the implementation of an ontology model enables the system to understand the semantic relations among skills in job vacancies and job seeker profiles, thereby producing more relevant recommendations [4][5][6][7].

Ontology is a computational model used to represent objects within a domain and their interrelationships, consisting of classes, individuals, data properties, and object properties, which together form an explicit representation of a knowledge domain [8]. Due to its high representational capacity, ontology models that are both human- and machine-readable have been employed in various applications to formally model fundamental concepts [8]. Ontologies provide a hierarchical structure that enables the systematic organization of knowledge, ranging from the most general to the most specific concepts [8].

In the development of the ontology model [9] skill classification (taxonomy) is required to calculate concept similarity based on the ontology structure using the Sanchez Similarity method [10]. Skill information was obtained from scraping data [10], [11] of job vacancies from job listing platforms, namely Glints [2] and Kalibrr [3].

Web scraping is a technique used to obtain data from websites that do not provide Application Programming Interface (API) access [11]. This technique is employed in data retrieval, market analysis, as well as social and economic research by accessing HTML elements to extract relevant information quickly and efficiently [11].

There is a difference in the data structures of the two platforms. Data from Glints can be used directly because the platform already separates information such as skills in a structured format. In contrast, data from Kalibrr requires further processing, as information regarding skills and experience is still mixed within the job description section. Therefore, Named Entity Recognition (NER) techniques are necessary to extract relevant skill and experience information from these descriptions so that skills can be used in the matching process and experience can serve as additional information [12].

Named Entity Recognition (NER) is a technique within the field of Natural Language Processing (NLP) that focuses on extracting information from both structured and unstructured documents [12]. The primary objective of NER is to identify and classify named entities present in a text, such as person names, organizations, locations, times, dates, and other entities relevant to the context of the developed application [12]. NER not only labels specific words or phrases as entities but also attempts to understand the context in which these entities appear [12].

This research aims to develop a job vacancy recommendation application that does not rely on keyword matching based on skills or job positions. Instead, it utilizes the proximity between the job seeker's skills and the skill qualifications described in the job vacancy using an ontology-based approach calculated by the Sanchez Similarity formula. In this study, the job vacancy search process based on skill proximity is referred to as semantic matching. Broadly speaking, there are three main conditions that trigger the semantic matching process in the job vacancy search (talent matching) application developed in this research. The first condition occurs after the system obtains the latest job vacancy data through a scraping process using Named Entity Recognition (NER). The second condition is triggered when the job seeker updates their skill list. The third condition takes place when a new job seeker registers in the application.

2. Method

Fig.1 illustrates the overall research methodology applied in this study. The process begins with data collection through web scraping from job vacancy platforms, followed by data pre-processing to ensure data quality. Named Entity Recognition (NER) is then applied to extract hard skills and work experience from unstructured job descriptions. The extracted skills are organized into a taxonomy using SkillsGPT and modeled as an ontology using Protégé. Semantic matching between job seeker skills and job requirements is performed using the Sánchez similarity method. Finally, the system is implemented using Neo4j with Neosemantics and evaluated through NER performance metrics and black-box testing.

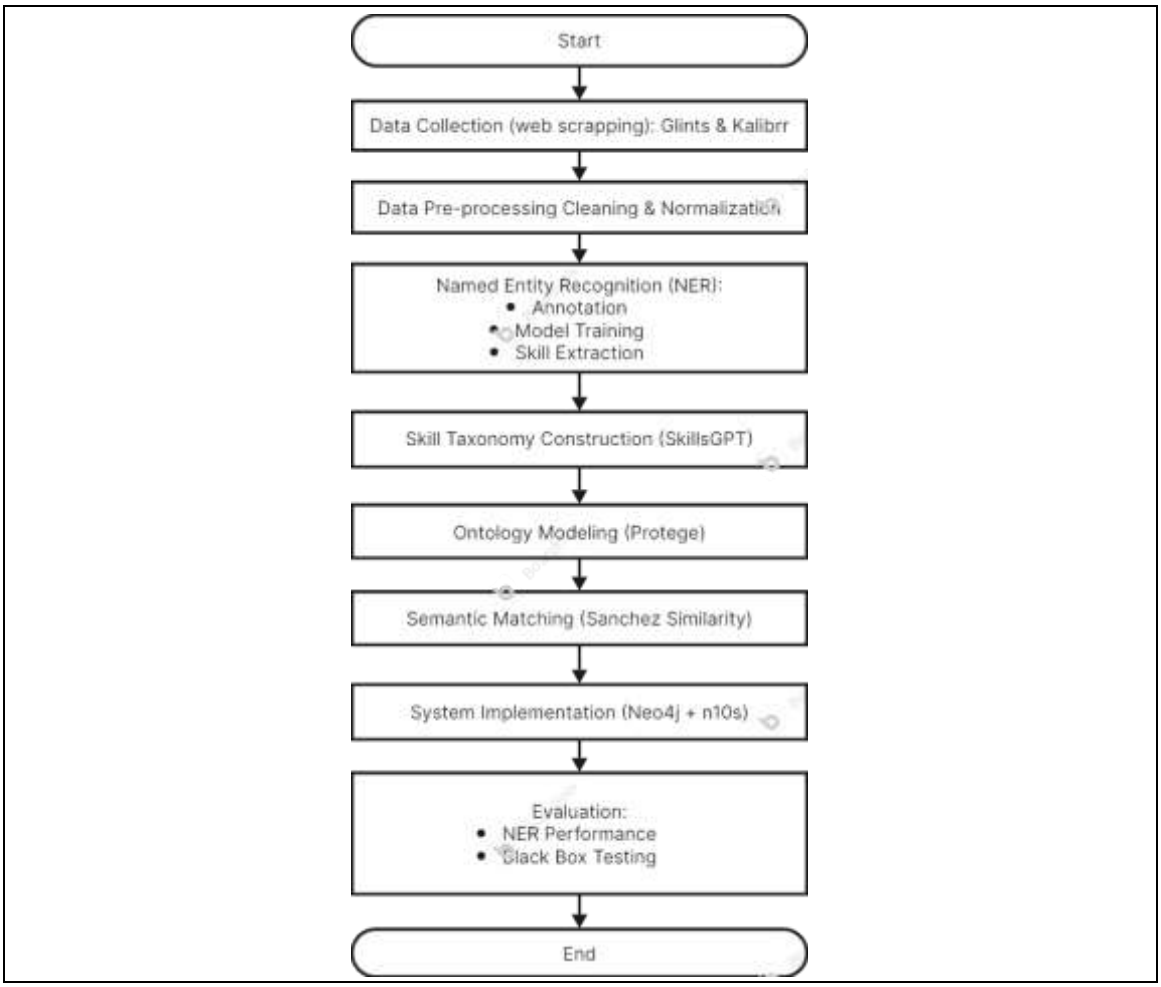

Fig. 1. Research methodology flowchart of the ontology-based job vacancy matching system

This study applies a case study approach on a talent matching application developed using Object Oriented Programming (OOP) methodology and following the Software Development Life Cycle (SDLC) model from the Rational Unified Process (RUP) [13]. The talent matching application provides job vacancy recommendations based on information technology (IT) job openings that best match the IT skills in the job seeker's profile [1]. Job vacancy data were collected through a web scraping process, gathering approximately 1,000 job listings from Glints [2] and Kalibrr [3]. Table 1 presents the attributes of the web-scraped job vacancy data.

Job vacancy data related to skills embedded within unstructured job description details were extracted using Named Entity Recognition (NER), involving a training process evaluated using a combination of precision, recall, and F1-score metrics [12], [14], [15]. The scraped IT-related skills were then organized into a taxonomy with the support of SkillsGPT [16], and subsequently modeled as an ontology using the Protégé tool [17]. This taxonomy serves as the foundation for semantic matching between job seekers' IT skills and those required in job vacancy descriptions [9], [18]. The matching process was conducted using the Sanchez Similarity method [10], which calculates the degree of conceptual similarity within the ontology.

Table 1. Data scraping attributes before of data pre-processing.

Web Scraping	Before Pre-processing	
	Attribute	**Description**
1	job_url	The link is provided to redirect users to the original source page of the job vacancy.
2	image_url	The link to access the company image files sourced from the relevant job vacancy platforms.

Web Scraping	Before Pre-processing	
	Attribute	**Description**
3	job_title	Job vacancy title.
4	company_name	The name of the company offering the job vacancy.
5	subdistrict	The name of the sub-district/regency where the company is located..
6	city	The city in which the company is located.
7	province	The name of the province where the company is located.
8	salary	The salary offered for the corresponding job vacancy.
9	employment_type	The types of employment, such as full-time, part-time, or contractual work.
10	work_setup	The workplace options for employees may include working from home, at the office, and other possible locations.
11	minimum_education	The minimum educational requirement specified in the job vacancy.
12	minimum_experience	The minimum experience required as stated in the job vacancy.
13	required_skills	The job vacancy skills in the field of Information Technology.
14	job_description	The job vacancy descriptions, including job requirements.
15	scraped_at	The timestamp when the data was collected during scraping.

There are four phases in the Rational Unified Process (RUP) [13], namely: inception, elaboration, construction, and transition. This research paper focuses specifically on how the Named Entity Recognition (NER) process is conducted at each phase of the RUP for a job vacancy search application using an ontology-based approach. Based on the business process analysis of job matching, the initial phase (inception) involves data collection for NER, where the volume of data is dynamic and depends on the number of available job listings at the time of web scraping. Once the job vacancy data is successfully retrieved, the subsequent phases include training the NER model (elaboration), designing the skill taxonomy (construction), and testing the NER results (transition).

3. Results and Discussion

This study presents the results focusing on the utilization of Named Entity Recognition (NER) processes to construct a skill taxonomy from job vacancy descriptions obtained through web scraping (Glints and Kalibrr), which is subsequently modeled into an ontology.

3.1 Inception

Following the scraping process, the raw data from both platforms could not be utilized directly, as several attributes remained unclean and inadequately separated. Prior to the pre-processing stage, the scraped data initially appeared as shown in Table 1. Therefore, a series of pre-processing steps was necessary to ensure that the data used in the system was sufficiently clean. In general, the steps conducted after scraping included the following:

1. Data Pre-processing

   Each platform provides data with differing structures and attribute completeness. At this stage, data cleaning was performed, including temporarily removing HTML tags from job descriptions (job_description) prior to entity extraction, as well as normalizing the salary and experience fields by converting string values into corresponding minimum and maximum numeric values. All attributes were standardized to ensure consistency between Glints and Kalibrr, such as splitting the location field into separate city and province attributes, harmonizing values of employment_type and work_setup, and aligning the handling of missing or empty attributes.

2. Entity Extraction Using Named Entity Recognition (NER)

   To extract critical information that is not always explicitly available in dedicated attributes—namely hardskills and work experience, a Named Entity Recognition (NER) technique was employed. The NER model was specifically designed to detect two primary entities: HARDSKILL, extracted from the skill attribute and job description, and EXPERIENCE, extracted from the job description..

3. Normalization of Skill Keywords

The extracted skills, obtained from both specific attributes and Named Entity Recognition (NER) results, were normalized using a manually constructed hard skill dictionary based on the scraped data. This normalization process aims to standardize various skill name variations into a unified keyword to ensure consistent matching within the system.

4. Data Integration

   Data obtained from both sources were subsequently integrated into a unified, standardized format, as illustrated in Table 2.

The outcomes of the entire pre-processing stage resulted in structural modifications to the dataset, as illustrated by the comparison between Table 1 and Table 2. The main changes include the separation of the salary attribute into two distinct attributes, namely minimum_salary and maximum_salary, to provide a clearer representation of salary ranges, and the addition of the maximum_experience attribute to complement the previously available minimum_experience information. Table 1 presents the job vacancy data obtained through web scraping prior to the pre-processing stage.

Table 2. Data scraping attributes after data pre-processing

Web Scraping	After Pre-processing	
	Attribute	**Description**
1	minimum_salary	The minimum salary offered in the relevant job postings.
2	maximum_salary	The maximum salary offered by the relevant job vacancy.
3	maximum_experience	The maximum required experience specified in the job vacancy..

The scraping results from the Glints [2] and Kalibrr [3] platforms reveal fundamental differences in data structure and completeness. Glints provides a relatively clean and structured website layout, where key attributes such as required skills, salary information, and work experience are separated into distinct fields. This facilitates the data extraction process, as the values can be retrieved directly from well-defined fields. In contrast, Kalibrr presents a more complex and inconsistent page structure. Critical information such as skills and work experience remains embedded within the job description block without clear semantic separation. This poses challenges in the data extraction process, requiring further analysis to identify and classify the information—one possible approach being the application of Named Entity Recognition (NER) techniques.

In addition to differences in data structure, the security aspects of both websites are also of concern. Both platforms implement protection mechanisms against suspicious activities, such as rate limiting, Cloudflare protection, and CAPTCHA. Therefore, web scraping cannot be performed arbitrarily. The scraping system was designed to consider anti-bot detection measures, including the use of undetected headless browsers, user-agent rotation, and request delay injection. In the case of Glints, scraping cannot proceed beyond the first page without prior authentication. In contrast, Kalibrr technically allows access without login; however, its guest pages tend to load slowly and heavily, resulting in unstable scraping performance.

The scraping process was not performed manually but was instead automated as a background process within the system. As illustrated in Fig. 2, the scraping workflow follows the same stages for both the Glints and Kalibrr platforms. The process begins with authentication to the respective websites, followed by accessing the IT job listings page, extracting all job listing links from each page while iterating through subsequent pages if available, then accessing the detail page for each collected job link, retrieving detailed job vacancy data, and performing data cleaning. Although the overall stages are generally similar for both Glints and Kalibrr, there are differences in the technical implementation of the data cleaning phase.

After completing the scraping process from the Glints and Kalibrr platforms, the collected data from both sources were stored in a Redis database for subsequent processing. The system was designed to handle large-scale scraping operations without requiring manual intervention by the administrator in retrieving job vacancy data. However, the scraped data were not always clean or ready for use. Several data quality issues were identified, such as skill fields containing irrelevant soft skills, inconsistent skill entries due to typographical errors, and relevant skills not listed explicitly in the skill field but embedded within the job description. These challenges highlight the critical need for further processing steps, particularly pre-processing and Named Entity Recognition (NER).

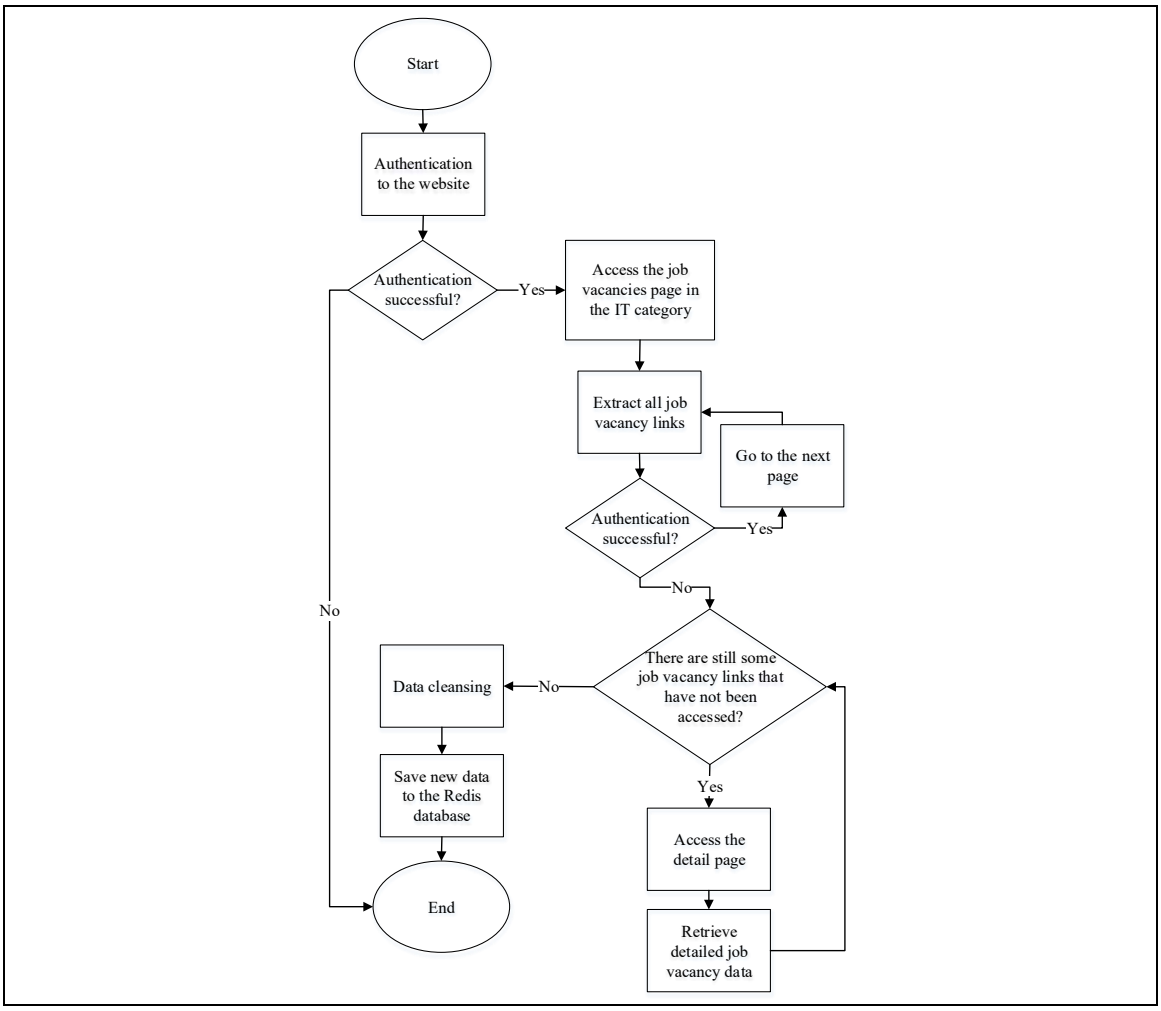

Fig. 2. Flowchart of scraping process

The initial phase (first condition) begins by inserting job vacancy data obtained through web scraping into the ontology, followed by the integration of pre-existing user data into the same ontology. Subsequently, the system performs a bulk matching process, in which all user data are compared against all job vacancies available in the system. The resulting match scores are then imported into Neo4j using Neosemantics [19], [20].

In the "Importing Data into Neo4j" stage, as illustrated in the flowchart, the system applies a full data replacement mechanism. This process begins by removing all existing job vacancy data from the Neo4j database, followed by inserting the newly scraped job postings along with the computed matching results. This replacement approach is adopted to ensure that the database consistently contains the most up-to-date job vacancy information and to prevent data duplication. Consequently, upon the completion of each scraping and matching cycle, the database is refreshed with the latest data, accurately reflecting the current job listings available on the Glints and Kalibrr platforms.

"Due to the large-scale nature of the matching process in this condition (involving all user and job vacancy data), the system enters maintenance mode during execution. In this mode, user access to the application is highly restricted; regular users are only permitted to access the exploration page to view a general list of job vacancies, while the recommendation page and other personalized features are temporarily disabled until the matching process is completed. Full access to the application during maintenance is granted exclusively to administrators through a dedicated admin interface. This strategy is implemented to ensure data integrity and maintain system performance throughout the mass matching process, as illustrated in Fig. 3.

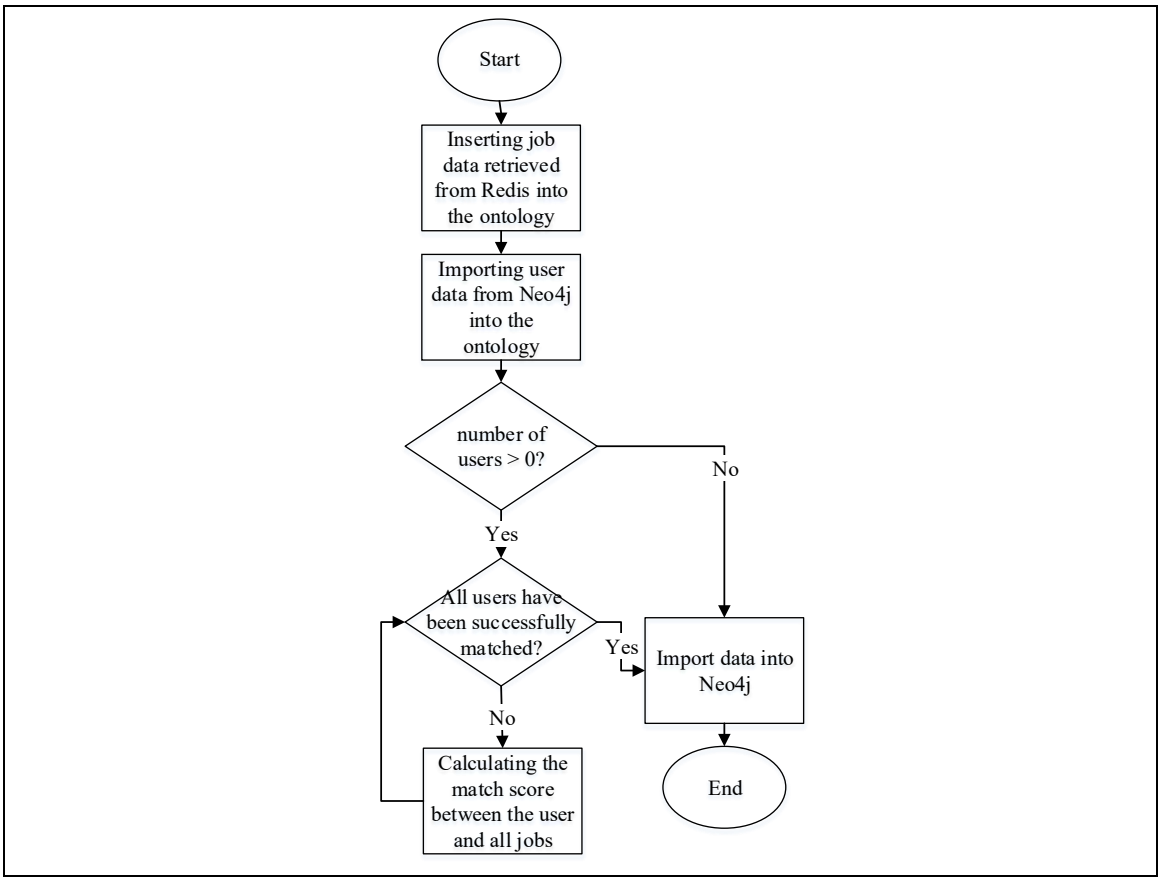

Fig. 3. Flowchart of the matching process after the scraping process

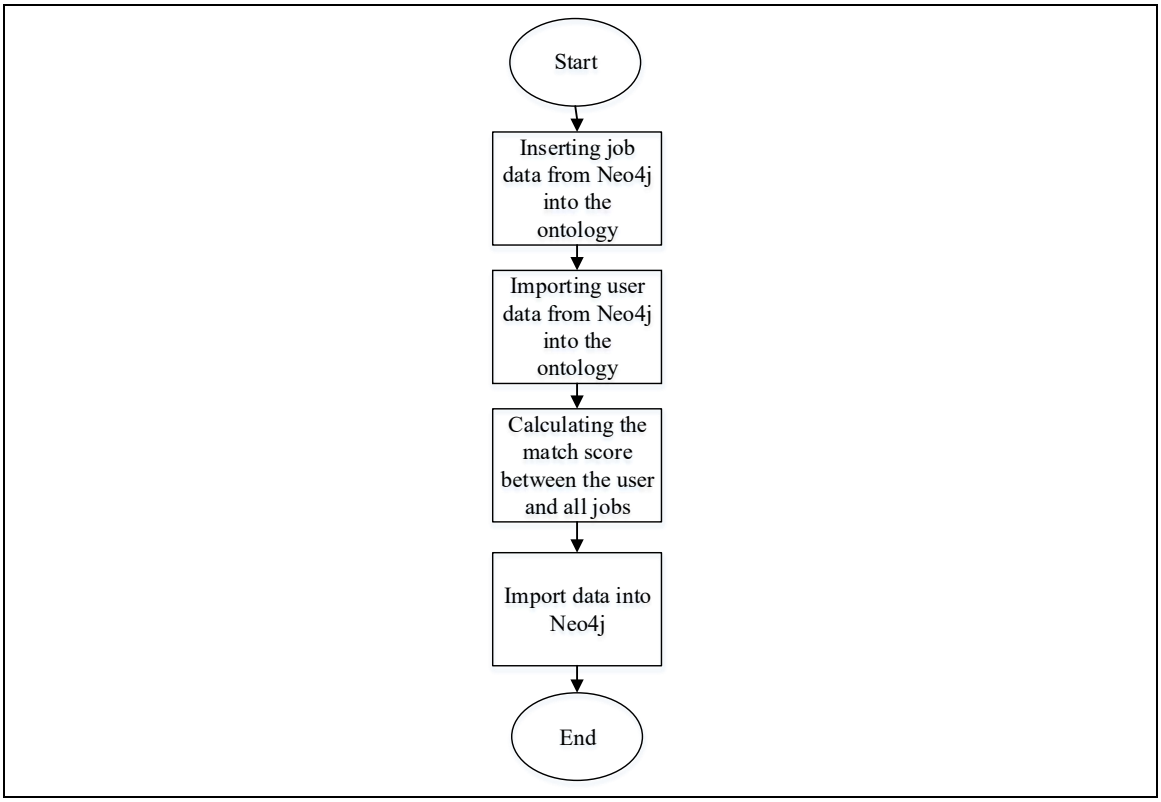

Fig. 4. Flowchart of the skill matching process triggered by user skill updates

The second possible condition is triggered when a user updates their skill list through the profile page. The process flow for this condition is illustrated in Fig. 4. In this scenario, the system retrieves the updated user data along with all available job vacancies and inputs them into the ontology for re-computation of similarity scores. The matching results between the user and all job vacancies are then imported into Neo4j to update the recommendation data.

Unlike the first condition, the scope of the matching process in this scenario is limited to a single user against all available job postings, making the process computationally lighter. However, the matching is still performed synchronously, meaning the user must wait until the process is completed before accessing certain application features, such as the recommendation page. This approach ensures that the presented recommendations are accurately aligned with the user's most recent skill updates.

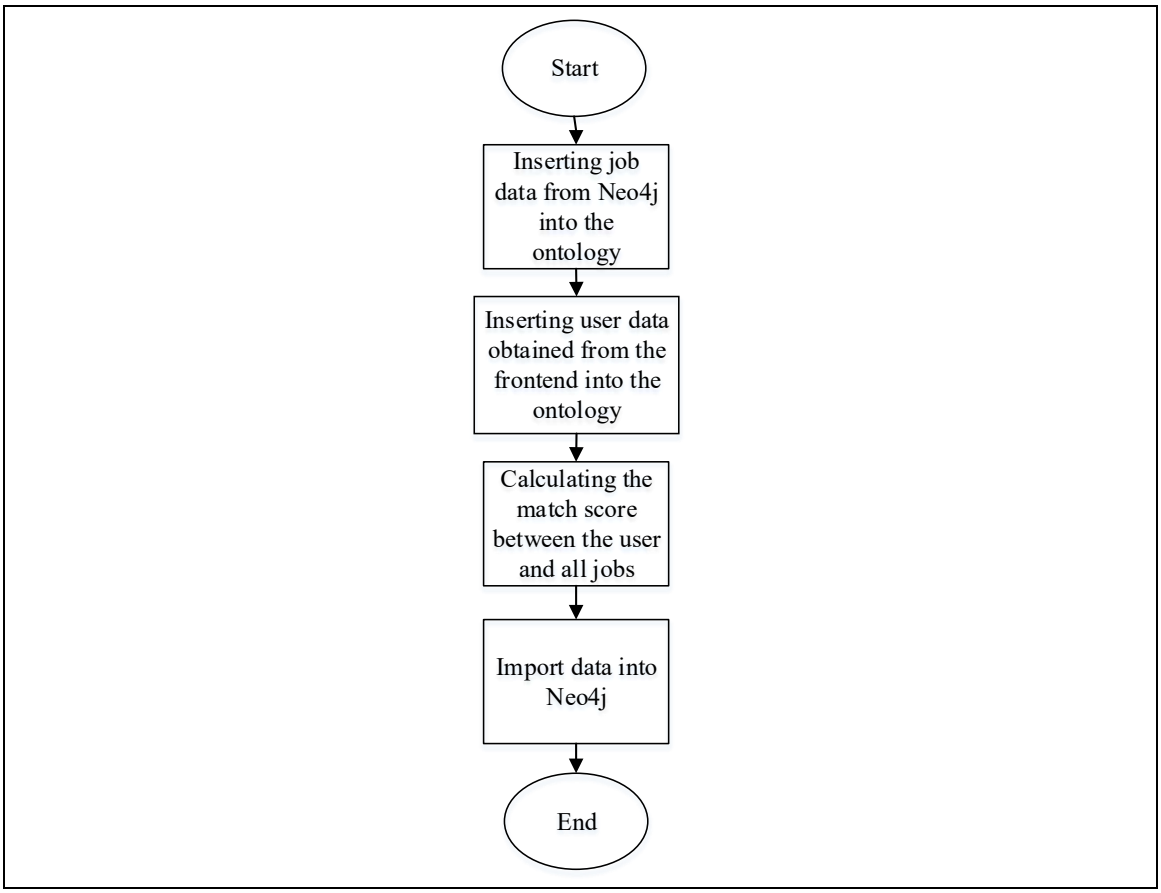

Fig. 5. Flowchart of the matching process during user registration

The final scenario is likely to occur when a new user registers in the system. The process flow for this condition is illustrated in Fig. 5. During registration, the newly inputted user data, along with all available job vacancy data, are integrated into the ontology. The system then calculates the similarity scores between the new user and all existing job vacancies. As in the second scenario, the matching process during registration is also executed synchronously. The user is required to wait until the matching process is completed (approximately 5–10 minutes) before accessing the recommendation page. This ensures that the system can immediately provide relevant job recommendations based on the user's profile and submitted skills. Once the matching process is completed, the results are imported into Neo4j, and the user can access the recommendation feature with the updated data.

3.2 Elaboration

To address issues arising from the unstructured nature of scraped data, a Named Entity Recognition (NER) approach was employed to identify and classify disorganized information. The NER model was utilized to detect HARDSKILL entities from both the skill attributes and job descriptions, as well as EXPERIENCE entities from job descriptions. However, pre-trained NER models, such as those provided by SpaCy, are not equipped to recognize domain-specific labels such as hardskill and experience. Therefore, a custom NER model was trained using 100 job vacancies postings selected from the initial scraping results. The training

process began with data annotation (tagging), which involved manually reviewing each job description text and marking specific segments as HARDSKILL and EXPERIENCE entities.

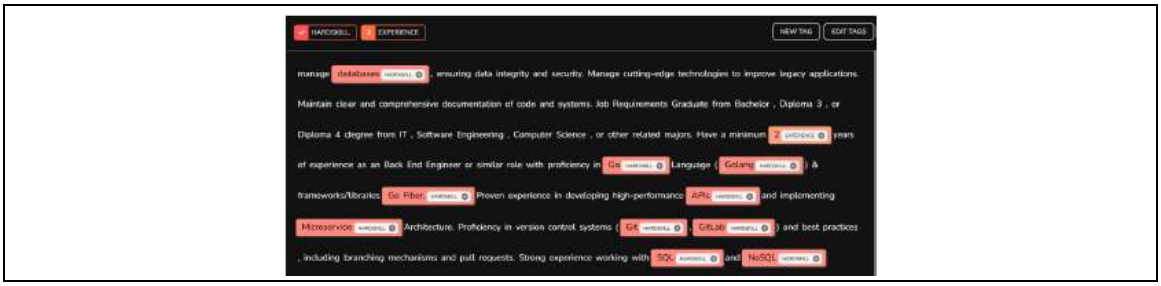

Fig. 6. UI of annotator application

The tagging process was conducted based on predefined criteria and standards to ensure annotation consistency. For the HARDSKILL entity, the criteria include measurable and specifically learnable technical capabilities that are directly related to information technology, such as programming languages, frameworks, databases, development tools, or technology platforms. This entity must represent a technical skill rather than a soft skill such as communication, leadership, or teamwork, and it should be verifiable through certifications, portfolios, or technical assessments. Examples of tagged hardskills include 'Java', 'React.js', 'MySQL', 'Docker', 'Git', 'AWS', and similar technologies.

For the EXPERIENCE entity, the tagging process focuses on identifying numerical values that indicate the required duration of work experience specified in the job vacancy descriptions. The tagging criteria include recognizing numbers that represent the length of experience, whether in singular form such as “2,” “3,” or “5,” or in range form such as “1–3,” “2–5,” or “3–7.” The tagged EXPERIENCE entity covers explicitly stated durations of work experience mentioned in the context of job requirements, for instance, in phrases such as “minimum 3 years of experience,” “2–4 years of experience in web development,” or “at least 5 years of professional experience.

The tagging criteria help distinguish between experience requirements and other general job descriptions, enabling the NER model to accurately identify critical information needed for the job vacancy recommendation process. The tagging process presents a significant challenge due to its demand for high precision and considerable time investment. To facilitate this process, an open-source tool, the NER Annotator UI (https://arunmozhi.in/ner-annotator/) [21], was utilized. This tool provides a user-friendly interface that enables manual annotation of named entities. Fig. 6 illustrates the application's interface as used during the tagging of job description data.

The result of this annotation process is a JSON file containing the text structure, a list of entity labels used, and the token positions within the text that have been marked as specific entities. The JSON file structure consists of three main components: text, representing the annotated text content; classes, containing the list of labels, namely HARDSKILL and EXPERIENCE; and entities, indicating the character ranges for each entity in the text. An example of the JSON output from the tagging process is shown in Fig. 7.

After the annotation process is completed, the resulting JSON file is utilized as training data for developing the NER model using the SpaCy framework. Fig. 9 presents the flowchart illustrating the stages from annotation to model development, beginning with text collection, annotation preparation, manual annotation using an NER tagging application, storing the annotated data in JSON format, and concluding with model training using the SpaCy pipeline.

Before training the NER model, it is necessary to prepare an appropriate configuration file to facilitate the training process. The basic configuration file can be obtained from the official SpaCy website (https://spacy.io/usage/training) [22]], which provides the base_config.cfg template as a starting point for training configuration. Fig. 8 illustrates the selection of initial configurations that can be chosen according to the specific requirements. This configuration file is subsequently used to conduct training with SpaCy Transformers tailored to the application needs.

The NER model was trained using SpaCy Transformers with customized configurations based on SpaCy’s default base_config.cfg. Key modifications included the use of the bert-base-multilingual-uncased model as the transformer backbone to support multilingual context understanding. The language setting was changed from "en" to "xx" (multilingual) to accommodate mixed Indonesian-English text commonly found in job vacancy descriptions. Additionally, the batch size was adjusted from 1000 to 128 to optimize memory usage and prevent overflow during training.

Fig. 7. The JSON file resulting from the tagging process

The configuration also enables PyTorch as the gpu_allocator to leverage GPU computing for accelerating the training process. The pipeline is extended with the components 'transformer' and 'ner' to support transformer-based entity extraction. The optimizer employs Adam with a warmup_linear learning rate schedule to ensure training stability, while the batching strategy utilizes batch_py_padded to improve training efficiency by grouping data with similar text lengths.

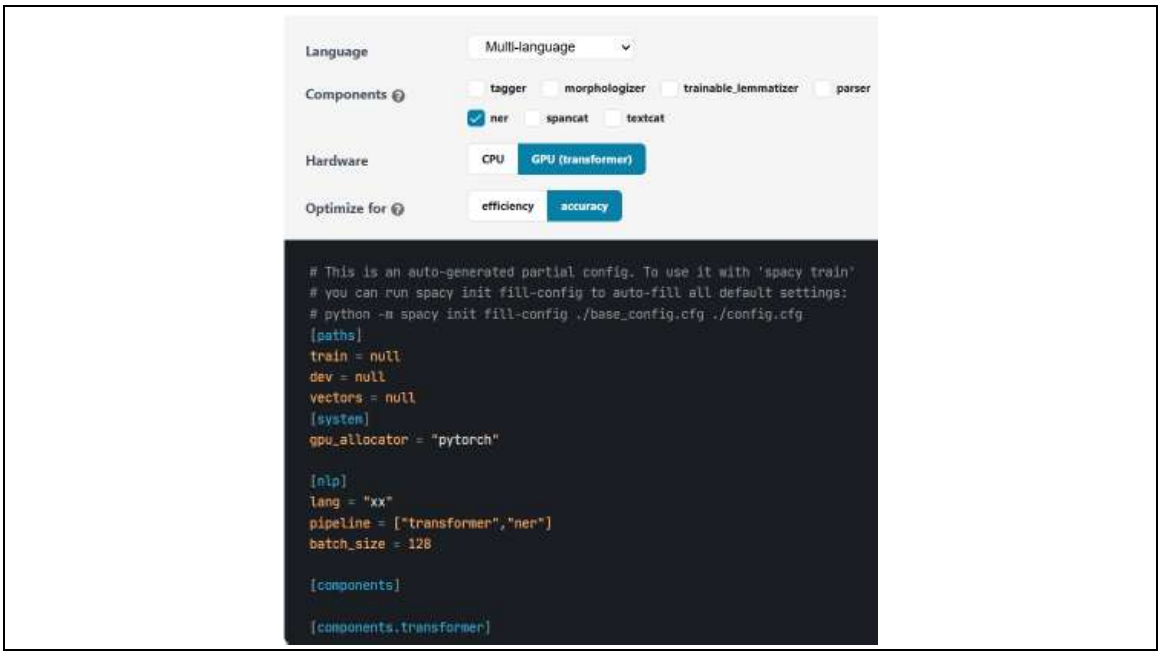

Fig. 8. Interface for generating the base_config.cfg template

The training process was conducted by monitoring the SCORE value, a composite evaluation metric derived from precision, recall, and F1-score. As shown in Fig. 10, the training was manually stopped once the SCORE exceeded 0.95, indicating that the model had achieved satisfactory performance for entity extraction of the Training label. The training results are presented in Fig. 7. The trained model is then ready to be integrated into the system to automatically perform entity extraction on new job vacancy data.

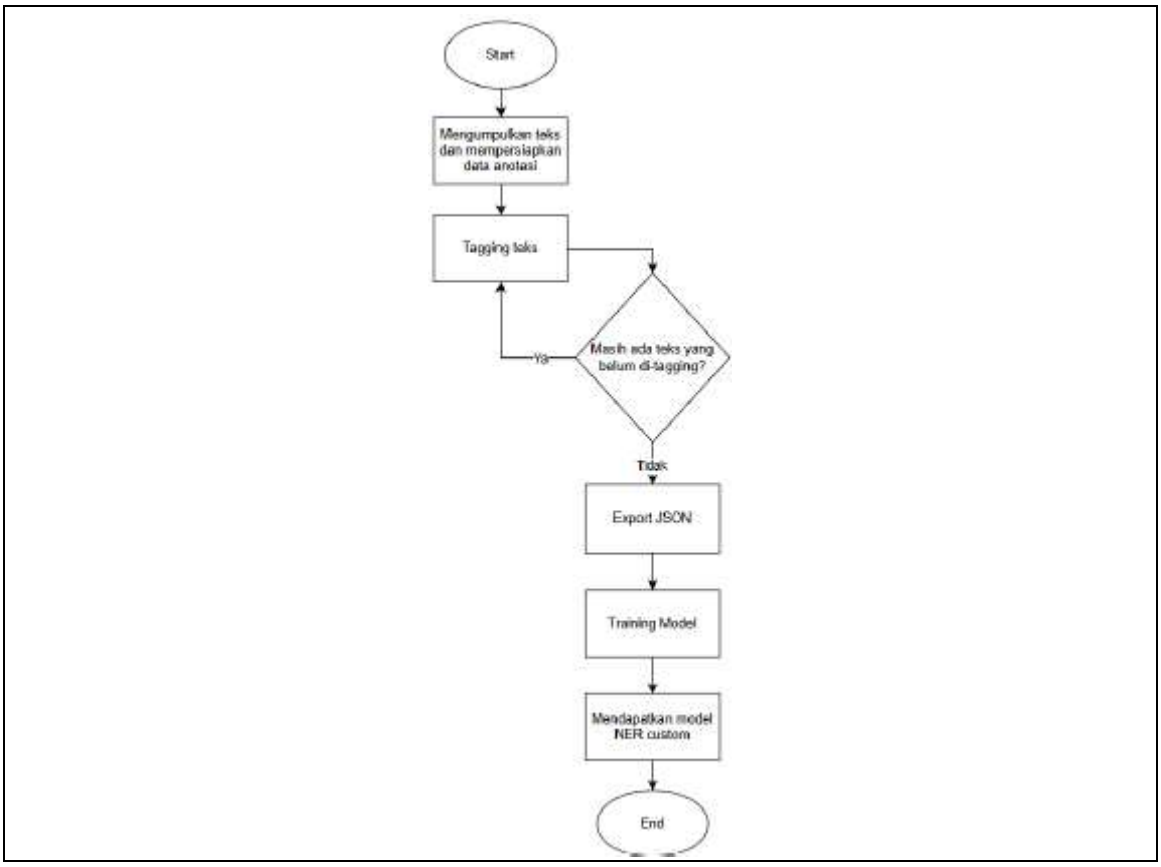

Fig. 9. Flowchart of the NER model training process

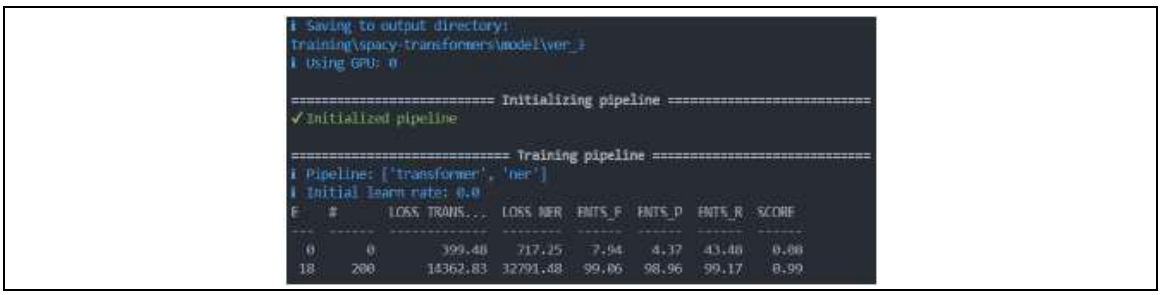

Fig. 10. The results of the NER model training

After the NER model is trained and integrated into the data scraping pipeline, a series of post-processing steps are applied to ensure the quality and consistency of the extracted entity data. Table 2 outlines the stages of the NER technique and its corresponding post-processing procedures.

1. Detection and filtering of hard skills from the job vacancy skills attribute

   This process employs a Named Entity Recognition (NER) model to analyze the required_skills attributed to in job vacancy data in order to distinguish between hard skills and soft skills. The NER model identifies each skill within the attribute and classifies it as either a HARDSKILL or not. Skills that are not recognized as hard skills—such as soft skills like 'communication', 'teamwork', or 'leadership'—are removed from the required_skills attribute, ensuring that only relevant hard skills remain for the subsequent matching process..

2. Detection of hard skills from job descriptions

   This process utilizes a Named Entity Recognition (NER) model to identify hardskills mentioned within the job description text that are not explicitly listed in the required_skills attribute. For example, in the Glints dataset for the position of Kubernetes Engineer, the NER model successfully extracted the hardskills "Grafana" and "Prometheus" from the sentence "Experience with monitoring

tools especially Grafana and Prometheus" found in the job_description. These skills were subsequently added to the required_skills attribute, which originally contained only [Rancher, Minio, K3s, Kubernetes, Rke2, Longhorn]. In contrast, for the Kalibrr dataset with the Oracle Functional Analyst position, where the initial required_skills attribute was empty, the NER model extracted all hardskills from the job description. The model identified "SQL" from the sentence "Strong knowledge and hands-on experience with complex SQL queries", "PL/SQL" and "API" from "Knowledge in EBS DBMS structure, PL/SQL, API", and "AWS" from "Knowledge in AWS concepts is an advantage".

3. Skill keyword normalization using a dictionary

   After the hardskills are detected, the system performs skill name normalization using a predefined dictionary. Based on the results from Glints, the skill variant 'Rke2' is standardized to the canonical keyword 'RKE2'. Meanwhile, from the Kalibrr dataset, the detected skill 'API' is normalized to the standardized keyword 'API Development'. This process ensures consistency in skill naming, which is crucial for the subsequent matching algorithm.

4. Extraction of work experience from job descriptions

   The NER model identifies mentions of work experience within job description texts from Kalibrr and converts them into numerical values. For instance, from the sentence “A minimum of 1–3 years of experience in an Oracle EBS R12” found in Kalibrr data, the system extracts the range “1–3” and stores it as minimum_experience: 1 and maximum_experience: 3. This extracted information replaces or supplements previously missing experience attributes. Based on the results, the integration of the NER model with post-processing stages successfully enriched the job vacancy data. In the Glints dataset, the number of skills was expanded from six to eight, with more specific and standardized entries. Meanwhile, the Kalibrr dataset, which initially lacked skill and experience information, was enhanced with five skills and an experience range of 1–3 years. The resulting enriched dataset is then ready to be utilized for the matching process within the application.

3.3 Construction

In this stage, a skill ontology was developed from job vacancy data obtained through web scraping, utilizing the SkillsGPT tool [9], This ontology was intended for use in skill matching processes. The analysis began by identifying the skills present in the scraped data. Job vacancy data in JSON format were parsed specifically at the “required_skill” field, which had been pre-categorized as hard skills. These skills were then filtered to retain only unique entries and subsequently saved in a .txt format. The final outcome of this process was a semantically-based skill ontology suitable for implementation in the Sanchez Similarity algorithm. Fig. 11 presents a flowchart that illustrates the complete process, from JSON extraction to the construction of the ontology.

In this process, several existing skill ontologies and taxonomies, such as ESCO, CSO, and Roadmap.sh, were analyzed. However, discrepancies were identified between the skills extracted from job vacancy scraping and those listed in these ontologies or taxonomies. For instance, ESCO and CSO do not include ReactJS, a popular programming skill frequently appearing in the scraped job data. This is primarily because ESCO covers all employment sectors, not exclusively the IT domain, while CSO focuses more on theoretical knowledge in computer science. Consequently, neither ESCO nor CSO includes programming frameworks in their taxonomies, but rather only the programming languages and their associated knowledge domains. Meanwhile, Roadmap.sh [23] contains a relatively limited number of skills, resulting in certain scraped skills not being represented within it. Therefore, to support a talent-matching application based on job scraping data, it is more appropriate to develop a custom ontology to ensure the skill taxonomy aligns with the actual skills appearing in job postings.

The ontology development was supported by SkillsGPT, as the process of identifying reference skills for ontology construction is highly time-consuming. Each skill extracted from job postings must be individually analyzed to determine its semantic relationships. SkillsGPT accelerates this process by leveraging its training on 30,000 LinkedIn skills, enabling faster identification and mapping of semantic relationships between skills.

The ontology was constructed using skills extracted from an initial set of 1,000 job vacancy data obtained through web scraping. This approach was adopted due to the lack of reliable automated solutions for comprehensive skill ontology construction. By developing the ontology manually, its structure could be tailored to align with the specific data requirements and methodological approach employed in the application.

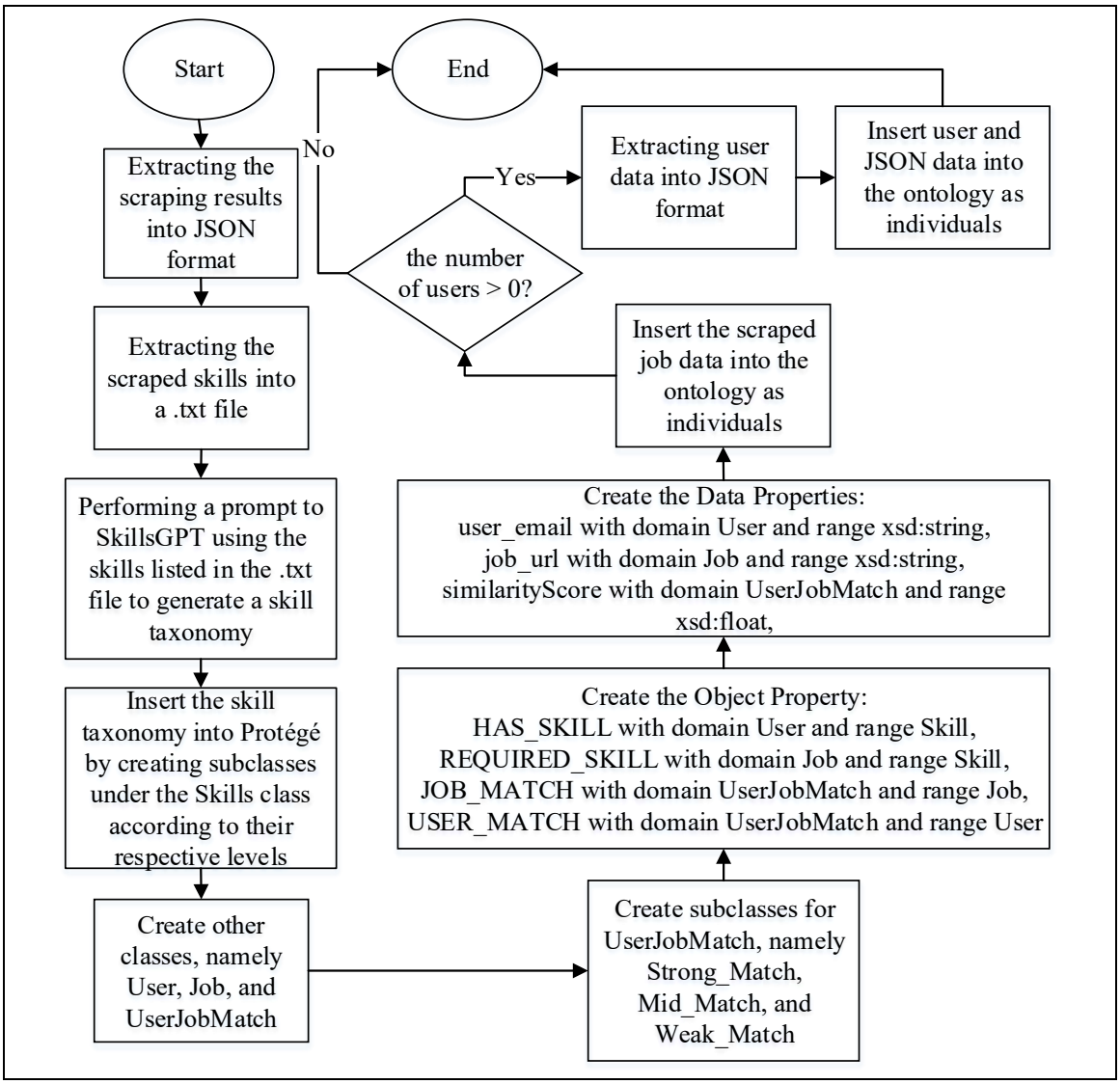

Fig. 2. Flowchart of ontology construction

Table 3. Ontology component

Ontology	Component	
	Name	**Description**
Class	User	The class that represents the User (Jobseeker) to be matched.
	Job	The class that represents job vacancies to be matched in the recommendation process.
	Skills	The class representing skills that can be matched, which may be possessed by both users and jobs.
	UserJobMatch	The class is designed to store the similarity value between a user and a job.
	Strong_Match	The class functions to categorize UserJobMatch instances with a similarity score greater than or equal to 0.65.
	Mid_Match	The class functions to categorize UserJobMatch instances with similarity values between 0.35 and 0.65.
	Weak_Match	The class functions to categorize UserJobMatch instances with a similarity score less than or equal to 0.35.
Object Property	HAS_SKILL	The object property represents the skills possessed by the user and is defined as a relation from the class User to the class Skill.
	REQUIRED_SKILL	The object property represents the skills required by a job. This property is defined as a relationship from the Job class to the Skill class.
	JOB_MATCH	The object property represents the job that has been matched, and it is established as a relation from UserJobMatch to Job within the ontology.

Ontology	Component	
	Name	**Description**
	USER_MATCH	The object property represents the user involved in the matching process and is defined from the UserJobMatch class to the User class..

SkillsGPT was provided with a .txt file containing the extracted skills and was then prompted with the instruction: 'From the provided file, generate a semantic skill hierarchy to be used as an ontology.' In response, SkillsGPT generated an initial hierarchy and offered the option to proceed or adjust the structure. The number of hierarchy levels is configurable; for the purpose of this application, a four-level structure was adopted. This decision was based on prior experiments showing that two- and three-level hierarchies resulted in ontologies that were too shallow for effective use with the Sanchez Similarity algorithm. Based on the generated hierarchy, the ontology was subsequently constructed by defining classes and subclasses using the Protégé application. Fig. 12 presents the resulting ontology graph employed during the skill-matching process including both solid blue lines and dashed lines. The solid blue lines represent the relationships between classes and their subclasses, while the dashed lines indicate the object properties (relationships) among the classes. Further explanation of the other components constructed in the ontology graph is provided in Table 3.

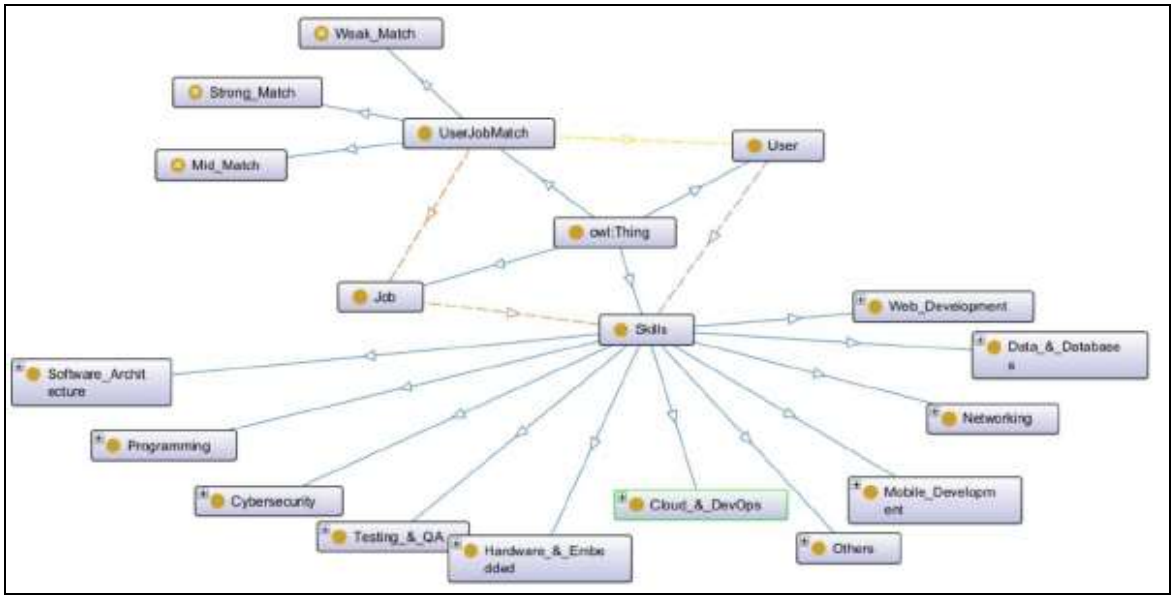

Fig. 3. DEL graph (ontology)

3.4 Transition

There are five packages used in the taxonomy construction process of NER-extracted skills to support ontology development in this study, as presented in Table 4.

Table 4. List of packages in the skill taxonomy

Skill Taxonomy	Package	
	Name	**Description**
1	api	The api/ module functions as the central logic component of the Talent Matching system, responsible for executing key processes including data scraping, skill processing, and talent-job matching
2	redis	This directory includes the Redis configuration, which serves as the message broker for handling asynchronous tasks via Celery. It enables the execution of background processes, including operations such as data scraping and matching.
3	services	Stores complex business logic that is not suitable to be directly embedded within views, such as processing the output of Named Entity Recognition (NER) or skill taxonomy data.
4	talent_matching_ner_model	This module is specifically designed to perform Named Entity Recognition (NER) for extracting skill entities from job advertisement texts.
5	tasks.py	Background tasks are executed asynchronously using Celery, including operations such as job data scraping, Named Entity Recognition (NER) processing, and storing the results of similarity computations.

The following outlines the testing requirements for the scraping and Named Entity Recognition (NER) processes:

- The application must enable administrators to perform job vacancy data scraping from external sources such as Glints and Kalibrr, and subsequently store the retrieved data in the database.
- The application must allow administrators to verify or delete the scraped data before it is published to the user interface.
- The application must automatically update the job vacancy data based on the latest scraping results.
- The application must provide a skill-matching feature once the scraping process is completed.

4. Conclusion

The talent matching application was successfully developed in accordance with the predefined objectives, namely to identify job vacancies that are relevant to users' skills in a more integrated manner compared to manually searching across multiple sources. This was achieved through the implementation of web scraping and data extraction using Named Entity Recognition (NER). Furthermore, the application enhances the relevance of job matching by employing an ontology-based model with Sanchez Similarity.

Acknowledgment

The authors express their deepest gratitude to Politeknik Negeri Bandung for the funding support provided through the Research Scheme for Laboratory Capacity Enhancement (PPKL) 2024/2025, which enabled the successful execution of this research and the writing of this scientific paper.

Declarations

This research was carried out under the coordination of Ade (principal investigator), with significant contributions from Rizki, Ikhsan, and Afyar (student researcher), as well as Urip, Wulan, and Riza (lecturer researchers), and support from fellow academic staff members, including Nanda and Hashri. This research was funded by Politeknik Negeri Bandung (POLBAN) under the Director's Decree Number 107.16/R7/PE.01.03/2025. The authors declare no conflict of interest. No additional information is available for this publication.

Data and Software Availability Statements

All data and software supporting the findings of this study are publicly available at the following GitHub https://github.com/RizkiGunawan23/talent-matching-server.

References

[1] Z. Elkaimbillah, B. El Asri, M. Mikram, and M. Rhanoui, "Construction of an Ontology-based Document Collection for the IT Job Offer in Morocco," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 7, pp. 453–461, 2023, doi: 10.14569/IJACSA.2023.0140749.

[2] PT. Glints Indonesia Group, "Lowongan Kerja Kompuer dan Teknologi Informasi (TI)," 2025. https://glints.com/id/job-category/computer-technology (accessed May 04, 2025).

[3] Kalibrr Technology Ventures Inc., "IT and Software," 2025. https://www.kalibrr.id/id-ID/home/i/it-and-software (accessed May 04, 2025).

[4] A. Habous and E. H. Nfaoui, "A fuzzy logic and ontology-based approach for improving the CV and job offer matching in recruitment process," Int. J. Metadata, Semant. Ontol., vol. 15, no. 2, pp. 104–120, 2021, doi: 10.1504/IJMSO.2021.120278.

[5] S. Paudel and A. Shakya, "Ontology based Job-Candidate Matching using Skill Sets," Proc. IOE Grad. Conf., vol. 8914, pp. 251–258, 2017.

[6] S. ScholarWorks and Y. Bashyam Balachander, "Ontology Based Technical Skill Similarity," Ontol. Based Tech. Ski. Similarity, 2018.

[7] A. Shakya and S. Paudel, "Job-Candidate Matching using ESCO Ontology," J. Inst. Eng., vol. 15, no.

1, pp. 1–13, 2019, doi: 10.3126/jie.v15i1.27699.

[8] X. Zheng, "Import and edit ontology with Neo4j," vol. 1613, no. 2020, 2022.

[9] T. Petrican, C. Stan, M. Antal, I. Salomie, T. Cioara, and I. Anghel, "Ontology-based skill matching algorithms," Proc. - 2017 IEEE 13th Int. Conf. Intell. Comput. Commun. Process. ICCP 2017, no. January 2020, pp. 205–211, 2017, doi: 10.1109/ICCP.2017.8117005.

[10] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," Expert Syst. Appl., vol. 39, no. 9, pp. 7718–7728, 2012, doi: 10.1016/j.eswa.2012.01.082.

[11] M. A. Khder, "Web scraping or web crawling: State of art, techniques, approaches and application," Int. J. Adv. Soft Comput. its Appl., vol. 13, no. 3, pp. 144–168, 2021, doi: 10.15849/ijasca.211128.11.

[12] T. Gelar, A. Nanda, and A. Bakhrun, "Serverless Named Entity Recognition untuk Teks Instruksional Pertanian Kota," J. Tek. Inform. dan Sist. Inf., vol. 8, no. 3, pp. 597–606, 2022, doi: 10.28932/jutisi.v8i3.5447.

[13] C. Larman, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition)," p. 656, 2001, [Online]. Available: http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0130925691.

[14] A. Thukral, S. Dhiman, R. Meher, and P. Bedi, "Knowledge graph enrichment from clinical narratives using NLP, NER, and biomedical ontologies for healthcare applications," Int. J. Inf. Technol., vol. 15, no. 1, pp. 53–65, 2023, doi: 10.1007/s41870-022-01145-y.

[15] T. Pratama and Suharjito, "IndoXLNet: Pre-Trained Language Model for Bahasa Indonesia," Int. J. Eng. Trends Technol., vol. 70, no. 5, pp. 366–380, 2022, doi: 10.14445/22315381/IJETT-V70I5P240.

[16] "SkillsGPT," 2025. https://www.skillsgpt.ai/.

[17] Buildmedia.readthedocs.org, "protege-tutorial Documentation," 2017, [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/go-protege-tutorial/latest/go-protege-tutorial.pdf.

[18] T. Derave, F. Gailly, T. P. Sales, and G. Poels, "A taxonomy and ontology for digital platforms," Inf. Syst., vol. 120, no. October 2023, p. 102293, 2024, doi: 10.1016/j.is.2023.102293.

[19] "Neosemantics(n10s) User Guide - Neosemantics." https://neo4j.com/labs/neosemantics/4.0/ (accessed Nov. 11, 2021).

[20] J. Barrasa and A. Cowley, "Neosemantics (n10s): A Linked Data Toolkit for Neo4j," YouTube, 2020, [Online]. Available: https://www.youtube.com/watch?v=LO-OvQaBq8s&mkt_tok=eyJpIjoiT0RkbE16UmxaVEF3TXpJeiIsInQiOiJ2bHZHdFZCV0xMRVRlWGw2U3dDdXk1NVNhK0JzT2xQT01lTmc5RGw2RWUzWWNCZ0RRRkpcL2x6ZE9FNmViajBzbG0yYWlCWFRucndlbHBDRWd2bVVWeGVVMmZnY3g5TnZHWmZDa044WUp3M1R2RlwvMkErcERnYWdRVk8.

[21] Tecoholic, "NER Annotator," 2025. https://arunmozhi.in/ner-annotator.

[22] Explosion, "spaCy," 2025. https://spacy.io/usage/training.

[23] Kamran, "Roadmap.sh," 2024. https://roadmap.sh/computer-science.