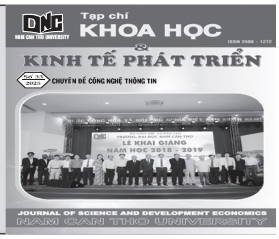
Journal of Science and Development Economics
Nam Can Tho University

Website: jsde.nctu.edu.vn



Matching ontology using machine learning

Nguyen Thai Kiet[1*], Tran Van Thien[1]

[1]*Nam Can Tho University*

[*]*Corresponding author: Nguyen Thai Kiet (email: kiet692002@gmail.com)*

Received: 8/1/2025
Revised: 24/1/2025
Accepted: 15/2/2025

***Keywords**: jaccard, levenshtein distance, machine learning, ontology matching, overfitting, word2Vec*

***Từ khóa:** jaccard, khoảng cách levenshtein, máy học, Word2Vec*

ABSTRACT

Ontology matching, a field within data management and information retrieval, utilizes machine learning to integrate and link different data sources by identifying equivalences and relationships between concepts across various ontologies. This report provides an overview of key techniques and methods in ontology matching, including vectorization and embeddings such as Word2Vec, similarity measure like Levenshtein Distance and Jaccard Index, and machine learning models that improve accuracy in classifying and matching concepts. These models use classification and regression tasks to categorize concept pairs based on numerical features. To handle heterogeneous data and minimize overfitting, multiple decision trees are employed, offering robust methods for predicting equivalences between concepts and addressing nonlinear relationships. While ontology matching has diverse applications, from data integration to intelligent search, the field still faces challenges such as semantic differences and data complexity. This report highlights emerging research trends and technologies, as well as potential future developments.

TÓM TẮT

Ontology matching (khớp nối ontology)bằng học máy là một lĩnh vực trong quản lý dữ liệu và truy xuất thông tin, hỗ trợ tích hợp và liên kết các nguồn dữ liệu khác nhau bằng cách xác định các tương đương và mối quan hệ giữa các khái niệm trong các ontology khác nhau. Báo cáo này cung cấp tổng quan về các kỹ thuật và phương pháp chính trong khớp nối ontologies, bao gồm vector hóa và nhúng như Word2Vec, các phép đo độ tương đồng như khoảng cách

pháp khác. Ngoài ra, các mô hình học máy được áp dụng để cải thiện độ chính xác trong phân loại và khớp nối khái niệm, với các bài toán phân loại và hồi quy giúp phân loại các cặp khái niệm dựa trên đặc trưng số. Nhiều cây quyết định được sử dụng để xử lý dữ liệu không đồng nhất và giảm thiểu hiện tượng quá khớp, cung cấp các phương pháp mạnh mẽ để dự đoán sự tương đương giữa các khái niệm với khả năng xử lý các mối quan hệ phi tuyến tính. Mặc dù khớp nối ontologies có nhiều ứng dụng đa dạng, từ tích hợp dữ liệu đến tìm kiếm thông minh, nhưng lĩnh vực này vẫn đối mặt với các thách thức như sự khác biệt về ngữ nghĩa và độ phức tạp của dữ liệu. Báo cáo này nêu bật các xu hướng nghiên cứu và công nghệ mới nổi, cùng với những phát triển tiềm năng trong tương lai.

Levenshtein và chỉ số Jaccard, cùng với các phương

1. INTRODUCTION

1.1 Context

Ontology matching is an interactive tool for the Semantic Web, as well as a useful technique in various traditional data integration tasks involving semantic heterogeneity issues. It takes ontologies as input and produces an alignment as output, which is a set of correspondences between semantically related entities of those ontologies. These correspondences can be used for various tasks, such as ontology merging, data linking, query answering, or navigating through knowledge graphs. Consequently, ontology matching enables knowledge and data represented by matched ontologies to interact with each other. Different methods are applied to identify equivalences and relationships between concepts from various ontologies. Traditional approaches often rely on similarity measurement techniques and logical rules, but they frequently encounter difficulties when dealing with complex and heterogeneous data.

This study focuses on applying machine learning to ontology matching because machine learning models have the capability to handle large volumes of data, detect patterns and nonlinear relationships, and improve the accuracy of classifying and predicting equivalences between concepts (OM-2024,2024).

1.2 Reason for choosing the topic

Today, ontology matching is still predominantly performed manually, leading to a labor-intensive and error-prone process. Manual matching is currently a major bottleneck in building large-scale information management systems. The development of technologies such as the WWW, XML, and the Semantic Web has not only facilitated information sharing but also exacerbated issues related to data matching and integration. Therefore, the development of tools to support the ontology matching process has become crucial for the success of many information management applications. To address this challenge, this study develops a

system that uses machine learning techniques to semi-automatically create semantic mappings between ontologies [1].

1.3 Objective of the topic

The objective of this study is to apply advanced machine learning models to improve the ontology matching process, overcoming the limitations of current manual methods. Specifically, the research will evaluate the effectiveness of models such as Logistic Regression, Random Forest, and XGBoost in identifying equivalences and relationships between concepts from different ontologies. By applying machine learning, the study aims to enhance accuracy, minimize errors, and improve data integration capabilities in large-scale information management systems.

1.4 Purpose of the topic

The goal of this research is to harness the potential of machine learning techniques to enhance the ontology matching process, with the aim of creating an efficient automated method for data integration across different sources. The study seeks to develop a solution that can be applied in real-world systems, such as healthcare information systems, enterprise management, and data-sharing platforms, to address the time-consuming and error-prone nature of manual data integration. By employing modern machine learning models, the research aims to improve synchronization and information integration, thereby supporting organizations in managing and utilizing data more effectively.

2. RESEARCH METHODS

Synthesizing and analyzing scientific papers, reports, books, and online resources related to Machine Learning and Ontology Matching is a crucial process for building a solid theoretical and practical foundation for research in this field. Below are some key points to consider when conducting this process [2],[3],[4].

2.1 Basic concept

An ontology is a formal description of knowledge in the form of a set of concepts within a specific domain and the relationships between them. To create such a description, we need to formally specify components such as individuals (instances of objects), classes, properties, and relationships, as well as constraints, rules, and axioms. This enables an ontology to not only provide a shareable and reusable knowledge representation but also to incorporate new knowledge about the domain. As a key component of Semantic Technology and the W3C standard for the Semantic Web, ontologies provide a structure for linking information on the Web of Linked Data. They enable the modeling of data from different systems and databases, support interoperability across databases, facilitate cross-database searching, and enhance effective knowledge management [5].

Machine Learning is a branch of artificial intelligence (AI) and computer science that focuses on using data and algorithms to enable systems to mimic human learning and gradually improve their accuracy over time [6]. Ontology matching using machine learning involves employing algorithms to identify similarities between two structured datasets, or ontologies. You can think of these datasets as two different maps of the same city, but drawn by different people. The task of ontology matching is to find corresponding locations on both maps, such as determining that “trường học” on map A corresponds to “school” on map B.

Previous research on ontology matching has employed various methods to align entities between ontologies. For example, [7] utilized a method called TEXTO (TEXT-based Ontology Matching). This approach focuses on using the textual descriptions of classes within ontologies for matching, rather than relying on the structural information of the ontologies.

2.2 Matching system

The system collects data from various sources, such as training and testing datasets, as well as ontology datasets, for analysis and comparison. Data preprocessing is performed to prepare the data, including cleaning, transforming, and extracting features from the ontologies to make them ready for the training and testing phases. Machine learning models such as Random Forest, Logistic Regression, and XGBoost are selected to train and evaluate the performance of the models in the ontology matching process.

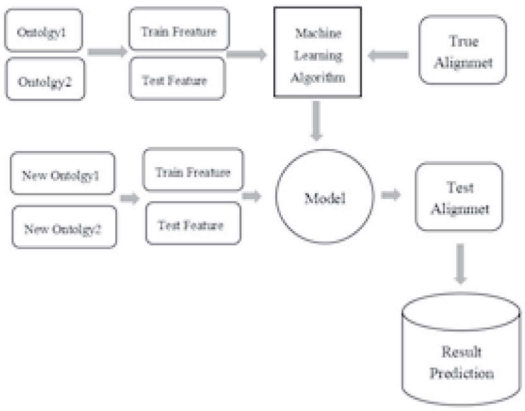


Figure 1. Matching system

2.3 Data collection

Two datasets are selected for the evaluation studies, referred to as Dataset1 and Dataset2. These datasets include entities and their actual correspondences, sourced from the Ontology Alignment Evaluation Initiative (OAEI). Some entity pairs and their actual correspondences are chosen for training and testing the machine learning models. This process is known as data splitting.

Table 1. Data source

Details of data collected from 2 datasets.

Dataset	Type	Total number of samples	Matching pattern	Sample does not match
Dataset1	Train	14,148	156	13,992
Dataset1	Test	14,940	172	14,768
Dataset2	Train	55,348	284	55,064
Dataset2	Test	114,045	253	113,792

2.4 Data processing and training algorithms

The following methods will be limited by the following constraints: entities can only be matched through equivalence relationships, classes can only be matched with classes, properties can only be matched with properties, and instance entities are not used. This approach includes two main algorithms: training machine learning models and using the models to predict matches [9].

Input:

ontology1, *ontology2* - input ontologies,

THRESHOLD - threshold for create matching between entities

Auxiliary functions:

get_classes - get list of classes,

get_properties - get list of properties,

create_alignment - Algorithm 2

Output: *final_alignment* - output alignment for *ontology1* and *ontology2*

```
classes1 ← get_classes(ontology1)
classes2 ← get_classes(ontology2)
alignment_classes ← create_alignment(classes1, classes2, THRESHOLD)
properties1 ← get_properties(ontology1)
properties2 ← get_properties(ontology2)
alignment_properties ← create_alignment(properties1, properties2, THRESHOLD)
final_alignment ← alignment_classes ∪ alignment_properties
return final_alignment
```

Figure 2. Algorithm: Match original data

Input:
entities1, *entities2* - input lists of entities (classes or properties),
THRESHOLD - threshold for create matching between entities
Auxiliary functions:
calculate_all_sim_measures - Algorithm 3,
predict_match - predict confidence based on similarity measures
Output: *alignment* - output alignment for *entities1* and *entities2*

```
for entity1 ∈ entities1 do
    for entity2 ∈ entities2 do
        sim_measures ← calculate_all_sim_measures(entity1, entity2)
        match ← predict_match(sim_measures)
        if match > THRESHOLD then
            alignment ← alignment ∪ (entity1, entity2)
        end if
    end for
end for
return alignment
```

Figure 3. Algorithm: Create predictive association from two entity lists

Input:
entity1, *entity2* - input entities (classes or properties)
Auxiliary functions:
get_name - get name of entity,
get_parent - get parent of entity,
get_path - get full path of entity,
calculate_sim_measures - calculates string-based and linguistic-based similarity measures listed in 4.2 and returns a list of 88 values
concat - merge lists
Output: *sim_measures* - output list of calculated similarity measures for *entity1* and *entity2*

```
name1 ← get_name(entity1)
name2 ← get_name(entity2)
parent1 ← get_parent(entity1)
parent2 ← get_parent(entity2)
path1 ← get_path(entity1)
path2 ← get_path(entity2)
name_sim_measures ← calculate_sim_measures(name1, name2)
parent_sim_measures ← calculate_sim_measures(parent1, parent2)
path_sim_measures ← calculate_sim_measures(path1, path2)
sim_measures ← concat(name_sum_measures, parent_sim_measures, path_sim_measures)
return sim_measures
```

Figure 4. Algorithm: Applying measures to data

Input:

train_pairs_ontologies - set of tuples *(ontology1, ontology2, true_alignment)*

model_name - name of machine learning method (logistic regression, random forest, gradient boosting),

model_params - set of parameters of machine learning model,

create_dataset - Algorithm 5

Auxiliary functions: *train_model* - train machine learning model on training dataset

Output: *model* - trained model for predicting matching

```
for ontology1, ontology2, true_alignment in train_pairs_ontologies do
    classes1 ← get_classes(ontology1)
    classes2 ← get_classes(ontology2)
    train_dataset_classes ← create_dataset(classes1, classes2, true_alignment, 'Class')
    properties1 ← get_properties(ontology1)
    properties2 ← get_properties(ontology2)
    train_dataset_properties ← create_dataset(properties1, properties2, true_alignment, 'Property')
    train_dataset ← train_dataset ∪ train_dataset_classes ∪ train_dataset_properties
end for
model ← train_model(train_dataset, model_name, model_params)
return model
```

Figure 5. Algorithm: Model training algorithm

Input:
true_alignment - set of matched pairs of entities,
entities1, *entities2* - input lists of entities (classes or properties),
type_entity - type of input entities (class or property)
Auxiliary functions: *train_model* - train machine learning model on training dataset
Output: *train_dataset* - output list of tuples with pairs of entities, their matchings and similarity measures

```
for entity1 ∈ entities1 do
    for entity2 ∈ entities2 do
        sim_measures ← calculate_all_sim_measures(entity1, entity2)
        if (entity1, entity2) ∈ true_alignment then
            train_dataset ← train_dataset ∪ (entity1, entity2, 1, type_entity,
                                             sim_measures)
        else
            train_dataset ← train_dataset ∪ (entity1, entity2, 0, type_entity,
                                             sim_measures)
        end if
    end for
end for
return train_dataset
```

Figure 6. Algorithm: Create the training dataset

2.4 Implement training process

This method is implemented using Python, a widely used programming language in machine learning workflows. Python is favored not only for its simplicity and flexibility but also for its rich ecosystem of useful libraries. Ontologies are represented as RDF/OWL files. The owlready2 library is used to parse these ontologies. owlready2 is a powerful Python library designed for working with OWL files. It provides convenient methods for loading, accessing, and manipulating ontology components, as well as supporting reasoning. owlready2 can perform tasks such as loading ontologies from file systems, and accessing classes and properties

In the ontology matching process, matching results are often defined in RDF (Resource Description Framework). RDF is a standard framework for representing information about resources on the web. To process and analyze these matching results, the BeautifulSoup library can be used, which is a powerful tool for parsing and analyzing HTML and XML documents. RDF represents information about web resources using triples, which describe relationships between objects. Each triple consists of a subject, a predicate, and an object. In the context of ontology matching, the matching results are typically represented as RDF triples, where each triple describes a matching relationship between entities from two different ontologies. To address the analysis of results, the BeautifulSoup library will assist in parsing RDF data and extracting relevant information.

During training, selecting optimal parameters for machine learning models to maximize their performance involves identifying and tuning important parameters. For Random Forest, key parameters include the number of trees (n_estimators), the maximum number of features (max_features), and the depth of the trees (max_depth). Logistic Regression mainly focuses on the regularization parameter (C) and the type of regularization (L1 or L2). XGBoost requires optimizing parameters such as the number of trees (n_estimators), learning rate (learning_rate), tree depth (max_depth), and sample ratio (subsample). Using techniques like Grid Search or Random Search combined with Cross-Validation helps in selecting the best parameter values, thereby enhancing the models' effectiveness and accuracy.

2.5 Characteristics calculation (corresponding measurements)

In the feature computation process of this study, a range of string-based similarity measures and feature computation methods are applied to evaluate the similarity between strings from the ontologies. These measures and methods include:

String-based: Utilizing various string similarity measures from previous works. Each

method is designed to handle different scenarios and contexts

Language-based: In cases where different words have nearly the same meaning, such as "car" and "automobile," WordNet can address this issue. Wu and Palmer similarity is used to handle such situations. If a string contains multiple words, the highest similarity score among all word pairs in the sets is selected. However, a limitation of WordNet is that it only contains a subset of the vocabulary of the language. To address this, word vector representations from the Word2Vec model are used. The cosine similarity between the word vector representations is computed to determine similarity.

Structure-based: Additionally, structure-based similarity measures are utilized. These measures are applied to assess similarity based on the hierarchical relationships between parent classes of entities and the paths between entities. The similarity measures assume that entities being matched have similar parent classes and occupy similar positions within the hierarchy. Since the same model is used for matching both classes and properties, an additional column labeled "Type" is introduced, where the label "1" represents a class and the label "0" represents a property.

String-based	N-gram 1, N-gram 2, N-gram 3, N-gram 4, Dice coefficient, Jaccard similarity, Jaro measure, Monge-Elkan, Smith,Waterman, Needleman-Wunsh, Affine gap, Bag distance, Cosine similarity, Partial Ratio, Soft TF-IDF, Editex, Generalized Jaccard, Jaro-Winkler, Levenshtein distance, Partial Token Sort, Fuzzy Wuzzy Ratio, Soundex, TF-IDF, Token Sort, Tversky Index, Overlap coefficient, Longest common.
Language-based	Wu and Palmer similarity, Word2vec and Sentence2vec similarity
Structure-based	All string-based and language-based similarity measures between parents of entities All string-based and language-based similarity measures between paths of entities

Figure 7. Corresponding measurements

2.6 Threshold of training model

Selecting the Threshold and Optimal Parameters: Choosing the threshold and optimal parameters is a crucial step to ensure the model's performance. The study focused on identifying the threshold with the highest F-measure value for each matching and creating a parameter grid to train the model.

Threshold Search Process: After training the machine learning models and finding the best parameters, the threshold selected is the one that yields the highest F-measure for each matching. This approach ensures that the model achieves the best balance between precision and recall. In ontology matching problems, models often predict the degree of similarity between classes from two different ontologies in the form of probabilities. For example, if a model predicts a probability of 0.8 that a class in ontology A corresponds to a class in ontology B, a classification threshold of 0.5 can be set. If the predicted probability exceeds the threshold, the two classes are considered a match; if it is below the threshold, they are not considered a match.

3. RESULTS AND DISCUSSION

The F-measure results for each match are presented in Tables 2, 3, and 4. On Dataset1, the logistic regression and random forest models demonstrated the best performance. Although the gradient boosting model had slightly lower accuracy, it remained competitive. However, on Dataset2, gradient boosting emerged as the best-performing model, with accuracy higher than logistic regression by approximately 0.02 and higher than random forest by approximately 0.01.

Table 2. F-measure values of models in Dataset1

Alignment	Logistic Regression	Random Forest	XGBoost
101-302	0.72	0.71	0.72
101-303	0.82	0.82	0.75
101-304	0.90	0.91	0.91
Average	0.81	0.81	0.79

The results show that Logistic Regression and Random Forest achieved the highest average performance (0.81), surpassing XGBoost (0.79). For the 101-302 alignment, Logistic Regression and XGBoost performed best (0.72), while Random Forest excelled in the 101-303 alignment (0.82). In the 101-304 alignment, XGBoost and Random Forest led with 0.91. Logistic Regression and Random Forest demonstrated stability across alignments, while XGBoost was better suited for more complex cases. The choice of model should depend on the specific characteristics of the dataset.

Table 3. F-measure values of models in Dataset2

Alignment	Logistic Regression	Random Forest	XGBoost
conference-edas	0.53	0.5	0.55
cmt-sigkdd	0.73	0.8	0.73
edas-sigkdd	0.53	0.63	0.63
ekaw-sigkdd	0.77	0.77	0.77
cmt-edas	0.72	0.76	0.63
conference-sigkdd	0.64	0.54	0.58
confof-edas	0.62	0.62	0.62
confof-iasted	0.71	0.61	0.66
conference-confof	0.61	0.54	0.57
cmt-confof	0.44	0.41	0.48
conference-ekaw	0.43	0.40	0.47
cmt-ekaw	0.58	0.62	0.70
confof-ekaw	0.58	0.68	0.64

iasted-sigkdd	0.75	0.81	0.81
cmt-iasted	0.88	0.88	0.88
edas-iasted	0.42	0.57	0.57
ekaw-iasted	0.58	0.75	0.70
Average	0.62	0.64	0.65

The results demonstrate that XGBoost achieved the highest overall average performance (0.65), slightly surpassing Random Forest (0.64) and Logistic Regression (0.62). In specific alignments, XGBoost excelled in cases such as cmt-ekaw (0.70) and conference-ekaw (0.47), while Logistic Regression and Random Forest performed better in alignments like cmt-iasted (0.88), where all models were equally strong. Random Forest led in several alignments, including cmt-sigkdd (0.80) and ekaw-iasted (0.75), highlighting its robustness in certain contexts. Logistic Regression showed stable performance across alignments but lagged behind in more complex ones. Overall, XGBoost's adaptability to diverse alignments and slightly higher average makes it a competitive choice for ontology matching [8].

Table 4. Average value of measurement over 2 data sets F-measure

Alignment	Logistic Regression	Random Forest	XGBoost
Average	0.62	0.64	0.65

The summarized average scores highlight that XGBoost achieved the highest overall performance with a score of 0.65, followed closely by Random Forest at 0.64, and Logistic Regression at 0.62. This suggests that while all models performed competitively, XGBoost demonstrated slightly better adaptability and accuracy in ontology matching tasks.

3.1 Matching new data

This matching process helps to integrate data from multiple sources and improve the completeness and accuracy of knowledge systems. Machine learning algorithms, which have been trained, are used to perform ontology matching on a new dataset. The dataset used in this research includes precomputed features stored in a CSV file. First, read the CSV file and create a new feature called "Type_encode" to encode the type of each entity in the dataset. If the type is 'Class', the encoding value will be 1; otherwise, it will be 0. Next, select the necessary features and replace any NaN values with 0 to prepare for the process.

Load the pre-trained model from the file and use it to predict match probabilities for the entities in the dataset. Use a threshold of 0.5 to determine whether a pair of entities is considered a match. Based on the predicted probability, if it is greater than the threshold, label the entity pair as 'Match' with a value of 1. Finally, calculate the number of successfully matched entity pairs and their percentage relative to the total number of entity pairs. This result helps to evaluate the performance of the machine learning models in the ontology matching task.

3.2 Jaccard index

The Jaccard formula is used to measure the similarity between two sets and can be applied to calculate the similarity between two ontologies based on the concepts that have been matched

between them. The Jaccard formula for two sets A and B is defined as:

The Jaccard index algorithm, also known as the Jaccard similarity coefficient, is a method proposed by the French mathematician Paul Jaccard. This algorithm calculates the similarity between two sets based on the ratio of the intersection to the union of the sets. Specifically, the Jaccard index is computed as follow: (Hoang Duc Quan, 2020, VIBLO) [10].

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

Figure 8. Jaccard index algorithm

In the context of ontology matching, the Jaccard index formula is adapted to account for the specific structure of ontologies. Instead of directly comparing raw sets, the formula considers the structure and relationships within the ontologies. Here's how the Jaccard index is modified for ontology matching:

$$\text{Jaccard Index} = \frac{M}{O_S + O_T - M}$$

Figure 9. Jaccard index ontology

In this formula, M represents the number of matched object pairs between the ontologies. OS denotes the total number of classes and properties in the source ontology, while OT indicates the total number of classes and properties in the target ontology. The Jaccard index quantifies the similarity between two ontologies by evaluating the proportion of matched pairs relative to the total number of unique entities across both ontologies.

4. CONCLUSION

The research and application of machine learning in ontology matching have yielded significant insights into various metrics, machine learning models, and the concept of ontology itself. Machine learning has proven to be effective and accurate in automating ontology matching, minimizing manual intervention, and accelerating processing speeds. Models such as logistic regression, random forest, and XGBoost have demonstrated their ability to learn from data and accurately identify similar entity pairs. Moreover, this approach shows high flexibility by integrating various types of similarity measures (string-based, character-based, language-based, structural) and has the potential to scale for handling large ontologies. Machine learning not only reduces manual intervention but also automates the matching process, enhancing speed and minimizing errors. Future directions include the need to further improve existing algorithms and develop more advanced machine learning models to enhance the effectiveness and accuracy of ontology matching. Additionally, integrating diverse data sources and developing web/app systems could make ontology matching using machine learning more accessible and user-friendly for a broader audience.

REFERENCES

[1] Madhavan, J., Domingos, P., & Halevy, A. Ontology Matching: *A MachineLearningApproach. University of Washington, Seattle, USA.* Retrieved from https://homes.cs.washington.edu/~pedrod/papers/hois.pdf

[2] SWJ (2023). Ontology Matching and Machine Learning. Special issue on Ontology Matching and Machine Learning. Semantic Web – Interoperability, Usability. *Applicability an IOS Press Journal.*

Retrived from https://www.semantic-web-journal.net/blog/special-issue-ontology-matching-and-machine-learning.

[3] Factil (2023). *Data Matching: Advanced algorithms for record matching and entity resolution.* Official website of Factil. Data Matching. Retrieved from https://www.factil.io/technology/matching/.

[4] Samuel, S., König-Ries, B., & Algergawy, A. (2023). The role of Ontology Matching in Ontology Network Development. *The 18th International Workshop on Ontology Matching collocated with the 22nd International Semantic Web Conference ISWC-2023 November 7th, 2023, Athens, Greece.* Retrieved from https://ceur-ws.org/Vol-3591/om2023_LTpaper2.pdf

[5] Ontotext (2018). *What Are Ontologies?. Official web site of Ontotext ltd.* Retrieved from https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies

[6] IBM. (2019). *Machine Learning. Official website of IBM.* Retrieved from https://web.archive.org/web/20240000000000*/https://www.ibm.com/topics/machine-learning

[7] Peng, Y., Alam, M., & Bonald, T. (2023). Ontology Matching using Textual Class Descriptions. *The 18th International Workshop on Ontology Matching collocated with the 22nd International Semantic Web Conference ISWC-2023 November 7th, 2023 Athens, Greece.* Retrieved from https://ceur-ws.org/Vol-3591/om2023_STpaper2.pdf

[8] OAEI (2023). *Ontology Alignment Evaluation Initiative. Official website of OAEI*. Retrieved from https://oaei.ontologymatching.org/.

[9] Bulygin, L., and Stupnikov, S. (2019). Applying of Machine Learning Techniques to Combine String-based, Language-based and Structure-based Similarity Measures for Ontology Matching. *International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2019): pp.129-147. Russia.* Retrieved from https://ceur-ws.org/Vol-2523/paper14.pdf.

[10] Hoang Duc Quan (2020). *Giải thuật Jaccard. VIBLO: Free service for technical knowledge sharing.* Retrieved from https://viblo.asia/p/giai-thuat-jaccard-djeZ1P9GKWz