Applying of Machine Learning Techniques to Combine String-based, Language-based and Structure-based Similarity Measures for Ontology Matching

Lev Bulygin[1][0000-0003-3244-1217] and Sergey Stupnikov[2][0000-0003-4720-8215]

[1] Lomonosov Moscow State University, Moscow, Russia
buliginleo@yandex.ru

[2] Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia
sstupnikov@ipiran.ru

Abstract. In the areas of Semantic Web and data integration, ontology matching is one of the important steps to resolve semantic heterogeneity. Manual ontology matching is very labor-intensive, time-consuming and prone to errors. So development of automatic or semi-automatic ontology matching methods and tools is quite important. This paper applies machine learning with different similarity measures between ontology elements as features for ontology matching. An approach to combine string-based, language-based and structure-based similarity measures with machine learning techniques is proposed. Logistic Regression, Random Forest classifier and Gradient Boosting are used as machine learning methods. The approach is evaluated on two datasets of Ontology Alignment Evaluation Initiative (OAEI).

Keywords: ontology matching, machine learning, similarity measures.

1 Introduction

An *ontology* is "a formal, explicit specification of shared conceptualization" [1], where conceptualisation is an abstract model of some phenomenon in the world. Ontologies were created to facilitate the sharing of knowledge and its reuse [2]. They are used for organization of knowledge and for communication between computing systems, people, computing systems and people [3]. Ontologies deal with the following kinds of e*ntities:* classes, properties and individuals. A *class* (concept) of ontology is a collection of objects, i.e., "Person" (the class of all people) or "Car" (the class of all cars). *Property* (attribute) describes characteristics of a class or relations between classes, i.e., "has as name" or "is created by". *Individual* (instance) is a particular instance or object represented by a concept, i.e., "a human cytochrome C" is an instance of the concept "Protein" [4].

Ontology matching is a process of establishing correspondences between semantically related entities in different ontologies [4]. A set of correspondences

(equivalence, subsumption, disjointness) between ontologies elements is called an *alignment*. Ontology matching can be applied in many different subject areas: Semantic Web, Peer-to-Peer (P2P) systems, learning systems, multi-agent systems [5] [6].

In the *Semantic Web*, ontologies are used to extract logical conclusions from data. Many ontologies on the same subject areas have been created recently. These ontologies have a different format and they cannot exchange information, so it is necessary to apply ontology matching [4]. In *P2P systems* ontology matching is used to reduce the semantical heterogeneity (differences in the interpretation of the meaning) between the queries of the users to system [7]. In *learning systems* ontology matching is a way to ease the knowledge share and reuse [8]. In *multi-agent systems*, ontology matching is used for interaction of different agents [9].

Ontology matching can be used also for schema mapping during data integration [10]. *Data integration* is a process of combining the heterogeneous data sources into a unified view. *Schema mapping* is a process of establishing correspondences between elements of two different semantically related schemas (i.e. database schemas) [11]. Ontology matching can help to resolve semantical heterogeneity during schema mapping, for instance, if the schemas have ontologies as metadata or external domain knowledge [12] [13].

The classification of ontology matching approaches is very similar to the classification of schema matching approaches [11] [10] [4]. Matchers can be *individual* (use single matcher criterion) or *combining* (combination of individual matchers). An individual matcher can be *schema-based* (uses information about classes, properties and their relationships) or *instance-based* (uses information about instances/content). Schema-based matchers are divided into *element-level* (uses information about element without its relationships) and *structure-level* (uses information about structure and hierarchy). Combining matchers can be *hybrid* (creates alignment using several matching criteria in sequentially) or *composite* (combines several independent matching results). Composite matchers are divided into *manual composition* matchers and *automatic composition* matchers [11].

This paper proposes an approach for combining individual element-level and structure-level matchers into an automatic composition matcher based on machine learning. Individual matchers produce similarity measures between ontology elements. In terms of machine learning, similarity measures are used as *features* [14] [15]. The composite matcher is a machine learning model trained on these features. Logistic Regression, Random Forest and Gradient Boosting are used as the machine learning methods in the paper. The idea of the approach is as follows: to combine a large number of different similarity measures from other papers [32][33][34] in hope of increasing the universality of the approach, that is, applicability to different subject areas.

The paper is structured as follows. In Section 2 related works on application of machine learning for ontology and schema matching are reviewed. Section 3 describes the formal problem statement and an evaluation metric. In Section 4, similarity measures and machine learning techniques applied are listed. Section 5 considers implementation and evaluation issues.

2 Related Work

This section reviews related works on schema matching and ontology matching because they often applies similar techniques.

In [11], a classification of approaches for schema matching is introduced and a review of existing systems for schema matching is conducted. In [4], the authors described the classification of ontology matching approaches based on [11], existing matching systems, evaluation methods, similarity measures and matching strategies. The most promising option is the apply combining matcher because it uses much more information than an individual matcher. Many papers showed that the combining matchers are more accurate than individual ones [16] [17] [18]. Most approaches use string-based similarity measures, i.e., N-gram [17] [19] [20], Soundex [17] [21] [22], Levenshtein distance [17] [21] [23], Jaro measure [24] [25] [26] and others. Language-based similarity measures are also used, i.e., information from lexical database WordNet [19] [27] [28] or vector representation of words from Word2vec models [29] [21] [30] [31]. Some articles described structure-based similarity measures: differences between numbers of properties [15], similarity measures based on subclasses or parents [15] [19] and graph-based similarity [32].

Supervised machine learning for combining similarity measures is used in [14]. The authors describe an approach matching only concepts of ontologies. They used the string-based similarity measures (prefix, suffix, Edit distance, n-gram), language-based similarity measures (WordNet, Wu&Palmer, description, Lin), similarity measures between lists of words (for instance, name "socialNetwork" is divided into list of words ["Social", "Network"]), and structure similarity measures (string-based and language-based similarity measures between parents). Support Vector Machine (SVM) is used as a machine learning method. The authors conducted experiments with data from Ontology Alignment Evaluation Initiative 2007 (OAEI). The data is constructed from three Internet directories (Google, Yahoo and Looksmart) and contains 4639 pairs of ontologies defined using OWL language. The authors used 10-fold cross validation and got 56.1% accuracy, 52.5% precision and 92.5% recall on average.

In [55] the authors combined the various similarity measures into a input sample for the first time. String-based, linguistic-based and structure-based similarity measures are used.

In [15] language, structural and web similarity measures are used. Web similarity measure "Web-dice" is the difference between the count of pages in a search engine when searching for an entity.` An SVM method is selected for training. The dataset used is OAEI benchmark tests ontologies. The authors trained two models "SVM-Class" and "SVM-Property" for matching classes and properties respectively.

In [54] 10 string-based, linguistic-based and instance-based similarity measures are used as features. Decision Tree (DT) and Naive Bayes are used for classification. The authors achieved 0.845 F-measure value.

In [33] SVM, K-Nearest Neighbours (KNN), SVM, DT and AdaBoost are used as the machine learning methods. The authors choose OAEI ontologies #301, #102 and #103 as train dataset and ontologies #302, #303, #304 as test dataset and achieved 0.99 F-measure value.

In [34] Stoilos, Soft Jaccard and Lin similarity measures are used for names, labels and comments of entities. The authors also used information on abbreviations. Samples from "Conference" track and benchmarks from OAEI are used as datasets. Multilayer perceptron, Decision Trees and M5Rules are selected as machine learning methods. The authors achieved 0.67 F-measure value.

In [31] the authors used string-based similarity measures, measures related with parents and children of entities and chose "Conference" track from OAEI and EuroVoc dataset.

Note that known works use different datasets for their experiments and it is very hard to compare them with each other.

3 Ontology Matching as a Machine Learning Problem

3.1 Formal Problem Statement of Ontology Matching

Let ontology be a tuple *(C, P, H)*. Here *C* is a set of classes, *P* is a set of properties. *H* define the hierarchical relationships between classes. Other components of ontologies like axioms and instances are not applied for ontology matching in the paper. The objective of ontology matching is to find an alignment between classes and properties of a source ontology O_1 and a target ontology O_2. An *alignment* is a set of tuples (e_1, e_2, con), where e_1 is an entity of O_1, e_2 is an entity of O_2, and con is the confidence of the correspondence. *A predicted alignment* is the alignment obtained by ontology matching. *A true alignment* is a manual alignment conducted by an domain expert.

3.2 Ontology Matching Problem as a Machine Learning Problem

Entity pairs are extracted from source and target ontologies. Each pair of entities is assigned with a label *"0"* or *"1"*, where *"0"* means that entities do not match, *"1"* means that entities match. Thus, the problem is reduced to a machine learning binary classification problem. The authors of most of the reviewed papers and OAEI used F-measure for the evaluation of their approaches [4][33].

4 An Approach for Ontology Matching Applying Machine Learning Models Trained on Similarity Measures

This section describes machine learning techniques applied (subsection 4.1), similarity measures used (subsection 4.2) and algorithms constituting the approach (subsection 4.3).

4.1 Machine Learning Techniques

The following machine learning methods are applied in this paper: Logistic Regression, Random Forest and Gradient Boosting. In [31][47] it is shown that a powerful ensemble

method Random Forest [48] and relatively simple and interpretable Logistic Regression outperformed other machine learning algorithms like Gaussian Naive Bayes, K-nearest Neighbors Algorithm, Classification and Regression Trees for ontology matching. Gradient boosting has proven itself in many machine learning contests [49][50], so it was also selected as a machine learning method to be applied. In the future, we also want to test neural network (multilayer perceptron) as a machine learning method and an approach based on automatic machine learning[1].

4.2 Similarity Measures

String-based. We used all string-based similarity measures listed in our previous work [21]. The listed metrics are aimed at handling various sorts of scenarios. *N-gram* consider similarity of substrings and it is efficient when some characters are missing [4]. *Dice coefficient* is defined as twice the number of common words of compared strings over the total number of words in both strings [35]. *Jaccard* and *Generalized Jaccard* similarity are defined as the size of the intersection divided by the size of the union of the sample sets of words **[**24]. *Levenshtein distance* between two strings is the minimum number of single-character edits required to change one word into the other [36]. *Jaro* and *Jaro-Winkler* measures is edit distance measure designed for short strings [37]. *Monge-Elkan* is a type of hybrid similarity measure that combines the benefits of sequence-based and set-based methods [38]. The *Smith-Waterman* measure determine similar regions between two strings [35]. The *Needleman-Wunsh* distance is computed by assigning a score to each alignment between the two input strings and choosing the score of the best alignment [39]. The *Affine gap distance* is an extension of the Needleman-Wunsch measure that handles the longer gaps more gracefully [40]. The *Bag distance* is edit distance for sets of words [52]. *Cosine similarity* transforms a string into vector so Euclidean cosine rule is used to determine similarity [24]. *Fuzzy Wuzzy Partial Ratio* finds the similarity measure between the shorter string and every substring of length *m* of the longer string, and returns the maximum of those similarity measures [41]. *Soft TF-IDF* and *TF-IDF* are numerical statistics that are intended to reflect how important a word is to a document in a collection or corpus [39]. *Partial Token Sort*[2] and *Token Sort* are obtained by splitting the two strings into tokens and then sorting the tokens. The score is the fuzzy wuzzy partial ratio raw score of the transformed strings. *Fuzzy Wuzzy Ratio* is the ratio of the number of matching characters to the total number of characters of two strings [41]. *Editex* [42] and *Soundex*[3] are phonetic matching measures. *Tversky Index* is an asymmetric similarity measure on sets that compares a variant to a prototype [43]. *Overlap coefficient* is defined as the size of the intersection divided by the smaller of the size of the two sets [44].

[1] https://github.com/automl/auto-sklearn

[2] https://anhaidgroup.github.io/py_stringmatching/v0.3.x/PartialTokenSort.html

[3] http://anhaidgroup.github.io/py_stringmatching/v0.4.1/Soundex.html

Language-based. It is possible that words differ but are close in meaning, i.e., "car" and "auto". *WordNet* can solve this problem. *Wu and Palmer similarity* are used for handling this scenario [45]. If the strings consist of several words then the maximum similarity measure of all possible pairs of sets of words is taken. But the weakness of WordNet is that it contains only a part of all words of the language. Usage of vector representations of words from *Word2vec* models [46] facilitates this problem. Cosine similarity between two vector representations of words is calculated. If the strings consists of several words then *Sentence2vec* algorithm from [30] is used.

Structure-based. Additionally, structure-based similarity measures are used: all listed string-based and language-based similarity measures between parents of entities and between paths of entities. These similarity measures embrace the hypothesis that matched entities have similar parents and a similar place in hierarchy.

Since we used the same model for the match of classes and properties, we added feature "Type", in which label "1" means class and label "0" means property.

Such an extensive selection of similarity measures is aimed to get as much information as possible so that a machine learning model is able to select the best factors for prediction. Finally, we chose for each pair of entities 88 similarity measures (29 for names, 29 for parents, 29 for paths, 1 for type), which are described in Table 1.

Table 1. Similarity measures

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 subsequence
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

4.3 Training and Matching Algorithms

The approach is restricted with the following limitations: entities are matched only by equivalence relation, classes are matched only with classes, properties are matched only with properties, instances of ontologies are not used. The approach includes two main algorithms: training of a machine learning model (training phase) and using it to predict alignment (testing phase).

The ontology matching algorithm using the trained model is described as follows:

Algorithm 1 Matching algorithm

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*

```
1 classes1 ← get_classes(ontology1)
2 classes2 ← get_classes(ontology2)
3 alignment_classes ← create_alignment(classes1, classes2, THRESHOLD)
4 properties1 ← get_properties(ontology1)
5 properties2 ← get_properties(ontology2)
6 alignment_properties ← create_alignment(properties1, properties2,
  THRESHOLD)
7 final_alignment ← alignment_classes ∪ alignment_properties
8 return final_alignment
```

Here ← denotes an assignment operation, and ∪ - the operation of merging lists. The input data of the algorithm are two ontologies and a matching probability threshold for filtering pairs of entities. If the probability is higher than the threshold, then the pair is added to the alignment. A list of classes is extracted from each ontology. Next, two lists of classes are fed to the input of Algorithm 2:

Algorithm 2 Creating predicted alignment from two lists of entities - *create_alignment(entities1*, *entities2, THRESHOLD)*

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*

```
1 for entity1 ∈ entities1 do
2     for entity2 ∈ entities2 do
3         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
```

Then, each class from the first ontology is matched with each class from the second ontology. For example, if in the first ontology includes 10 classes and in the second ontology includes 12 classes, then 120 pairs are matched. Each pair is fed to the input of a machine learning model, which calculates the probability (confidence) of matching for each pair. Then the threshold is set: if the probability is above the threshold, then the pair is added to the final alignment. Similar actions are performed for properties. The similarity measures for each pair are calculated in Algorithm 3:

Algorithm 3 Calculating similarity measures algorithm - *calculate_all_sim_measures(entity1, entity2)*

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
```

Name, parent name, and the full hierarchical path are retrieved from each class. The parent of a class is its super class. The full path is a string that describe the entire hierarchy of classes: from the most general class to the current class. For example, the class “Book” has the name “Book”, the parent name “Publication” and the full path “Thing/Publication/Book”. Thus, a list of pairs for matching is generated. For properties, the parent is the class that it describes. And the full path is a string describing the complete hierarchy up to the class that describes the property. For each pair, all similarity measures listed in Section 4.2 are calculated. Then all similarity measures are combined into a list.

The algorithm of model training is described as follows:

Algorithm 4 Creating dataset and training a machine learning model

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 f**unctions**: *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
```

The input data is a list of ontology pairs and the true alignment between them. A model from Section 2.5 and its parameters are also selected. The process is similar to the first algorithm: the names of objects, the names of parents and full paths are retrieved, and the similarity measures are calculated. First, a dataset is created for the classes, then for properties, and after that the datasets are combined.

The algorithm for creating a dataset is described in Algorithm 5:

Algorithm 5 Creating dataset from two lists of entities - *create_dataset(entities1, entities2, true_alignment, type_entity)*

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 f**unctions**: *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

```
1  for entity1 ∈ entities1 do
2      for entity2 ∈ entities2 do
3          sim_measures ← calculate_all_sim_measures(entity1, entity2)
4          if (entity1, entity2) ∈ true_alignment then
5              train_dataset ← train_dataset ∪ (entity1, entity2, 1, type_entity,
                                                sim_measures)
6          else
7              train_dataset ← train_dataset ∪ (entity1, entity2, 0, type_entity,
                                                sim_measures)
8          end if
9      end for
10 end for
11 return train_dataset
```

The input is a true alignment, two lists of entities and the type of input entities. Each entity from the first list is mapped to each entity from the second list. Then, if a pair of entities is contained in the true alignment, then the pair is assigned label “1”, otherwise - label “0”. Also, each pair indicates the type of entity (either “Class” or “Property”) because the same model was used to map classes and properties. Then all pairs are combined into one dataset. Further, the model is trained on the created dataset with the selected parameters.

5 Implementation and Evaluation Results

5.1 Datasets

Two datasets are selected for evaluation experiments (called as Dataset #1 and Dataset #2 below). These datasets are sets of ontologies and their true alignments taken from Ontology Alignment Evaluation Initiative (OAEI). Some pairs of ontologies and their true alignments are selected for training the machine learning models and their testing. This selection is called a *partition*. Ontologies from OAEI are used in many papers. These papers include [33] and [34]. [33] presents an approach to combining similarity measures without instances of ontologies and user feedback. KNN, SVM, DT and AdaBoost were used as machine learning models. The authors achieve on some alignments the value of F-measure 0.99. [34] proposed a new ontology matching approach. The authors used five different similarity measures: syntactic, semantic, abbreviation and context similarity. Multilayer Perceptron, REPTree, M5Rules are used as the machine learning models. Average F-measure is 0.67. Dataset #1 is a partition from third experiment of [33]. Dataset #2 is a partition from [34]. The used pairs of ontologies and their true alignments are described in Tables 3 and 4. All ontologies are defined using OWL-DL[4] language in the RDF and XML format.

Dataset #1 is a set of ontologies about Bibliographic references from Benchmark test library. Ontology #101 is the reference ontology. Other ontologies (#102-#103, #301-#304) are compared with the reference ontology. Dataset has 7 ontologies and 6 true alignments: 3 alignments for training and 3 alignments for testing.

Dataset #2 consists several ontologies from Benchmark test library and all ontologies from Conference track of OAEI. Conference track contains 16 ontologies, which dealing with conference organization, and 21 true alignments. Dataset has 27 ontologies and 26 alignments: 8 alignments for training and 18 alignments for testing.

The pairs of entities from each pair of ontologies and their alignments are extracted. Dataset #1 has 14148 training samples (156 positive and 13992 negative samples) and 14940 testing samples (172 positive and 14768 negative samples) and Dataset #2 has 55348 training samples (284 positive and 55064 negative samples) and 114045 testing samples (253 positive and 113792 negative samples). A positive sample is a pair of entities which are matching, and a negative example is a pair of non-matching entities. Note that the datasets are very unbalanced.

5.2 Implementation

The approach was implemented using Python 3.5. This language is widely used for implementation of machine learning workflows and possesses a lot of useful program libraries.

[4] https://www.w3.org/TR/owl-features/

Table 2. Example part of true alignment 101-302

Entity from Ontology #101	Entity from Ontology #302
Collection	Book
TechReport	TechReport
Report	Publication
Reference	Resource
date	publishedOn

Ontologies are represented as RDF/OWL files. The *owlready2*[5] library was used for syntactic parsing of ontologies. Alignments are defined in RDF format. For parsing alignments, the *BeautifulSoup*[6] library was used. As implementation of logistic regression and random forest machine learning techniques *sklearn*[7] library is used. As a gradient boosting implementation the XGBoost[8] library is used. Dataset is formed as a dataframe of the *pandas*[9] library. To evaluate F-measure, the Alignment API[10] library was used. Computation experiments: training of machine learning models and the selection of their parameters were performed at the Hybrid high-performance computing cluster [51]. WordNet dictionary is taken from the *nltk*[11] library. Word2vec model was trained on GoogleNews[12] news. N-gram implementation is taken from the *ngram*[13] library. Similarity measures based on edit distance implementation is taken from the *editdistance*[14] library.

5.3 Experiments

The best parameters for the models were selected by the brute force method (a grid of values was created for each parameter): the models were trained on all combinations of parameters and the model with the best F-measure value using threshold 0.5 was selected.

For logistic regression, the following parameters were selected: inverse of regularization strength, weights of classes and norm used in the penalization. For random forest the number of trees in the forest, the maximum depth of the tree, the

[5] https://owlready2.readthedocs.io/en/latest/
[6] https://pypi.org/project/beautifulsoup4/
[7] https://scikit-learn.org/stable/
[8] https://xgboost.readthedocs.io
[9] https://pandas.pydata.org
[10] http://alignapi.gforge.inria.fr
[11] https://www.nltk.org
[12] https://github.com/mmihaltz/word2vec-GoogleNews-vectors
[13] https://pythonhosted.org/ngram/
[14] https://pypi.org/project/editdistance/

number of features to consider when looking for the best split and class weights were selected. For XGBoost, minimum sum of instance weight, minimum loss reduction requited to make a further partition, subsample ratio for training instances, subsample ratio of columns when constructing each tree and maximum depth of a tree were selected.

After training and searching for the best parameters, a threshold was selected with the highest F-measure value for each alignment. For each machine learning model, a grid of values for parameters was created manually. For numerical parameters, a grid of 3-5 values with different steps was created, i.e. for numbers of estimators in random forest: 10, 100, 200, 500, 1000. For parameters with options, all possible options were taken (2-4 options).

The values of F-measure for each alignment are presented in tables 3, 4 and 5. The best models for the Dataset #1 are logistic regression and random forest. Gradient boosting is a bit less accurate. However, the gradient boosting is the best model on average on Dataset #2. It is more accurate than logistic regression at 0.02 and than random forest at 0.01. The values of F-measure on Dataset #1 are comparable with the classical methods [37] [38] [54] [55] but lower than [33]. This may be associated with a specific set of training and test datasets, and it is also possible that the metrics that were not implemented in this work have an impact. In [33] the importance of each similarity measures is not described, but there is a hypothesis that the main contribution comes from similarity measures associated with comments to entities, and two structural measures from [55]. The study of this issue is future work. The values of F-measure on Dataset #2 are comparable with [34].

The computational complexity of the approach is $O(n_1n_2 + m_1m_2)$, where n_1, n_2 are the number of classes in the ontology O_1 and O_2, and m_1 and m_2 are the number of properties. The computation time and the used memory depending on the size of the ontology are showed on figures 1 and 2. The calculations were performed on MacBook Air 1.8 GHz 8GB RAM. The dependence of training and testing time on the ontology size is showed on figure 1. 20 points (evenly distributed between 10 and 1000) were used to build the figure. Training of random forest is longer than training of logistic regression and gradient boosting. The reason for the jumps on the figure is that the dataset is sampled randomly. Unfortunately, the used machine did not have enough capacity to calculate the time for training and testing gradient boosting with an ontology size of more than 600. The testing time of logistic regression and gradient boosting is much less than a random forest, therefore in the figure the graphs are close to zero. In general, there is a quadratic dependence. The dependence of memory usage on the ontology size is showed on figure 2. 20 points were also used to build the figure. Memory was measured using the memory profiler[15] package: the amount of used memory was measured when running the training and testing script. It is hard to understand why increasing the size of the ontology does not increase the amount of memory used, perhaps this is due to the internal work of the Python language. It is noticeable that the most memory is used by gradient boosting.

[15] https://pypi.org/project/memory-profiler/

Table 3. F-measure values for Dataset #1 with best thresholds

Alignment	Logistic Regression	Random Forest	XGBoost	Best results from [33]	FOAM [37]	DT [54]	OMAP [38]	OLA [55]
101-302	0.72	0.71	0.72	0.92	0.77	0.759	0.74	0.34
101-303	0.82	0.82	0.75	0.90	0.84	0.816	0.84	0.44
101-304	0.90	0.91	0.91	0.97	0.95	0.96	0.91	0.69
Average	0.81	0.81	0.79	**0.93**	0.85	0.845	0.83	0.49

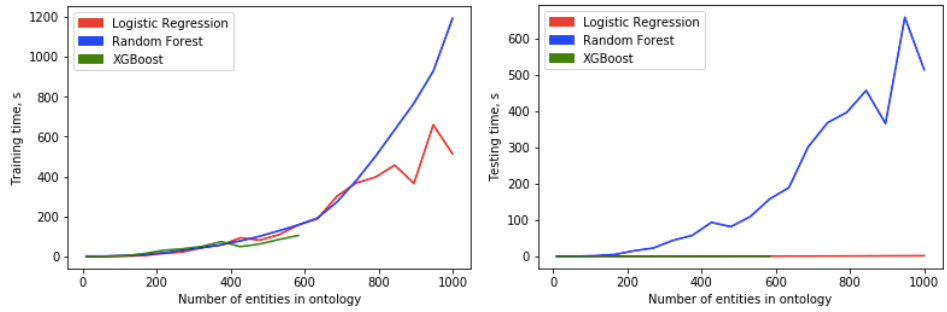

Fig. 1. Training and testing time of approach depending on the size of ontology.

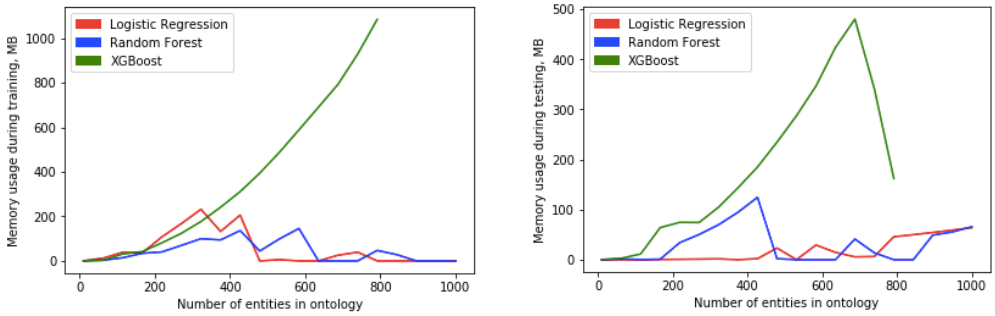

Fig. 2. Memory usage during training and testing.

Table 4. F-measure values for Dataset #2 with best thresholds

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
confof-sigkdd	0.72	0.72	0.72
Average	0.62	0.64	0.65

Table 5. Comparison of F-measure values

Alignment	Logistic Regression	Random Forest	XGBoost	Multi-layer perceptron [34]	REPTree [34]	M5 Rules [34]
Average	0.62	0.64	0.65	**0.67**	0.65	0.65

Conclusions and Future Work

We combined string-based, language-based, and structural-based similarity measures using three different machine learning models and apply them for ontology matching problem. The approach is implemented and evaluated using datasets selected from Ontology Alignment Evaluation Initiative (OAEI).

Due to the large number of similarity measures, there is hope that there is a potential for a more universal use of the approach. Universality refers to the applicability of the different subject areas. It is necessary to test the approach on ontologies with other subject areas. As a future work we would like to add similarity measures based on comments of entities, more structure-based similarity measures, such as a path length, a number of children, a number of properties of a class. It is also necessary to test the similarity measure from [53]. Neural network (multilayer perceptron) is planned to be used as a machine learning model. Evaluation issues to be resolved are checking the effectiveness of learning two different models separately for classes and properties and testing different strategies to resolve a problem of the strong imbalance of classes as well as strategies for significant reduce of a number of pairs of entities for matching.

Acknowledgments. The research is financially supported by Russian Foundation for Basic Research, projects 18-07-01434, 18-29-22096. The calculations were performed by Hybrid high-performance computing cluster of FRC CS RAS [51].

References

1. Gruber, T.: A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition - Special issue: Current issues in knowledge modeling, vol. 5, issue 2 (1993). doi: 10.1006/knac.1993.1008
2. Fensel, D.: Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. doi: 10.1007/978-3-662-09083-1
3. Gruninger, M., Lee, J.: Ontology Applications and Design - Introduction. In: Communications of the ACM (2002). doi: 10.1145/503124.503146
4. Euzenat, J., Shvaiko, P .: Ontology Matching. Springer-Verlag Berlin Heidelberg, Berlin (2007). doi: 10.1007/978-3-642-38721-0

5. Otero-Cerdeira, L., Rodríguez-Martínez, F., Gómez- Rodríguez, A.: Ontology Matching: A Literature Review. In: Expert Systems with Applications, vol. 42, issue 2, pp. 949-971 (2015). doi: 10.1016/j.eswa.2014.08.032
6. Shvaiko, P., Euzenat, J.: A Survey of Schema-Based Matching Approaches. In: Journal on Data Semantics IV , pp. 146-171 (2005). doi: 10.1007/11603412_5
7. Atencia, M., Euzenat, J., Pirro, G., Rousset, M.: Alignment-Based Trust for Resource Finding in Semantic P2P Networks. In: The Semantic Web – ISWC 2011: 10th International Semantic Web Conference, pp.51-66 (2011).doi: 10.1007/978 -3-642-25073-6_4
8. Arch-int, N., Arch-int, S.: Semantic Ontology Mapping for Interoperability of Learning Resource Systems using a rule-based reasoning approach. In: Expert Systems with Applications, vol. 40, issue 18, pp. 7428-7443 (2013). doi: https://doi.org/10.1016/j.eswa.2013.07.027
9. Mascardi, V., Ancona, D., Bordini, R., Ricci, A.: CooL-AgentSpeak: Enhancing AgentSpeak-DL Agents with Plan Exchange and Ontology Services. In: WI-IAT '11 Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, vol. 2, pp. 109-116 (2011). doi: 10.1109/WI-IAT.2011.255
10. Dong, X., Srivastava, D.: Big Data Integration. In: 2013 IEEE 29th International Conference on (2015). doi: 10.1109/ICDE.2013.6544914
11. E. Rahm, P. Bernstein.: A survey of approaches to automatic schema matching. In: The International Journal on Very Large Data Bases, vol. 10, issue 4, December 2001, pp. 334-350. doi: 10.1007/ s007780100057
12. Hlaing, S.: Ontology based schema matching and mapping approach for structured databases. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, pp. 853-859 (2009). doi: 10.1145/1655925.1656080
13. Nathalie, A.: Schema Matching Based on Attribute Values and Background Ontology. In: 12th AGILE International Conference on Geographic Information Science (2009).
14. Ichise, R.: Machine Learning Approach for Ontology Mapping using Multiple Concept Similarity Measures. In: Seventh IEEE/ACIS International Conference on Computer and Information Science (2008). doi: 10.1109/ICIS.2008.51
15. Mao, M., Peng, Y., Spring, M.: Neural Network based Constraint Satisfaction in Ontology Mapping. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, vol. 2, pp 1207-1212 (2008). http://www.dit.unitn.it/~p2p/RelatedWork/Matching/AAAI10-MaoM.pdf
16. Do, H., Melnik, S., Rahm, E.: Comparison of Schema Matching Evaluations. In: Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems, pp. 221-237 (2002). doi: 10.1007/3-540-36560-5_17
17. Do, H., Rahm, E.: COMA: a system for flexible combination of schema matching approaches. In: VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases, pp. 610-621 (2002). doi: 10.1016/B978-155860869-6/50060-3
18. L. Xu, D. Embley.: Automating Schema Mapping for Data Integration. (2003). http:// www.deg.byu.edu/papers/AutomatingSchemaMatching.journal.pdf
19. Lambrix, P., Tan, H.: SAMBO—A system for aligning and merging biomedical ontologies. In: Journal of Web Semantics, vol. 4, issue 3, pp. 196-206 (2006). doi: 10.1016/j.websem.2006.05.003
20. Ngo, D.: Enhancing Ontology Matching by Using Machine Learning, Graph Matching and Information Retrieval Techniques. In: University Montpellier II - Sciences et Techniques du Languedoc (2012). doi: 10.1.1.302.587

21. Bulygin, L.: Combining Lexical and Semantic Similarity Measures with Machine Learning Approach for Ontology and Schema Matching Problem. In: Selected Papers of the XX International Conference on Data Analytics and Management in Data Intensive Domains, pp. 245-249 (2018)
22. Gal, A., Modica, G., Jamil, H., Eyal, A.: Automatic Ontology Matching Using Application Semantics. In: AI Magazine - Special issue on semantic integration, vol. 26, issue 1, pp. 21-31 (2005).
23. Hariri, B., Sayyadi, H., Abolhassani, H.: Combining Ontology Alignment Metrics Using the Data Mining Techniques. In: Proceedings of the 2nd International Workshop on Contexts and Ontologies: Theory, Practice and Applications (2006).
24. Stoilos, G., Stamou, G., Kolias, S.: A String Metric for Ontology Alignment. In: The Semantic Web – ISWC 2005, pp. 624-637 (2005). doi: 10.1007/11574620_45
25. Cheatham, M., Hitzler, P.: String Similarity Metrics for Ontology Alignment. In: The Semantic Web – ISWC 2013, pp. 294-309 (2013). doi: 10.1007/978-3-642-41338-4_19
26. Saruladha, K., Aghila, G., Sathiya, B.: A Comparative Analysis of Ontology and Schema Matching Systems. In: International Journal of Computer Applications, vol. 34, issue 8, pp. 14-21 (2011).
27. Jean-Mary, R., Shironoshita, P., Kabuka, M.: Ontology Matching with Semantic Verification. In: Web Semant, vol. 7, issue 3, pp. 235-251 (2009). doi: 10.1016/j.websem.2009.04.001
28. Seddiqui, H., Aono, M.: Anchor-flood: Results for OAEI 2009. In: Proceedings of the 4th International Workshop on Ontology Matching collocated with the 8th International Semantic Web Conference (2009).
29. Kolyvakis, P., Kalousis, A., Kiritsis, D.: DeepAlignment: Unsupervised Ontology Matching with Refined Word Vectors. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (2018). doi: 10.18653/v1/N18-1072
30. Zhang, Y., Wang, X., Lai, S., He, S., Liu, K., Zhao, J., Lv, X.: Ontology Matching with Word Embeddings. In: Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data, pp 34-45 (2014). doi: 10.1007/978-3-319-12277-9_4
31. Nagy, M., Vargas-Vera, M., Motta, E.: DSSim-ontology mapping with uncertainty. In: 1st International Workshop on Ontology Matching (2006).
32. Nkisi-Orji, I., Wiratunga, N., Massie, S., Hui, K., Heaven, R.: Ontology alignment based on word embedding and random forest classification. In: Energy Transfer Processes in Polynuclear Lanthanide Complexes, pp.557-572 (2018). doi: 10.1007/978-3-030-10925-7_34
33. Nezhadi, A., Shadgar, B., Osareh, A.: Ontology Alignment Using Machine Learning Techniques. In: International Journal of Computer Science & Information Technology, vol. 3, pp. 139-150 (2011). doi: 10.5121/ijcsit.2011.3210
34. Alboukaey, N., Joukhadar, A.: Ontology Matching as Regression Problem. In: Journal of Digital Information Management, vol. 16, issue 1 (2018). http://dline.info/fpaper/jdim/v16i1/jdimv16i1_4.pdf
35. Cohen, W., Ravikumar, P., Fienberg, S.: A Comparison of String Metrics for Matching Names and Records.
36. Euzenat, J.: An API for ontology alignment. In: The Semantic Web - ISWC 2004: Third International Semantic Web Conference (2004). doi: 10.1007/978-3-540-30475-3_48

37. David, J., Guillet, F., Briand, H.: Association Rule Ontology Matching Approach. In: International Journal on Semantic Web and information systems, vol. 3, issue 2, pp. 27-49 (2007).
38. Straccia, U., Troncy, R.: oMAP: Combining Classifiers for Aligning Automatically OWL Ontologies. In: Web Information Systems Engineering, pp. 133-147 (2005). doi: 10.1007/11581062_11
39. Needleman, S., Wunsch, C.: A General Method Applicable to Search for Similarities in Amino Acid Sequence of 2 Proteins. In: Journal of Molecular Biology, vol. 48. issue 3, pp. 443-53 (1970). doi: 10.1016/0022-2836(70)90057-4
40. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. (2012). doi: 10.1016/C2011-0-06130-6
41. Appa Rao, G., Srinivas, G., Venkata Rao, K., Prasad Reddy, P.: A partial ratio and ratio based fuzzy-wuzzy procedure for characteristic mining of mathematical formulas from documents. (2018). doi: 10.21917/ijsc.2018.0242
42. Zobel, J., Dart, P.: Phonetic String Matching: Lessons from Information Retrieval. In: SIGIR '96 Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 166-172 (1996). doi: 10.1145/243199.243258
43. Tversky, A.: Features of Similarity. In: Psychological Review, vol. 84, issue 4, pp. 327-352 (1977). doi: 10.1037/0033-295X.84.4.327
44. Vijaymeena, M., Kavitha, K.: A Survey on Similarity Measures in Text Mining. (2016). doi: 10.5121/mlaij.2016.3103
45. Wu, Z., Palmer, M.: Verbs Semantics and Lexical Selection. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics (1994). doi: 10.3115/981732.981751
46. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: Proceedings of the International Conference on Learning Representations (2013).
47. Jurisch, M., Igler, B.: RDF2Vec-based Classification of Ontology Alignment Changes. (2018).
48. Breiman, L.: Random Forests. In: Machine Learning, vol. 45, issue 1, pp. 5-32. doi: 10.1023/A:1010933404324
49. Volkovs, M., Wei Yu, G., Poutanen, T.: Content-based Neighbor Models for Cold Start in Recommender Systems. In: Proceedings of the Recommender Systems Challenge (2017). doi: 10.1145/3124791.3124792
50. Sandulescu, V., Chiru, M.: Predicting the future relevance of research institutions - The winning solution of the KDD Cup 2016. (2016).
51. Federal Research Center Computer Science and Control of Russian Academy of Sciences. Available at: http://hhpcc.frccsc.ru (accessed 09/12/2018)
52. Nobarian, M., Derakhshi, M.: The Review of Fields Similarity Estimation Methods. In: International Journal of Machine Learning and Computing, vol. 2 (2012). doi: 10.7763/IJMLC.2012.V2.200
53. Znamenskij, S.: Stable assessment of the quality of similarity algorithms of character strings and their normalizations. In: Program systems: theory and applications, vol. 9, issue 39, pp. 561–578 (2018). doi: 10.25209/2079-3316 -2018-9-4-561-578
54. Eckert, K., Meilicke, C., Stuckenschmidt, H.: Improving Ontology Matching using Meta-level Learning. In: The Semantic Web: Research and Applications, pp. 158-172 (2009). doi: 10.1007/978-3-642-02121-3_15
55. Euzenat, J., Guégan, P., Valtchev, P.: OLA in the OAEI 2005 alignment contest. In: Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies (2005).