Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5648–5653
Marseille, 11–16 May 2020

Ontology Matching Using Convolutional Neural Networks

Alexandre Bento, Amal Zouaq, Michel Gagnon
École Polytechnique de Montréal
{alexandre.bento, amal.zouaq, michel.gagnon}@polymtl.ca

Abstract

In order to achieve interoperability of information in the context of the Semantic Web, it is necessary to find effective ways to align different ontologies. As the number of ontologies grows for a given domain, and as overlap between ontologies grows proportionally, it is becoming more and more crucial to develop accurate and reliable techniques to perform this task automatically. While traditional approaches to address this challenge are based on string metrics and structure analysis, in this paper we present a methodology to align ontologies automatically using machine learning techniques. Specifically, we use convolutional neural networks to perform string matching between class labels using character embeddings. We also rely on the set of superclasses to perform the best alignment. Our results show that we obtain state-of-the-art performance on ontologies from the Ontology Alignment Evaluation Initiative (OAEI). Our model also maintains good performance when tested on a different domain, which could lead to potential cross-domain applications.

Keywords: ontology matching, machine learning, convolutional neural networks

1. Introduction

An ontology is a set of structural rules designed to represent concepts in order to perform logic-based operations to retrieve or infer new information. As Semantic Web technologies are expanding and becoming more popular, the amount of data that needs to be represented grows proportionally, as does its complexity. Since web technologies are designed to be decentralised, redundancy inevitably occurs among knowledge bases. For this reason, finding methods to align equivalent data or their model is crucial. This is the goal of our research, which is described in this paper.

Traditional ontology matching methods are mainly based on string similarity and structure analysis. In this paper we propose another approach using machine learning techniques. The core of our method is based on the idea that semantic information can be retrieved automatically from text information within ontologies, and within their structure. To achieve this task, convolutional neural networks were used.

2. Previous Works

As indicated earlier, the usual way to align ontologies generally consists in standard string similarity measurement techniques, and the analysis of the ontologies' structure to identify equivalent classes and properties.

There are three main string analysis techniques: 1) string distance metrics (Levenshtein, Jaccard, string equality, etc.), 2) syntactic transformations (tokenization, lemmatization, stop word removal, etc.) and 3) semantic operations (finding synonyms, translating, categorization, etc.) (Cheatham and Hitzler, 2013). Although using only string metrics can lead to significant results for the ontology matching problem, it is crucial to choose the right metric to obtain the best alignment. Some advanced string metrics can also be defined and show powerful results, such as metrics that take both the similarities and the differences between two labels into account (Stoilos et al., 2005). Other powerful string metrics are based on n-gram decomposition, using the number of common n-grams to measure similarity (Sun et al., 2015). But in any case, a confidence value must be attributed to any of these metrics.

Another possible approach consists of analyzing similarities between data types of key properties for each studied concept (Granitzer et al., 2010); however, this approach cannot be used on its own, and must be coupled with a text analysis approach, such as string distance, as discussed above.

The structure of an ontology can also be used to determine similarity between concepts: if two classes have similar children or parents, they can be considered equivalent (Granitzer et al., 2010). The Ontology Alignment Evaluation Initiative (OAEI)[1] features some state-of-the-art ontology matching systems, among which the most effective are AML (Faria et al., 2013), the LogMap family (Jiménez-Ruiz and Grau, 2011) and POMAP (Laadhar et al., 2017). AML mainly uses three types of ontology matchers: a lexical matcher (that matches ontological classes using literal names), a mediating matcher (that uses an external ontology and the previously mentioned lexical matcher to establish "bridge" alignments between the two input ontologies) and a word matcher (that is derived from a Jaccard index between the words in the classes' names). LogMap uses a lexical matcher, that is enriched by a context analyzer: if class C_1 and class C_2 are mapped, then classes that are semantically close to C_1 are more likely to be mapped to classes that are semantically close to C_2. POMAP matches ontologies at two different levels: first at the element level (all classes of ontology O_1 are compared to all classes of ontology O_2, and the matches with the highest level of confidence are chosen); then at the structural level (if two classes are matched, then their siblings are more likely to be matched, as well as their subclasses).

Finally, some machine learning approaches have been implemented but are still uncommon in the field of ontology alignment. Some tried and tested algorithms such as K Nearest Neighbors (KNN), Support Vector Machine (SVM) and decision trees, which can outperform state-of-the-art matching tools (Nezhadi et al., 2011). Machine learning

[1] http://oaei.ontologymatching.org

techniques can also be used as a side tool for the alignment task, such as word embeddings to determine string similarity (Dhouib et al., 2019). Our approach goes a step further and frames the ontology alignment as a binary classification task that uses character embeddings of the class label and a set of superclasses' labels as an input.

3. Data

The data we used come from the Ontology Alignment Evaluation Initiative (OAEI). This data is organized in different tracks. For each track, several ontology matching tools (designed by various research teams) compete to obtain the best alignment performance. For training, we used data from the LargeBio track (Section 3.1); for testing, we used other independent tracks, described in Section 3.2.

3.1. Training - LargeBio Ontologies

LargeBio consists of three biomedical ontologies:

- Foundational Model of Anatomy (FMA)[2];
- SNOMED[3];
- National Cancer Institute Thesaurus (NCI)[4].

Table 1 shows the number of classes and properties for each ontology used for training. Table 2 indicates the number of reference alignments.

Ontology	# Classes	# Properties
FMA	78988	54
NCI	66724	190
SNOMED	223418	55

Table 1: The number of classes and properties (object properties + datatype properties) for each ontology used for training.

Ontologies	# Alignments
FMA-NCI	2686
FMA-SNOMED	6026
SNOMED-NCI	17210

Table 2: Reference number of aligned classes for training.

OAEI provides alignments between each of these ontologies. We used them to build a dataset that consists of class label pairs coming from the two ontologies to be aligned as positive examples. In order to obtain a balanced dataset (with the same number of positive and negative alignments), we randomly generated the same number of negative examples, matching classes that were not referenced as similar in the ontologies. Some classes in the ontologies did not have any label, and since this makes them unusable with our approach, they were excluded from the dataset. We also noticed that classes referenced as equivalent often had identical labels (after converting labels to lower case), which biased the training process. Indeed, given their predominance in the dataset of positive alignments, they biased the model to classify all pairs without similar labels as negative alignments. This rendered these classes as irrelevant since their alignments were trivial. Hence these examples were excluded from the dataset as well. With these constraints, we found 18105 positive examples within LargeBio ontologies with a complete dataset containing 36210 examples (after adding negative examples).

Finally, this dataset was divided into training, validation and test sets. We used 80% of the original dataset for training, 10% for validation and 10% for testing. The test set was used to choose the model with the best predictive capabilities, but it was not used to evaluate our approach and to compare it to the state of the art. In fact, we considered that data coming from the same ontologies for both training and testing would not reflect the challenges of real alignment tasks, and could induce a favorable bias for our model. Therefore, the final comparison and testing was performed on other completely independent ontologies as described in the next section.

3.2. Testing - Other OAEI Ontologies

In order to evaluate our method and compare it to previous works, we used the other ontologies provided by the Ontology Alignment Evaluation Initiative (OAEI). OAEI evaluation is divided in different tracks, based on the domain and complexity of ontologies. We chose to evaluate our method on four OAEI tracks:

- Anatomy
- Phenotype and Disease
- BioDiv
- Conference

Table 3 indicates the number of classes and properties for each ontology used for testing. Table 4 shows the number of reference class alignments.

Track	Ontology	# Classes	# Properties
Anatomy	Mouse	2744	3
	Human	3304	2
Phenotype and Disease	DOID	17122	42
	HP	32655	130
	MP	31721	130
	ORDO	13504	24
BioDiv	ENVO	8970	165
	FLOPO	29088	156
	PTO	1505	1
	SWEET	4538	3359
Conference	All	837	724

Table 3: The number of classes and properties (object properties + datatype properties for each ontology used for testing.

[2]http://sig.biostr.washington.edu/projects/fm/
[3]http://www.ihtsdo.org/index.php?id=545
[4]http://ncit.nci.nih.gov/

Track	Ontologies	# Alignments
Anatomy	Human-Mouse	1516
Phenotype and Disease	DOID-ORDO	1237
	HP-MP	696
BioDiv	ENVO-SWEET	421
	FLOPO-PTO	153
Conference	All	249

Table 4: Reference number of aligned classes for testing.

To make sure that our method was tested in the same way as other competitors from the OAEI competition, we integrated our tool with the SEALS platform[5], as required by the OAEI guidelines.

4. Description of the Alignment Method

In this section, we explain how our alignment system is built, how input data is processed and how the final alignment is generated.

4.1. Preprocessing

The preprocessing stage consists of two different steps:

- transforming ontological data into numerical vectors that a neural network model can use as input;
- pre-selecting alignments that are too trivial to be submitted to the neural network model.

4.1.1. Input Data Transformation

For the task we aim to achieve, input data consist of two separate ontologies that must be aligned. As this cannot be directly fed to a machine-learning model, it must be transformed into a set of numerical vectors that a model can use as input. The final alignment must be a set of comparisons between each class of both ontologies to be aligned.
First, the structure of each ontology is extracted using a dedicated tool (we used Apache Jena for this application). Extracting the structure enables us to make a list of all classes in the ontology. For each class that has a label, we also extract the labels of its superclasses.
Our convolutional neural network takes as input a pair of class labels and their superclasses (we observed experimentally that using subclasses does not make predictions more accurate ; results are detailed in section 5) and returns whether they are similar or not. As neural network models require a fixed input vector length, some limits have to be set in order to respect this constraint. First, for each class label, we set a maximum length of 150 characters (blank spaces are appended to shorter labels; longer labels are cropped). We observed that this value led to a minimal prediction error.
As classes have a variable number of superclasses, we must set a limit for this parameter as well. We chose to use five superclasses for each class. For classes with fewer superclasses, we used padding with blank spaces; for classes with more superclasses, we only took the first five levels in the structure of their superclasses.

[5]Semantic Evaluation At Large Scale (SEALS) platform: http://www.seals-project.eu/

A neural network model cannot take strings of characters as direct input. Hence characters must be converted to numbers. A simple character encoding could have been used (such as the ASCII code), but we chose to use character embeddings[6]. These embeddings provide a representation vector for each possible character. The embedding we used is of dimension 300. Each provided value is normalized between 0 and 1.
To limit noise within the input data, we also lowercased each class label, and replaced underscores with spaces.

4.1.2. Pre-selecting Trivial Examples

In order to prevent combinatorial explosion when comparing classes from both ontologies (each ontology can have thousands of classes, leading to millions of possible combinations to consider), we needed to prune obvious examples before passing input data to the neural network model. Two cases are worth noting:

- Trivial positive examples: across two different ontologies, some classes have identical names, as discussed in section 3.1. When this happens, these classes are automatically considered equivalent.
- Trivial negative examples: since only a small proportion of the candidate pairs to align contain equivalent classes, many completely different classes would have to be evaluated if no preselection was carried out. To limit this problem, we chose to apply a distance-based preselection, using two criteria: Levenshtein distance and the length of each class label. If the Levenshtein distance between the labels of two classes is large, then these classes are considered different. We also noticed that short labels lead to many classification errors. To solve this problem, we chose to weight the Levenshtein distance with a factor that is inversely proportional to the length of the labels: the shorter the labels, the greater the final distance. Let a and b be the labels of two classes that need to be compared. Let L_a and L_b their respective length. We compute the following index i:

$$i = \frac{levenshtein(a, b)}{L_a * L_b} \quad (1)$$

  We then define a threshold th. If $i > th$, we classify the two classes as different. We found experimentally that $th = 1.5$ leads to reliable results.

4.2. Alignment using Machine Learning Models

Once input data has been preprocessed, it can be used to train a neural network model. We chose to use convolutional neural networks (henceforth CNN) in order to capture relevant structural information inside text data.

4.2.1. Neural network model

The model we used is composed of two main modules:

- CNN layers;
- Multilayer perceptrons (MLP).

[6]https://github.com/minimaxir/char-embeddings

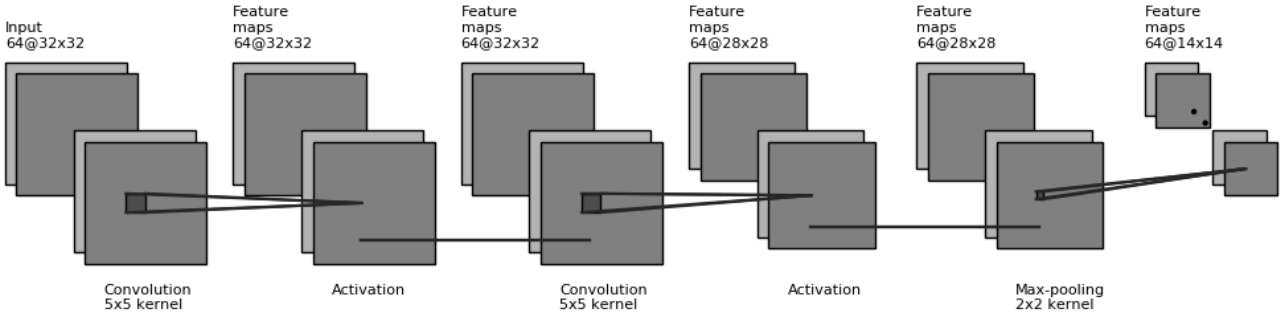

Figure 1: An example of a super-convolution layer that takes 64 features of size 32x32 as input.

Class 1 Label	Class 1 Superclass 1	Class 1 Subclass 1	Class 2 Label	Class 2 Superclass 1	Class 2 Subclass 1	Alignment
Central nervous System Part	Nervous System Part	Brain Part	Lymph Node Part	Immune System Part	Germinal Center	0
Vein	Venous Blood Vessel	Adrenal Vein	Vein	Blood Vessel	Femoral Vein	1

Figure 2: A negative and a positive example using classes' labels, one superclass and one subclass. These examples come from the Anatomy track.

We used several layers of CNN, with a different filter size for each. This allows us to capture patterns at different scales, which are obtained by successive applications of average pooling (which is equivalent to a x2 zoom each time: for example, if the network analyzes data inside a window of five characters, this window changes to ten characters).
We define a super-convolution layer with the following configuration:

- a convolution layer with a kernel size of five characters;
- an activation layer (MLP with Rectified Linear Unit activation (ReLU));
- a second convolution layer with a kernel size of five characters;
- a second activation layer (MLP with ReLU);
- an average pooling layer of size two characters.

A super-convolution layer is shown in Figure 1.
The CNN part of the model is composed of eight super-convolution layers (which allows us to process all scales with a division factor of two in between each layer).
Super-convolution layers are then followed by MLP layers. We observed experimentally that the depth of the network had a more significant impact on performance than the size of the layers. Thus the model we used was composed of 10 MLP layers, of size 500 neurons each. We used ReLU to activate each layer.

Finally, the last layer of the model leads to a single neuron (with sigmoid activation) in order to produce a binary classifier that indicates whether the two classes are aligned or not.

4.2.2. Training strategy

In a real-life situation, when aligning two ontologies, the number of positive and negative examples that have to be classified is unbalanced. For example, for two ontologies with 2,000 classes each, we need to process four millions examples to complete the alignment, when only a few hundred classes are equivalent in these ontologies. If this issue is not considered during training (i.e., if the training dataset has the same number of positive and negative examples), the trained model can generate a large number of false positive alignments (which remains marginal when the test dataset is balanced as well, but in a real-life situation, this is no longer the case).
To address this problem, a proper training strategy must be adopted. We chose to grant more weight to negative examples during training, as the false positive problem comes from them. We found experimentally that a weight of 40 (versus a weight of 1 for positive examples) works best. Below this value, too many false positives still appear. Above it, the model fails to identify some positive examples.
As the model needs to be symmetrical (i.e., if class a is equivalent to class b, then b is equivalent to a), the training dataset was duplicated and the class positions were swapped: instead of building an input vector with data from class a followed by that of class b, we built it with class b followed by class a. In this case, the order of the classes does not impact classification.

Adam optimizer (Kingma and Ba, 2014) was used to manage the training phase. Considering the high number of training examples, the learning rate was set to a low value to achieve a stable training phase (i.e. the training loss decreases smoothly, instead of having local highs and lows). We found that a value of 10^{-5} gives reliable results. We also used learning rate decay, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

In order to make the training process automatic, we used early stopping to prevent overfitting (i.e. to prevent the neural network model from learning too much from its training data, leading to poor performance on test data, hence when validation loss starts to increase, the training process is stopped). As training is not perfectly stable, it is necessary to use a patience value of two (i.e. when the validation loss goes up during two consecutive iterations, the training process is stopped) in order to avoid local minima. Validation loss was used as the early stopping criterion.

To get the best alignment performance, it is good practice to train several models and select the one that demonstrates the best prediction capability on a test dataset. As discussed in section 3.1, a dedicated test set was created for this purpose. We trained five models and chose the one with the lowest error on test data.

5. Evaluation and Results

Our approach was evaluated on OAEI ontologies as described in Section 3.2. Three of these ontologies are composed of biomedical data (Anatomy, Phenotype and Disease, and BioDiv, which is the same domain as the training data), and the last one (Conference) is based on conference organisation data, which represents a different domain.

We first evaluated which structure was the most appropriate for our classification problem: we tried considering class labels only, then we tried various numbers of superclasses and subclasses at different depths. An example is provided in Figure 3. Figure 2 shows positive and negative examples before their transformation into vectors of numbers that are fed to the neural network model.

Detailed results are presented in Table 5. These results are computed on our test dataset (which is different from the final test set, described in Section 3.2).

# Superclasses / # Subclasses	0	1	5	10
0	0.907	0.921	0.952	0.953
1	0.894	0.916	0.941	0.942
5	0.887	0.913	0.928	0.932
10	0.859	0.863	0.878	0.881

Table 5: F1-score for each tested structure. Each column represents the number of levels of superclasses, and each row represents the number of subclasses. The cell without any superclass and any subclass means that only class labels were used for classification.

As shown in Table 5, adding subclasses only leads to lower performance, hence they were not used in the final neural network model. The best results were obtained with the classes labels and ten levels of superclasses. However, with five levels of superclasses, the model has very similar performance, hence we chose this value, since it made the computation time a lot lower.

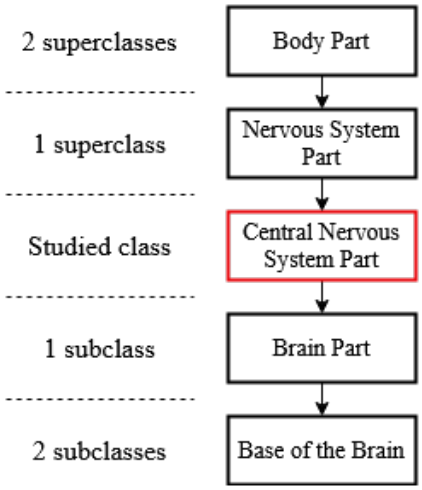

Figure 3: An example of the class structure used for classification. Here we are considering the Central Nervous System Part, and we show two of its superclasses and two subclasses. This example comes from the Human ontology, in the Anatomy track.

We integrated our tool with the SEALS platform so that we could evaluate it in the same conditions as other participants of the OAEI workshop; we then compared the results we obtained with the 2018 OAEI results[7]. Results are presented in Table 6.

Task	F1	Best F1	Position
Anatomy	0.881	0.943	5/16
HP/MP	0.850	0.855	3/9
DOID-ORDO	0.831	0.848	4/10
FLOPO-PTO	0.829	0.86	2/8
Conference	0.75	0.77	3/15

Table 6: Final results on OAEI tracks. HP/MP and DOID-ORDO are tasks from the Phenotype and Disease track. FLOPO-PTO is a task from BioDiv. Best F1 corresponds to the F1-score of the best participant for each task. Position represents the position of our system in the competition based on the best F1.

Task	Precision	Best precision	Position
Anatomy	0.871	0.998	15/16
HP-MP	0.882	0.997	4/9
DOID-ORDO	0.870	0.996	7/10
FLOPO-PTO	0.872	0.987	3/8
Conference	0.80	0.88	10/15

Table 7: Final results on OAEI tracks. Best precision corresponds to the maximum precision obtained for each track. Position represents the position of our system in the competition based on the best precision.

[7] 2018 OAEI results: http://oaei.ontologymatching.org/2018/results/

Task	Recall	Best recall	Position
Anatomy	0.890	0.936	3/16
HP-MP	0.819	0.855	4/9
DOID-ORDO	0.795	0.870	4/10
FLOPO-PTO	0.79	0.84	2/8
Conference	0.70	0.76	3/15

Table 8: Final results on OAEI tracks. Best recall corresponds to the maximum recall obtained for each track. Position represents the position of our system in the competition based on the best recall.

As shown in Table 6, our approach leads to similar results as state-of-the-art competitors. Table 7 and Table 8 show that our system still lacks precision (which is related to the problems discussed in Section 4.2.2), but does a good job at retrieving non-trivial alignements (since the recall value places our system among the best). Also, as the results for the Conference track show, our system achieves good performance for a non-biomedical ontology. This means that our CNN model was successful in aligning classes from different domains even if it was trained on biomedical ontologies, which is a valuable result. Additional experiments on various domains are left for future work. The preprocessing part of the methodology presented in this paper could also be replaced by more advanced techniques (possibly machine learning as well). This could help solve the false-positive problem presented in Section 4.2.2 by producing more accurate results. Finally, we plan to include datatype and object properties as inputs to our neural network.

6. Conclusion and future works

In this paper, we showed that a machine learning approach to the ontology matching problem using convolutional neural networks leads to state-of-the-art results. As there is no domain-dependant variable in our methodology, it can be applied as-is to any domain. We also showed that a model trained on biomedical data gives results that are consistent when applied to other domains; however this needs to be confirmed on more datasets.

Our approach could be improved using other types of neural networks, or a combination of different models: recurrent networks, especially Long Short-Term Memory networks (LSTM), are very appropriate for text data, as they can be used to analyse sequences, and may improve performance for ontology matching. Similarly transformers might lead to better results and are a research direction we would like to explore, together with different inputs to the network.

7. Acknowledgement

This research was supported by Canada First Research Excellence Fund program.

8. References

Cheatham, M. and Hitzler, P. (2013). String similarity metrics for ontology alignment. In *International semantic web conference*, pages 294–309. Springer.

Dhouib, M. T., Zucker, C. F., and Tettamanzi, A. G. (2019). An ontology alignment approach combining word embedding and the radius measure. In *International Conference on Semantic Systems*, pages 191–197. Springer.

Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I. F., and Couto, F. M. (2013). The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 527–541. Springer.

Granitzer, M., Sabol, V., Onn, K. W., Lukose, D., and Tochtermann, K. (2010). Ontology alignment—a survey with focus on visually supported semi-automatic techniques. *Future Internet*, 2(3):238–258.

Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Laadhar, A., Ghozzi, F., Megdiche, I., Ravat, F., Teste, O., and Gargouri, F. (2017). Pomap: An effective pairwise ontology matching system. In *KEOD*, pages 161–168.

Nezhadi, A. H., Shadgar, B., and Osareh, A. (2011). Ontology alignment using machine learning techniques. *International Journal of Computer Science & Information Technology*, 3(2):139.

Stoilos, G., Stamou, G., and Kollias, S. (2005). A string metric for ontology alignment. In *International Semantic Web Conference*, pages 624–637. Springer.

Sun, Y., Ma, L., and Wang, S. (2015). A comparative evaluation of string similarity metrics for ontology alignment. *Journal of Information &Computational Science*, 12(3):957–964.