Ontology Matching Using Convolutional Neural Networks

Alexandre Bento, Amal Zouaq, Michel Gagnon

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

Ontology matching is a key problem to solve in the context of Semantic Web, in order to achieve interoperability of information. As the number of ontologies grows for a given domain, and as overlapping between ontologies follows the same path, developing accurate and reliable techniques to operate this task automatically is becoming more and more crucial. While traditional approaches to solve this problem are based on string metrics and structure analysis, in this paper we present a methodology to align ontologies automatically, using machine learning techniques. We use convolutional neural networks to analyse structure in text data for OAEI ontologies. We obtain state-of-the-art performance, as well as 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 the same way, and so does its complexity. Since web technologies are designed to be decentralised, redundancy among knowledge bases occurs inevitably. For this reason, finding methods to align equivalent data or their model is becoming crucial. This problem is addressed 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 complete this task, convolutional neural networks were used.

2. Previous Works

As explained earlier, the usual way to align ontologies consists in standard string similarity measurement techniques, and the analysis of the ontologies' structure in order to identify equivalent classes and properties (Hooi et al., 2014).

There are three main string analysis techniques: 1) string distances (Levenshtein, Jaccard, string equality, etc.),2) syntactic transformations (tokenisation, 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 when aligning ontologies, it is crucial to choose the right metric for the ontologies being studied.

Some more advanced string metrics can be defined and show powerful results, such as metrics that take into account both the similarities and the differences between two labels (Stoilos et al., 2005). But in any case, a confidence value must be attributed to any of these metrics.

Another possible approach consists in analysing 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.

Also, the structure of an ontology can be used to determine similarity between concepts: if two classes have similar children or parents, they should be considered as equivalent (Granitzer et al., 2010).

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 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 methodology goes a step further and uses embeddings as an input to make classification more accurate.

3. Data

The data we used come from the Ontology Alignment Evaluation Initiative (OAEI)[1]. This data is organised in different tracks. For training, we used data from the LargeBio track; 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].

OAEI provides alignments between each of these ontologies. We used them to build a dataset. Some classes in these ontologies do not have any label, and since this makes them

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

unusable with our approach, they were excluded from the dataset.
We also noticed that classes referenced as equivalent often had the exact same label (after passing labels to lower case), which biased the training process, and were completely irrelevant since these alignments are trivial. Hence these examples were excluded from the dataset as well.
With these constraints, we found 18105 references within LargeBio ontologies. In order to make a balanced dataset (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. This leads to a complete dataset containing 36210 examples.
Finally, this dataset was divided into training, validation and test sets (this test set is used to choose the model with best predictive capabilities, but was not used to evaluate our methodology and to compare it to others, as this would induce a bias when choosing the best model (final test data is not supposed to be used for any kind of training). The test was performed on other ontologies - see next section). We used 80% of the original dataset for training, 10% for validation and 10% for testing.

3.2. Testing - Other OAEI Tracks

In order to evaluate our method and compare it to previous works, we used the 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

To make sure that our method was tested the same way as other competitors in the OAEI workshop, we wrapped our tool with the SEALS platform, which is used by the OAEI organization to evaluate participants.

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 step consists in two different actions:

- 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 model.

4.1.1. Input Data Transformation

For the task we aim to achieve, input data consist in two separate ontologies that must be aligned. As this cannot directly be 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). This allows us to make a list of each class of the ontology. For each class that has a label, we also gather the label of its superclasses, allowing us to use the structure of the ontology as well.
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) 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 (shorter labels are filled with blank spaces at the end; 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 5 superclasses for each class. For classes with fewer superclasses, we use padding with blank spaces; for classes with more superclasses, we only take the first five levels in the structure of its superclasses.
A neural network model cannot take strings of characters as direct input. These must be converted into numerical vectors. A simple character encoding could have been used (such as the ASCII code), but we chose to add more semantic information during this preprocessing step, and hence used a character embedding[5]. This embedding provides a representation vector for each possible character. The embedding we used is of dimension 300. Each provided value is normalized between 0 and 1, which is ideal for feeding a neural network model.
To limit noise within the input data, we also lowercased each class label, as well as replaced underscores with blank 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 evaluate), a preselection must be applied before passing input data to the neural network model. Two cases must be studied:

- trivial positive examples: across two different ontologies, some classes can be identified with the exact same name, as discussed in section 3.1. When this happens, these classes are automatically considered as equivalent.
- trivial negative examples: since only a small proportion of the candidate pairs to alignment contain equiv-

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

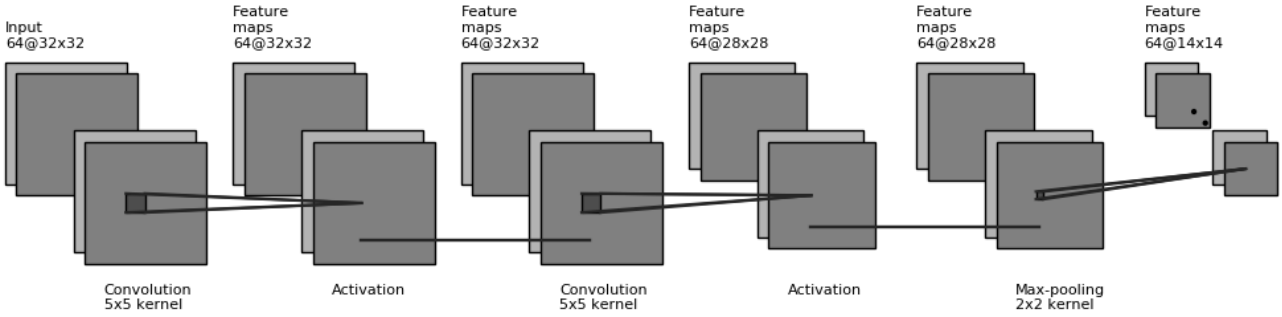

Figure 1: A super-convolution layer which takes 64 features of size 32x32 as input.

alent classes, this implies that many completely different classes would have to be evaluated if no preselection was involved. 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 too large, then these classes are considered as different. We also noticed that short labels lead to many mistakes during classification, hence we need to take this problem into consideration. To solve it, 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 higher 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. Then we compute the following index i:

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

We then define a threshold th. If $i > th$, we directly classify the two classes as different.

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 (noted as CNN in the following paragraphs) 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 sections:

- CNN layers;
- a multilayer perceptron (noted as MLP).

We used several layers of CNN, with a different filter size for each. This allows capturing patterns at different scales, which are obtained by successive applications of average pooling (which is the equivalent of a x2 zoom each time). We define a super-convolution layer with the following:

- a convolution layer with a kernel size of 5;
- an activation layer (we use ReLU);
- a second convolution layer with a kernel size of 5;
- a second activation layer (ReLU);
- an average pooling layer of size 2.

A diagram of a super-convolution layer is shown in Figure 1.
The CNN part of the model is composed of 8 super-convolution layers (which allows processing all scales with a division factor 2 in between each layer).
CNN 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 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 make a binary classifier.

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 are heavily unbalanced. Indeed, for two ontologies with 2000 classes each, we need to process 4000000 different examples in order to complete the alignment, when only a few hundred classes are equivalent in between 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).
In order to face 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 correctly.
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.
Adam optimizer was used to manage training. Considering the high number of training examples, the learning rate must be set to a low value in order to get stable training.

We found that a value of 10^{-5} gives decent 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. As training is not perfectly stable, it is necessary to use a patience value of 2 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 pick the one with best prediction capability on a test dataset. As discussed in section 3.1, a dedicated test set was created for this purpose. We trained 5 models and chose the one with the lowest error on test data.

5. Evaluation and Results

This methodology was evaluated on OAEI tracks described in Section 3.2. Three of them are composed of biomedical data (Anatomy, Phenotype and Disease, and BioDiv; same domain as the training data), and the last one (Conference) is different and is based on conference organisation data.
We wrapped our tool with the SEALS platform so that we could evaluate it in the same conditions as other participants for the OAEI workshop; we then compared the results we obtained with the 2018 OAEI results[6]. Results are presented in Table 1.

Task	Prec.	Recall	F1	Best F1	Pos.
Anatomy	0.871	0.890	0.881	0.943	5/16
HP/MP	0.882	0.819	0.850	0.855	3/9
Doid/Ordo	0.870	0.795	0.831	0.848	4/10
Flopo/Pto	0.872	0.790	0.829	0.86	2/8
Conference	0.80	0.70	0.75	0.77	3/15

Table 1: 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-measure of the best participant for each task. Pos represents the position of our system in the competition based on the best F1.

As shown in Table 1, our methodology leads to similar results as state-of-the-art competitors. Also, as the results for the Conference track show, our system produces relevant performance for non-biomedical ontologies. This means that our network was successful in aligning classes from different domains, which is a valuable result.

6. Conclusion and future works

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 to any domain, with no necessary adaptation. We also showed that a model trained on biomedical data gives consistent results on other domains; however this needs to be confirmed on more datasets.
This approach could be improved using other types of neural networks, or a combination of different models: recurrent networks, especially LSTM, are very appropriate for text data, as they allow analysing 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.
The preprocessing part of the methodology presented in this article could also be replaced with more advanced techniques (possibly machine learning as well). This could help solve the false-positive problem presented in Section 4.2.2 with more accurate results.

7. 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.

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.

Hooi, Y. K., Hassan, M. F., and Shariff, A. M. (2014). A survey on ontology mapping techniques. In *Advances in Computer Science and its Applications*, pages 829–836. Springer.

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.

[6] 2018 OAEI results: http://oaei.ontologymatching.org/2018/results/