Ontology Clustering with OWL2Vec*

Ashley Ritchie[1], Jiaoyan Chen[2], Leyla Jael Castro[3],
Dietrich Rebholz-Schuhmann[3], Ernesto Jiménez-Ruiz[1]

[1] City, University of London, London, United Kingdom
[2] University of Oxford, United Kingdom
[3] ZB MED Information Centre for Life Sciences, Cologne, Germany

Abstract. In this work we present an exploratory study to apply OWL2Vec* to drive the clustering of ontology entities (i.e., ontology clustering). OWL2Vec* is a state-of-the-art system that creates embeddings, capturing the semantics of both entities and tokens that appear in the ontology.

1 Introduction

Vector embeddings are used to process large datasets as the vectorized space makes it easier to find insights and combine data from multiple sources. Although initially used for text, embeddings are also used to process graph-based structures such as knowledge graphs (KGs) largely composed of facts. Quite a few KG embedding techniques have been proposed and applied to entity clustering [1], but few have considered ontologies which are more complicated. In this work, we present an exploratory study using OWL2Vec* to drive the computation of ontology entity clusters. OWL2Vec* [2] is a state-of-the-art system that creates embeddings for both the entities and the lexical information that appears in an ontology. We have explored 327 embedding configurations by varying the hyperparameters of OWL2Vec*. These embeddings were then analyzed by measuring the distances between each entity's embeddings as well as clustering the embeddings using the *k*-means algorithm.

The motivation comes from OntoClue [3], which aims to aid researchers with better information retrieval (IR) by finding relevant articles, although not necessarily similar. In a similar vein to topic modelling where topics are found and assigned to documents, OntoClue plans to use clusters emerging from ontology embeddings to later assign topics to scholarly publications. To do so, embedding-based clusters will be combined with text-mining and named entity recognition to find related articles across multiple clusters. In this work we focus the experiments on the Gene Ontology (GO) [4], an ontology that is structurally and lexically rich. Our main goal was understanding how different embedding configurations and cluster sizes would affect the resulting clustering. Our next step, outside the scope of this paper, will be using such clusters to assign topics to a corpus of documents and therefore improve results from IR.

2 Ontology Embedding

Machine learning models typically assume dense numeric representation, which is separate from how ontologies are expressed [5]. Therefore, ontology embedding is used as a way to condense the rich semantic and structural information of an ontology into a low-dimensional vector space that can be used downstream by machine learning algorithms. A common method for ontology embedding is through the use of word embedding models such as Word2Vec, which enables the computing of continuous vector representations from a large corpus of text [6].

OWL2Vec* [2] is an embedding method built on top of RDF2Vec [7] and Word2Vec supporting multiple configurations and hyperparameters. OWL2Vec* has been tested with medium (e.g. an events ontology) and small (e.g. pizza ontology) size ontologies and has shown good performance; however, the performance is expected to improve with larger vocabularies. The algorithm accepts an ontology as input and produces embeddings as output. It first generates random walks over the ontology to extract structural, lexical, and semantic information in order to create a corpus of IRI and word sequences. Another option is to replace an entity in the sequence by its Weisfeiler Lehman sub-tree kernel which measures the entity's neighborhood sub-graph in tree shape. This corpus is then fed to the word embedding model to create IRI and word (token) vector representations, V_{iri} and V_{word}.

OWL2Vec* has four document settings, the first of which is the Structure Document, D_s. It is composed of IRI sequences captured from the walks as well as the axioms, or relationships, between the classes within the ontology. The second document configuration is the Lexical Document, $D_{s,\,l}$, which replaces the IRIs of the structural document with the entity labels. Additionally, it includes sentences of lexical entity annotations.

The last two document settings are both Combined Documents, whereby one element of each sentence remains as an IRI and the others elements are converted to lexical labels. The first strategy, $D_{s,\,l,\,rc}$, is based on random selection. A randomly selected element will remain as an IRI, while the other elements are converted to their lexical labels. The other strategy, $D_{s,\,l,\,tc}$, is to traverse each IRI in the sentence, replacing other IRIs with their lexical labels. If there are n IRIs in the sentence, this will create n new sentences. Table 1 shows examples of document sentences produced from random walks of depth two starting at *obo:GO_0007611* in Figure 1.

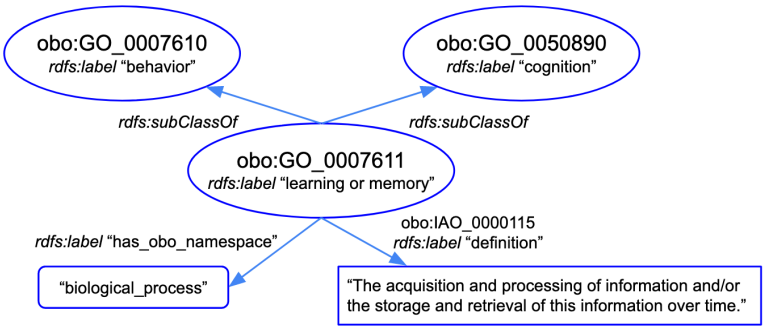


Fig. 1. A Snapshot of Gene Ontology.

Table 1. Sentence examples extracted from the Gene Ontology snapshot in Figure 1 with random walk depth of two.

Document	Sentence
Structural (D_s)	(*obo:GO_0007611, rdfs:subClassOf, obo:GO_0007610*)
Literal (D_{s_l})	("learning", "or", "memory", "subclassof", "behavior")
Combined, random ($D_{s,\ l,\ rc}$)	("learning", "or", "memory", "subclassof", *obo:GO_0007610)*
Combined, traverse ($D_{s,\ l,\ tc}$)	("learning", "or", "memory", "subclassof", *obo:GO_0007610),* (*obo:GO_0007610*, "subclassof", *"behavior")*

3 Methods

Our input data is the Gene Ontology (GO) [4] retrieved on 9th October 2020 from the Open Biological and Biomedical Ontologies (OBO) Foundry website. GO is one of the most successful ontology development projects to date, broadly used in Life Sciences. GO has 44,272 classes split into three branches: biological process (28,922 classes), molecular function (11,157 classes), cellular component (4,193 classes). We use OWL2Vec* to create embeddings which are later clustered by *k*-means. We then use the Silhouette score and Bag of Words to evaluate the output.

3.1 OWL2Vec* and Clustering

OWL2Vec* requires some hyperparameters and settings to run. First we have to define the *Document and Embedding Settings***.** OWL2Vec* has ten different combinations of document and embedding settings. These combinations come from the four document options (Structural, Lexical, Combined - Random, Combined - Traversal) and three output vector options (IRI, Word, or a concatenation of IRI and Word vectors). The annotations that are used as lexical information (i.e., *Annotation Type*) can also be configured. This project tested whether limiting the type of entity annotations to *rdfs:label* and *oboInOwl:hasExactSynonym* would reduce noise compared with including all annotations. Therefore the settings tested are 'All Annotations' and 'Limited Annotations'. While the *Walker Type* determines the walking method, the *Walk Depth* controls the length of the walk. Two ontology walking methods were considered, random walks and random walks with Weisfeiler Lehman sub-tree kernel enabled, and walking depths of 2, 4, and 6 were tested. Finally, the *Embedding Size* is defined, having an impact on the output size. Sizes of 50, 100, and 200 were tested. For each of the embeddings produced, *k*-means clustering with three (trying to reflect the three GO branches) and 400 clusters were implemented (so we would get small clusters but still with meaningful information). This was to assess the ability to capture the structure of the three ontology branches as well as the granularity of the produced clusters, respectively.

3.2 Evaluation Metrics of Embedding Distances and Clusters

Silhouette Score. The primary evaluation metric used for this project is the average Silhouette score across all classes (see Eq. 1). It measures both cohesion and separation of the clusters by calculating the mean intra-cluster distance, *a(i)*,

differencing it with the mean inter-cluster distance of the next closest cluster, *b(i),* and dividing by the greater of the two. Here *i* and *j* correspond to two items from the same or a different cluster, and N is the number of total items.

$$S = \frac{1}{N}\sum_{i=1}^{N}\frac{b(i)-a(i)}{\max\{a(i),b(i)\}} \; where, \quad \begin{matrix} a(i) = \frac{1}{|\mathrm{C}_i|-1}\sum_{j\in\mathrm{C}_i, i\neq j} d(i,j) \\ b(i) = \min_{k\neq i}\frac{1}{|\mathrm{C}_k|}\sum_{j\in\mathrm{C}_k} d(i,j) \end{matrix} \tag{1}$$

Silhouette scores can range from $-1 \leq s(i) \leq 1$. A positive Silhouette score closer to one is desirable and indicates better cohesion and separation. A negative score indicates clusters are dispersed and lack clear inter-cluster separation. Due to the nature of ontologies and their interconnectedness, a high degree of separation is unlikely and may result in lower silhouette scores. In this study, average Silhouette scores are used merely as a measure to compare embedding settings.

Bag of Words. In order to assess the granularity achieved with the clusters, we look at 'semantic cohesion', namely how lexically or semantically similar items within the same cluster are. In this assessment, the lexical information of each entity's label, branch, parent, and synonyms are extracted from the annotations. The Bag of Words then calculates the most frequent 1-gram and 2-gram sequence of tokens within the cluster.

4 Results

Capturing Ontology Structure. To assess how well the embeddings capture the structure of GO, distances between entity embeddings were calculated. The ontology is naturally divided into three branches (biological process, molecular function, and cellular component) and it is expected that classes within the same branch are closer than classes in different branches. As such, we would expect classes within the same branch to be cohesive and slightly separated from other branches, which should yield a positive average Silhouette score.

Table 2 shows the average Silhouette scores based on ontology branches. While the best performing settings are those belonging to $D_s + V_{iri}$ with a walk depth of 2, there are other interesting results from the table. Most notably, when lexical information was included in the document, embeddings created from word vectors improved the Silhouette score while IRI vectors hindered the Silhouette score. The table also shows that the *Combined Document* settings performed worse than the *Lexical Document* settings. This suggests that using both IRIs and lexical information in the document created noise instead of improving correlation for the case of the GO. The table also shows that limiting annotations did not have substantial effect on performance.

Table 2. Average Silhouette score based on ontology branches. The subscripts *s*, *l*, *rc* and *tc* denote structure, literal, random-based combination, traversal-based combination documents, respectively; the subscripts *iri* and *word* denote IRI and word vectors, respectively. Color scale goes from dark green (highest score) to dark red (lowest score).

Document Embedding Settings	Score Average	Annotation Type		Walker Type		Walk Depth			Embedding Size		
		A	L	R	WL	2	4	6	50	100	200
$D_s + V_{iri}$	0.0569	0.0569		0.0552	0.0585	0.1667	0.0116	-0.0077	0.0586	0.0563	0.0557
$D_{s,l} + V_{iri}$	0.0173	0.0173	0.0173	0.0221	0.0126	0.0643	-0.0079	-0.0044	0.0211	0.0167	0.0141
$D_{s,l} + V_{word}$	0.1008	0.1024	0.0992	0.1022	0.0994	0.0918	0.1038	0.1068	0.1048	0.0984	0.0991
$D_{s,l} + V_{iri,word}$	0.0646	0.0652	0.0640	0.0721	0.0570	0.0912	0.0563	0.0462	0.0664	0.0641	0.0632
$D_{s,l,rc} + V_{iri}$	0.0003	0.0008	-0.0003	-0.0046	0.0051	0.0055	-0.0051	0.0004	0.0026	0.0002	-0.0020
$D_{s,l,rc} + V_{word}$	0.0994	0.1010	0.0979	0.1000	0.0989	0.0915	0.1021	0.1046	0.1050	0.0961	0.0971
$D_{s,l,rc} + V_{iri,word}$	0.0571	0.0579	0.0564	0.0609	0.0534	0.0714	0.0528	0.0472	0.0596	0.0563	0.0555
$D_{s,l,tc} + V_{iri}$	0.0048	0.0050	0.0046	-0.0004	0.0115	0.0072	0.0030	0.0039	0.0049	0.0056	0.0039
$D_{s,l,tc} + V_{word}$	0.0959	0.0974	0.0944	0.0957	0.0962	0.0920	0.1005	0.0950	0.1013	0.0929	0.0939
$D_{s,l,tc} + V_{iri,word}$	0.0565	0.0571	0.0560	0.0572	0.0555	0.0653	0.0531	0.0459	0.0593	0.0560	0.0542
Overall Average	0.0553	0.0562	0.0545	0.0560	0.0546	0.0747	0.0470	0.0427	0.0585	0.0543	0.0532

The settings that produced the highest branch-based Silhouette score were with D_s+V_{iri} (structural document with IRI vector) using the Weisfeiler Lehman sub-tree kernel with walking depth of two and producing embedding size of 100. As such, this combination of OWL2Vec* settings were able to capture the structure of the three ontology branches. Figure 2a shows that the embeddings have good intra-branch cohesion and slight inter-branch separation.

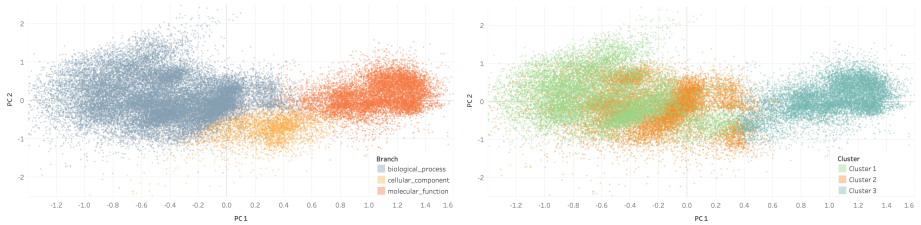
This set of embeddings was also used to inform a *k*-means clustering task. Ideally, *k*-means would create clusters very similar to the ontology branches. Figures 2a and 2b compare the plot of embeddings based on the ontology's branches and the clusters assigned by *k*-means. Overall, *k*-means produced fairly even clusters (Cluster 1: 16,401 classes, Cluster 2: 15,118 classes, and Cluster 3: 12,753 classes) resulting in a Silhouette score of 0.183. The composition of classes in each of the clusters are:

Cluster 1: biological process 15,217; cellular component 1,137; molecular function 47.

Cluster 2: biological process 13,691; cellular component 1,322; molecular function 105.

Cluster 3: biological process 14; cellular component 1,734; molecular function 11,005.

As is visible in the cluster composition and in Figure 2b, *k*-means does not separate classes within the biological function and cellular component branches, capturing the relationships that exist across branches. However, it was able to cluster 98.64% of molecular function classes into the same cluster.



(a) Ontology branches. (b) Three *k*-means clusters.

Fig. 2. Plot of embedding principal components

Cluster Granularity with *K*-Means. A *k*-means clustering was implemented for 400 clusters on each of the embeddings to see if it was possible to produce clusters of around 110 classes each while achieving a fine leve of granularity. It was assessed that this number of clusters would likely yield meaningful clusters, neither too narrowly defined nor overly broad.

Silhouette score was used to identify the embedding settings that produced clusters with good cohesion and separation. One of the interesting outcomes from Table 3 is that increasing the walking depth improved performance of embeddings with IRI vectors, V_{iri}, and decreased performance of embeddings with word vectors, V_{word}. This is an opposite effect from the results in Table 2. One possible explanation for this could be attributed to the increased challenge of keeping compact and meaningful clusters as the number of clusters increases. It is likely for lexical similarity to decrease as walk depth increases, thus shallower walks may produce tighter clusters when based on lexical information. In contrast, increasing the walk depth for structural information may help the method to create neighborhoods around common parent classes, improving granularity as walk depth increases. Further analysis is needed to verify this hypothesis and will be carried out by testing intermediary cluster sizes in conjunction with OntoClue requirements.

Table 3. Average Silhouette score based on 400 *k*-means clusters with various settings. The subscripts and color scale have the same meaning as Table 2.

Document Embedding Settings	Score Average	Annotation Type		Walker Type		Walk Depth			Embedding Size		
		A	L	R	WL	2	4	6	50	100	200
$D_s + V_{iri}$	0.1079	0.1079		0.1154	0.1004	0.0909	0.1107	0.1220	0.1159	0.1065	0.1012
$D_{s,l} + V_{iri}$	0.1022	0.1017	0.1026	0.1082	0.0961	0.0690	0.1183	0.1192	0.1159	0.1010	0.0895
$D_{s,l} + V_{word}$	0.1403	0.1390	0.1417	0.1467	0.1339	0.1729	0.1322	0.1159	0.1470	0.1397	0.1343
$D_{s,l} + V_{iri,\,word}$	0.1197	0.1189	0.1206	0.1268	0.1127	0.1458	0.1119	0.1015	0.1283	0.1190	0.1120
$D_{s,l,rc} + V_{iri}$	0.0915	0.0910	0.0921	0.0886	0.0944	0.0670	0.1003	0.1072	0.1053	0.0898	0.0795
$D_{s,l,rc} + V_{word}$	0.1250	0.1256	0.1245	0.1315	0.1186	0.1479	0.1189	0.1083	0.1341	0.1240	0.1170
$D_{s,l,rc} + V_{iri,\,word}$	0.1044	0.1046	0.1041	0.1064	0.1023	0.1162	0.1001	0.0967	0.1136	0.1037	0.0958
$D_{s,l,tc} + V_{iri}$	0.0985	0.0990	0.0981	0.1013	0.0950	0.0932	0.0982	0.1069	0.1106	0.0981	0.0880
$D_{s,l,tc} + V_{word}$	0.1168	0.1165	0.1172	0.1198	0.1127	0.1346	0.1062	0.1047	0.1269	0.1157	0.1080
$D_{s,l,tc} + V_{iri,\,word}$	0.1034	0.1030	0.1039	0.1048	0.1014	0.1174	0.0939	0.0946	0.1119	0.1028	0.0957
Overall Average	0.1112	0.1109	0.1115	0.1150	0.1070	0.1155	0.1091	0.1086	0.1212	0.1102	0.1022

Overall, embeddings created from word vectors produced a higher Silhouette score. More specifically, the *Lexical Document* setting produced the ten highest Silhouette scores. The highest Silhouette score was 0.1842 and was achieved with the document and vector settings of $D_{s,\,l} + V_{word}$ with limited annotations, using a random walker of depth 2 and the embedding size of 50. There were on average 110 classes per cluster with the median at 75 classes per cluster. This shows that the majority of clusters are smaller than the target of 110 with some clusters being much larger.

Looking into the makeup of clusters with Bag of Words, it is possible to see these embedding settings capture neighborhoods well. For example, Cluster 66, highlighted in Table 4, is composed of 108 classes, largely related to ‘transaminase activity’ and ‘aminotransferase activity’. Of the 108 classes within the cluster, 106 are descendants of GO term *GO_0008483* 'transaminase activity’ within the molecular function branch. Comparing this with the ontology, *GO_0008483* has 112 descendants, meaning this cluster was able to capture 94.6% of the entity’s descendants. It is

interesting to note that two items within the cluster are not descendants of *GO_0008483* yet have similar lexical information in their labels. *GO_0005969* and *GO_0032144* have labels 'serine-pyruvate aminotransferase complex' and '4-aminobutyrate transaminase complex'. Both belong to the cellular component branch but are capable of transaminase activity, highlighting the ability for the algorithms to capture cross-branch relationships to a fine degree.

Table 4. Cluster 66 Bag of Words - Top 5 (1,2)n-grams with counts

Label n-gram	count	SubClassOf n-gram	count
'activity'	105	'activity'	107
'transaminase'	69	'transaminase'	90
'transaminase activity'	66	'transaminase activity'	90
'aminotransferase'	39	'aminotransferase'	16
'aminotransferase activity'	37	'aminotransferase activity'	16

5 Conclusion and Future Work

In this paper, embeddings for the Gene Ontology were produced using various settings of OWL2Vec*. When assessing the ability to capture the ontology structure, the structure document with shallow walks and Weisfeiler Lehman sub-tree kernel improved the Silhouette score. These settings showed good cohesion and separation between the three ontology branches. *K*-means was also able to capture relationships that exist across branches. Additionally, a *k*-means clustering with 400 clusters was implemented to assess cluster granularity. Using the lexical document with word vectors and shallow walks improved the Silhouette score. Bag of Words was used to assess cluster makeup and lexical similarity. It would be interesting to improve upon this method by naming clusters based on a common ancestor or by using the label of the entity closest to the cluster centroid. In the near future we plan to apply such clusters to assign topics (i.e., a set of ontology classes) to a corpus of documents to improve the results of an IR task. An exploration of incremental cluster sizes may also be useful to assess cluster makeup for the downstream IR task and gain further insight into what the results show. In addition, OWL2Vec* embeddings can be applied in diverse applications, for example, to predict entity relationships (as reported in [2]), or in an ontology alignment task [8].

References

1. Goyal, P., & Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems, 151, 78-94 (2018).
2. Chen J, Hu P, Jimenez-Ruiz E, Holter OM, Antonyrajah D, Horrocks I. OWL2Vec*: Embedding of OWL Ontologies. Machine Learning, Springer, 2021. https://github.com/KRR-Oxford/OWL2Vec-Star
3. Castro, LJ, Rebholz-Schuhmann D. 2021 OntoClue project. Retrieved on 14.03.2021. Available from: https://zbmed-semtec.github.io/ontoclue/
4. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong, Nucleic Acids Research, Volume 47, 2019

5. Hogan A, Blomqvist E, Cochez M, d'Amato C, de Melo G, Gutierrez C, et al. Knowledge Graphs. arXiv:200302320. 2020. Available from: http://arxiv.org/abs/2003.02320
6. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. ICLR 2013
7. Ristoski P, Rosati J, Di Noia T, De Leone R, Paulheim H: RDF2Vec: RDF graph embeddings and their applications. Semantic Web 10(4): 721-752 (2019). We rely on the pyRDF2Vec implementation: https://github.com/IBCNServices/pyRDF2Vec
8. Chen, J., Jimenez-Ruiz, E., Horrocks, I., Antonyrajah, D., Hadian, A., Lee, J.: Augmenting OntologyAlignment by Semantic Embedding and Distant Supervision. In: European Semantic Web Conference, ESWC 2021