Ontology alignment based on word embedding and random forest classification

Ikechukwu Nkisi-Orji[1][0000−0001−9734−9978], Nirmalie Wiratunga[1], Stewart Massie[1], Kit-Ying Hui[1], and Rachel Heaven[2]

[1] Robert Gordon University, Aberdeen, UK
{i.o.nkisi-orji✉, n.wiratunga, s.massie, k.hui}@rgu.ac.uk
[2] British Geological Survey, Nottingham, UK
reh@bgs.ac.uk

Abstract. Ontology alignment is crucial for integrating heterogeneous data sources and forms an important component of the semantic web. Accordingly, several ontology alignment techniques have been proposed and used for discovering correspondences between the concepts (or entities) of different ontologies. Most alignment techniques depend on string-based similarities which are unable to handle the vocabulary mismatch problem. Also, determining which similarity measures to use and how to effectively combine them in alignment systems are challenges that have persisted in this area. In this work, we introduce a random forest classifier approach for ontology alignment which relies on word embedding for determining a variety of semantic similarity features between concepts. Specifically, we combine string-based and semantic similarity measures to form feature vectors that are used by the classifier model to determine when concepts align. By harnessing background knowledge and relying on minimal information from the ontologies, our approach can handle knowledge-light ontological resources. It also eliminates the need for learning the aggregation weights of a composition of similarity measures. Experiments using Ontology Alignment Evaluation Initiative (OAEI) dataset and real-world ontologies highlight the utility of our approach and show that it can outperform state-of-the-art alignment systems.

Keywords: Ontology alignment · Word embedding · Machine classification · Semantic web.

1 Introduction

Ontology alignment or matching deals with the discovery of correspondences between the entities of different ontologies. This has been the subject of various research works over the years with several techniques adopted from methods for integrating heterogeneous databases. The utility of ontologies are enhanced through alignment and the reduced semantic gap enables applications requiring cross-ontology reasoning or data exchange. Interest in ontology align-

ment is reflected through the Ontology Alignment Evaluation Initiative (OAEI)[3] which provides a platform to assess and compare systems for automated or semi-automated alignment. Also, the Linking Open Data community project[4] which aims to align ontologies on a Web scale currently have hundreds of datasets from different contributors in multiple domains such as DBpedia, WordNet, GeoNames, and MeSH.

The ontology alignment process is challenging, especially when the ontologies are of heterogeneous origins leading to inherent differences between them. Ontologies can vary vastly in levels of formalisation and vocabulary use even when they are of similar domain. The predominant methods for alignment use a composition of multiple string-based similarity metrics on textual features of entities [2]. Semantic matching is essential for discovering correspondences by meaning when the vocabularies of source and target ontologies differ. However, there is a shortage of semantic matching techniques [18, 19]. Lexical databases such as WordNet have been leveraged for semantic matching but they lack sufficient coverage and this becomes apparent when dealing with domain-specific terminology. Accordingly, word embedding approaches which are effective at capturing language semantics have been proposed for semantic matching in ontology alignment [21, 22]. Semantic matching approaches do not always outperform string-based similarity and effectively combining both strategies in alignment systems remain a challenge [18].

In this work, we introduce a novel matching system that integrates string-based similarity and semantic similarity features using word embedding to build a machine learning model, a random forest classifier. Alignment is completed in two stages by first selecting a set of candidate alignments using basic matching techniques. Afterwards, a machine classifier determines which entity pairs of the candidate alignments are true alignments. The classifier uses feature vectors that are generated from a variety of direct and indirect similarity indicators. Our main contributions are the incorporation of word embedding for semantic match discovery in the alignment process and the introduction of novel features for a machine classifier for alignment. The alignment system relies on minimal information from the ontologies making it suitable for aligning knowledge-light ontological resources. Although it requires training a classifier model, our approach eliminates the need to learn aggregation weights for multiple similarity measures. We evaluate the alignment system on benchmark datasets from OAEI and dataset from EuroVoc (EU's multilingual thesaurus)[5].

The remainder of this paper is organised as follows: section 2 reviews relevant works in literature; section 3 presents our ontology alignment approach; section 4 is an experimental evaluation which compares our approach to alternative approaches; and section 5 concludes with an outline for future work.

[3] `http://oaei.ontologymatching.org/`
[4] `http://linkeddata.org/`
[5] European Union, 2018, `http://eurovoc.europa.eu/`

2 Related work

Ontology alignment establishes semantic links between the entities of different ontologies which is a solution to the semantic heterogeneity problem [6, 19]. Alignment reduces the semantic gap between overlapping representations of a domain and trends show increasing interest in this area [18]. Establishing correspondences between the entities of different ontologies generally follows pairwise comparisons (direct or indirect) to identify best matches. Techniques for matching entities can be element-level or structure-level [18]. Element-level matching uses intrinsic features of entities such as natural language labels and definitions [10]. Instead of exact string matching, edit distance approaches such as Levenshtein and Jaro–Winkler distances are commonly used for fuzzy matching to account for spelling variations and word inflection. Structure-level matching considers the ontological neighbourhood of entities in order to determine similarity. Even when entities share little element-level features, correspondences can be discovered by similarity of structures such as having similar ancestors or descendants [17].

String similarity methods differ and an individual approach cannot be always relied on for effective alignment [2]. Accordingly, most alignment systems use a composition of multiple similarity metrics (basic matchers) which are aggregated sequentially or in parallel [3, 10] or form features for a machine learner [5, 16]. This leads to a categorisation of research in ontology alignment as matching techniques or matching systems [18]. Matching techniques deal with measures of similarity and strategies that determine the extent to which the concepts of different ontologies relate while matching systems use one or more matching techniques to align ontologies. The choice of matching techniques and determining composition weights for multiple similarity metrics have been subject of several research works [7, 13]. As ontologies differ widely, it is not unusual to encounter alignment systems which work well for some alignment tasks and perform weakly on others.

String comparison is less effective for alignment when the vocabulary of ontologies differ. As a result, external knowledge resources such as WordNet and Wikipedia have been used to estimate semantic similarities [8, 9, 12]. Use of external resources requires anchoring entities being compared to the external resources which is then used for inferencing. By matching by meaning, semantic matching can discover alignments which are omitted by string-based similarity approaches. Yet, semantic matching is rarely used because effective integration of string-based similarity and semantic similarity remains a challenge [18, 19]. Recent experiments show that matching using word embedding vectors outperforms use of lexical databases such as WordNet for semantic matching [22]. Word2vec models are popular implementations of word embedding using shallow neural network architecture to embed words in a dense continuous vector space based on their linguistic contexts in a corpus [14]. Word embedding preserves several linguistic regularities and similarity between word vectors have been shown to correlate well with human judgements. The use of word embedding is also promising for cross-lingual alignment by jointly embedding ontologies

in a vector space [21]. An even more effective use of word embedding for ontology alignment is a hybrid similarity approach that incorporates string similarity using edit distance [22]. To the best of our knowledge, no other system has extended use of word embedding for alignment beyond a hybrid similarity of edit distance and vector similarity. We extend the hybrid similarity approach by introducing other similarity features which are used by a random forest classifier to align ontologies.

3 Classifier-based ontology alignment

Our approach is based on generating a machine classifier model using a hybrid of element-level string-based features, semantic similarity features, and context-based structure-level similarity features. A high-level overview of the alignment process is presented in Figure 1 and the rest of this section describes the process in detail. The alignment process starts with the selection of candidate alignments using a variety of basic matching techniques. A feature vector is then generated for each candidate alignment which is passed to the machine classifier. The classifier determines whether the concept pair is accepted as a correspondence or discarded.

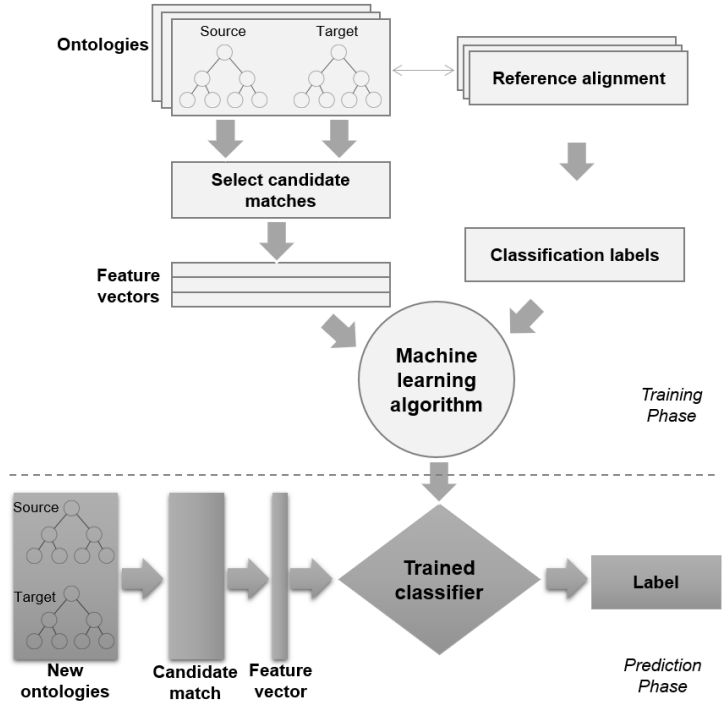

Fig. 1: Overview of ontology alignment process using machine learning.

Notations, scope and assumptions An ontology, θ specifies a set of concepts (or entities), $\theta = \{c_1, ..., c_n\}$. A concept $c \in \theta$ represents the semantic definition of a meaningful entity in a domain. Although some ontologies also specify data properties and object properties, we use this minimal specification to include knowledge-light ontological resources such as thesauri and controlled vocabularies. Let $labels(c)$ return the set of textual labels of a concept including alternative names (or synonyms), $labels(\theta)$ return an ontology's document collection which is all labels of all concepts of θ, and $tok(l)$ return all words from a concept's label, $l \in labels(c)$. To illustrate with Figure 2, concept #3945 has two labels making $label(\#3945) = \{$"petroleum industry","oil industry"$\}$, $tok($"petroleum industry"$) = \{$"petroleum","industry"$\}$, and $label(\theta)$ returns eight labels. We assume that the ontologies being aligned specify some form of subsumption relations between concepts such as "is-a" or "broader-than" relations. This allows for the identification of a concept's semantic context and depth on the ontology structure. The subsumption relation between two concepts c_i and c_j is represented as $c_i \prec c_j$ specifying that c_i is a broader concept of c_j (e.g. $\#2673 \prec \#3945$ in Figure 2).

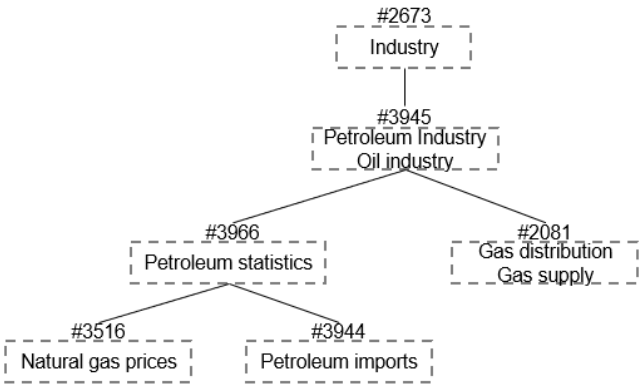

Fig. 2: Example of concepts' hierarchy with textual labels.

The output of the alignment process between the source ontology θ and target ontology θ' is the alignment, A which is a set of correspondences between semantically equivalent concepts of both ontologies. Each correspondence $a \in A$ is a 4-tuple, $a :< c, c', \equiv, s >$ where $c \in \theta$, $c' \in \theta'$, $\equiv$ indicates equivalent relation type between c and c', and s is the confidence of alignment correspondence in $[0.0, 1.0]$ interval. Confidence is either 1 (correspondence) or 0 (no correspondence) for crisp alignment.

3.1 Identification of alignment candidates

The objective for selecting candidate alignments is to avoid including concept pairs that have little or no chance of being aligned in subsequent machine classi-

fication stage. A pair of concepts being compared become candidate alignments if their similarity exceeds the threshold for any of four similarity measures. Accordingly, similarity thresholds for candidate selection are kept low enough to maximise recall but not very low to select the entire similarity matrix. This avoids having to generate features for concept pairs with very low similarities and also leads to a better class balance for training a classifier. We also use a $Max1$ selection approach for each similarity measure such that if multiple concepts in the target ontology exceed the selection threshold, we only choose the pair(s) with highest similarity value. This is commonly used to enforce a one-to-one correspondence in alignment [19]. A variety of ways in which concepts can be similar were considered in selecting similarity measures for identifying candidate alignments as follows.

1. Hybrid similarity (*hybrid*): combines word embedding and edit distance,
2. Vector space model (*vsm*): cosine similarity of term vectors using term frequency – inverse document frequency (tf-idf) scheme,
3. ISUB similarity (*isub*): string similarity metric designed for ontology alignment, and
4. Similarity of semantic context (*context*): indirect similarity between concepts by comparing their neighbours on the ontology structure.

Hybrid similarity Hybrid similarity combines the use of word embedding and edit distance measures [22]. After discarding words which occurred less than 10 times, we embedded a November 2016 database dump of Wikipedia English language articles in vector space of 300 dimensions using Word2vec's continuous skip-gram architecture. The word embedding model was generated using an open-source deep learning library[6]. There is an abundance of literature and software tools on word embedding therefore, we will not discuss details of implementation further. We also used the Google New Corpus model[7] as an alternative word embedding model for comparison. The edit distance component of our hybrid similarity is based on Levenshtein distance. In contrast to [22], a threshold is imposed on the edit distance component. This is because below certain thresholds, similarity by sharing similar characters is no more than a coincidence. Similarity between terms is based on the approach for measuring sentence similarity [11] as shown in equation 1.

$$hybrid(c, c') = \max_{\{l \in labels(c), l' \in labels(c')\}} \left\{ \frac{1}{maxLen(l, l')} \cdot \sum_{w \in l} \sum_{w' \in l'} max(emb(w, w'), lev(w, w')) \right\} \tag{1}$$

[6] https://deeplearning4j.org/word2vec.html

[7] https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz

$maxLen(l, l') = max(|tok(l)|, |tok(l')|)$ is length of the longer label, $emb(w, w')$ is the cosine similarity between the embedding vectors of w and w', and $lev(w, w')$ is normalised Levenshtein similarity. First, Levenshtein distance is normalised to $[0.0, 1.0]$ interval by dividing by the length of the longer string. Similarity is then determined as $1 -$ normalised distance and is only considered when up to 0.8. In other words, equation 1 compares each word from one label with every word in the other label and selects the maximum similarity of either word embedding or edit distance. The sum of best pairwise similarities is then divided by the length of the longer label. For example, in comparing "oil industry" and "petroleum industry", the best similarities are $emb(\text{oil}, \text{petroleum}) = 0.65$ using the Google model and $lev(\text{industry}, \text{industry}) = 1.0$ giving an overall similarity $\frac{1}{2}(0.65+1) = 0.825$. The most similar labels are used when concepts have multiple labels. A low hybrid similarity threshold of 0.4 was chosen in our experiments to maximise recall.

Vector space model The second similarity measure is based on the vector space model using cosine similarity of tf-idf weights. Each ontology forms a collection, D ($D = labels(\theta)$). The tf-idf weight of each word, w in a document, d (a concept's label) is determined as shown in equation 2.

$$\text{tf-idf}(w) = f_{w,d} \cdot \log \frac{|D|}{n_w} \tag{2}$$

$f_{w,d}$ is the frequency of w in d, and n_w is the number of documents in which w appears. Since multiple documents can belong to a concept, VSM similarity is determined as the maximum similarity of the documents of a concept pair as shown in equation 3.

$$\text{vsm}(c, c') = \max_{\{d \in c, d' \in c'\}} \{cosSim(d, d')\} \tag{3}$$

$cosSim(d, d')$ is the cosine similarity between documents d and d' using their tf-idf weight vectors. By weighing terms such that frequently occurring words in an ontology contribute less to similarity, we discover alignments that will otherwise be missed as observed in [17]. Similarity threshold was set at 0.7 which is low enough for good recall.

ISUB similarity The third similarity approach is a string similarity metric which was specifically designed for the purpose of aligning ontologies [20]. The similarity between two strings is determined by the extent of their common substrings which is offset by their differences (equation 4).

$$\text{isub}(c, c') = \max_{\{l \in labels(c), l' \in labels(c')\}} \{Comm(l, l') - Diff(l, l') + winkler(l, l')\} \tag{4}$$

$Comm(l, l')$ is a function of common substrings, $Diff(l, l')$ is a function of the difference between the strings, and $winkler(l, l')$ is for improving the results. We used an implementation of ISUB similarity in the Alignment API [4].

Context similarity When the lexical forms of textual features of a pair of concepts are different, comparing their ontological neighbourhoods can discover correspondences which are missed by direct comparisons. Accordingly, we indirectly measure the similarity of concepts by comparing their semantic contexts. If the parents and children of the concepts being compared are similar, the pair are included in the set of candidate alignments. Let the immediate parent concepts of c be $P(c)$ and its immediate child concepts be $C(c)$, we implemented context similarity as in equation 5.

$$\text{context}(c, c') = max \left\{ \frac{1}{2} \cdot (hybrid(c_p, c'_p) + hybrid(c_c, c'_c)) \right\} \tag{5}$$

max indicates that only the most similar parent and child concepts are used to determine context similarity with $c_p \prec c | c_p \in P(c)$, $c \prec c_c | c_c \in C(c)$, $c'_p \prec c' | c'_p \in P(c')$ and $c' \prec c'_c | c'_c \in C(c')$. We set selection threshold at half of hybrid similarity threshold since equation 5 is an average.

3.2 Features for alignment classification

In the second stage, feature vectors are generated for candidate alignments which are used by a machine classifier to determine whether they are actual alignments. We introduce various novel features in addition to similarity metrics that are commonly used for basic matching. Features are grouped into three categories (selection, direct similarity, and context features) and summarised in Table 1. Recall that each alignment candidate comprises of a concept from the source ontology ($c \in \theta$) and the most similar concept it in the target ontology ($c' \in \theta'$). We also note the next most similar concept to c in the target ontology ($c'' \in \theta'$) for the purpose of determining features which are related to similarity offsets.

Selection features These features are determined during the selection of candidates alignments to reflect the best similarity value (sim), the method of similarity used ($matchType$), and similarity offset to the next most similar concept in target ontology ($simOffset$). $matchType$ is a nominal attribute used to indicate the similarity method that was used to select a candidate alignment. sim is determined as $max(hybrid(c, c'), vsm(c, c'), isub(c, c'), context(c, c'))$. $simOffset$ is determined as $sim(c, c') - sim(c, c'')$ and this captures the distinctiveness of a candidate alignment. High sim and $simOffset$ values are expected to be good indicators of actual alignments. Finally, we also include each of the similarity methods for selecting candidate alignments as a separate feature.

Direct similarity features This category comprises other similarity metrics that directly compare textual labels of concepts. These include five commonly used string-based similarity measures – Levenshtein (lev), Fuzzy Score[8] ($fuzzy$),

[8] `https://commons.apache.org/proper/commons-text/apidocs/org/apache/commons/text/similarity/FuzzyScore.html`

Table 1: Feature vectors for alignment

Feature category	Feature	Description
Selection	$matchType$	Similarity method to select alignment
	sim	$max(hybrid, vsm, isub, context)$
	$simOffset$	Offset to the next best sim
	$hybrid$	Combines lev and emb
	vsm	Similarity based on vector space model
	$isub$	String similarity for ontology alignment
	$context$	$hybrid$ of semantic contexts
Direct similarity	lev	Similarity based on Levenshtein distance
	$fuzzy$	Fuzzy string score gives bonus points as characters in matched substrings increases.
	lcs	Similarity based on Longest Common Subsequence
	$dice$	Similarity based on Sorensen-Dice coefficient
	$mongeElkan$	Monge-Elkan similarity measure
	$prefixOverlap$	Prefix overlap divided by length of shorter string
	$suffixOverlap$	Suffix overlap divided by length of shorter string
	emb	Similarity of word embedding vectors
Context	$parentsOverlap$	Hybrid similarity of parent concepts
	$childrenOverlap$	Hybrid similarity of child concepts
	$contextOverlap$	Hybrid similarity of all context words
	$contextOverlapOffset$	Offset to next best $contextOverlap$
	$hasParents$	Indicates if both, one, or none of the concepts have parent nodes
	$hasChildren$	Indicates if both, one, or none of the concepts have child nodes
	$depthDiff$	Difference in relative depths of concepts

Longest Common Subsequence (lcs), Sorensen-Dice ($dice$), and Monge-Elkan ($mongeElkan$) [2, 15]. These were chosen to provide a variation of string similarities as each algorithm differs in its approach. Also, we include features for similarity based on word embedding alone (emb) and maximum prefix overlap ($prefixOverlap$) and suffix overlap ($suffixOverlap$) of concept labels. Prefix overlap and suffix overlap are the number of contiguous characters shared at the beginning and ending of strings respectively and are normalised by diving by the length of the shorter string. Most of the string similarity measures were implemented using publicly available API[9].

[9] `http://github.com/tdebatty/java-string-similarity`

Context features Features in this category are determined by the placement of concepts on the ontology structure. These include *parentsOverlap* and *childrenOverlap* which are *hybrid* similarities of parent and child concepts (of candidate nodes) respectively. We also introduce *contextOverlap* which is the *hybrid* similarity between all context words. That is, $contextOverlap(c, c') = hybrid((P(c) \cup C(c)), (P(c') \cup C(c')))$. $contextOverlapOffset$ is given as $contextOverlap(c, c') - contextOverlap(c, c'')$. Furthermore, we introduce two features ($hasParents$ and $hasChildren$) for additional insight into the neighbourhood of candidate alignments. $hasParents$ uses nominal features to indicate whether both concepts in a candidate alignment have parent nodes, only one concept have parent nodes, or none have parent nodes. Similarly with $hasChildren$, we indicate the presence or absence of child nodes. Finally, $depthDiff$ is the absolute difference of the relative depths of concepts being compared. The depth of a concept is the number of edges in the shortest path between the root node and that concept. We assume the presence of a top concept (root node) even when an ontology does not specify one. A concept's relative depth is the ratio of its depth to the total number of edges on the concept's path (i.e. from root to leaf passing through the concept). In Figure 2 for example, the relative depth of concept #3945 is 0.5 since #3945 is halfway down on the shortest path.

3.3 Machine learning

The final step is the classification of candidate alignments as either true or false correspondences. We use a Random Forest classifier which is an ensemble method using multiple decision trees for improved classification and to avoid overfitting. Each decision tree uses a subset of features and classification is based on majority voting of decision trees' predictions [1]. Decision trees have been previously shown to outperform other machine learning algorithms for aligning ontologies [16]. In the training phase, feature vectors (as in Table 1) are generated for candidate alignments and class labels are determined by the reference alignments. Reference alignments form the gold standard as they specify actual correspondences between source and target ontologies. When a correspondence from the candidate alignments is also present in the reference alignment, it is labelled as a true alignment, otherwise, it is labelled as a false alignment. In the prediction (or classification) phase, the trained model uses generated feature vectors to determine if unseen candidate alignments are true alignments.

4 Evaluation

4.1 Experiment setup

We perform experiments to evaluate the performance of our approach on two alignment datasets as follows.

Benchmark dataset The Conference track of 2016 Ontology Alignment Evaluation Initiative (OAEI)[10] which consists of 7 small to medium-sized ontologies specifying concepts in the domain of conference organisation. The ontologies have heterogeneous origins resulting in differences in structure and vocabulary. The gold standard is 21 reference alignments representing the entire alignment space between ontology pairs.

EuroVoc dataset This consists of two large controlled vocabularies – the European Union multilingual thesaurus (EuroVoc)[11] and the GEneral Multilingual Environmental Thesaurus (GEMET)[12] describing 7,234 and 5,220 concepts respectively. The gold standard is 1,126 correspondences between equivalent concepts in both ontologies[13].

Alternative alignment approaches

- *StringEquiv*: An OAEI baseline which discovers alignments by exact string matching of concept labels.
- *edna*: Another OAEI baseline which uses edit distance (Levenshtein distance) for approximate string matching of concept labels.
- $WordEmb$ Word embedding approach using Word2Vec's continuous skip-gram model and Wikipedia data dump (version 20161130). Concepts are compared by the cosine similarity their label vectors.
- $Hybrid$ Combines word embedding and edit distance to discover correspondences [22].

Our approach which we refer to as $Rafcom$[14] with two variants, $Rafcom_W$ and $Rafcom_G$ for Wikipedia-based and Google News word embedding models respectively. Leave-one-out approach is used for the Conference dataset by leaving a pair of ontologies out in turn while a model is trained using the remaining dataset. The trained model is then used to aligned left out ontologies. Since the EuroVoc dataset have a pair of ontologies only, we use ten-fold cross-validation for evaluation. Alignment performance is based on standard precision, recall and F-measure which are averaged over all the folds for each dataset. Precision is the proportion of set of correspondences returned that are present in the reference alignment. Recall is the proportion of correspondences in the reference alignment that are discovered by an alignment system.

4.2 Results and discussion

The performances of alignment approaches at best F1-measures are as shown in Tables 2 and 3 for the Conference and EuroVoc datasets respectively. Best

[10] `http://oaei.ontologymatching.org/2016/conference/`
[11] `http://eurovoc.europa.eu`
[12] `http://www.eionet.europa.eu/gemet/en/themes`
[13] `http://data.europa.eu/euodp/en/data/dataset/eurovoc/resource/3430afb6-51c7-44d8-b1c7-a1e045ef5696`
[14] `https://bitbucket.org/paravariar/rafcom`

performances for each evaluation metric are in boldface. Our approach clearly outperformed the others on the Conference dataset for all evaluation metrics with $Rafcom_G$ slightly outperforming $Rafcom_W$. About 84% of true correspondences were discovered in the candidate selection stage and the classifier achieved about 96% accuracy in classifying candidate alignments. Performance differences were more subtle for EuroVoc. In this dataset, $Rafcom_W$ and $Rafcom_G$ had better precisions while *edna* was best in recall. Similar to the Conference dataset, 84% of true correspondences were included in the candidate alignments selected. However, the classifier achieved about 90% accuracy in telling true alignments and false alignments apart. *edna* outperformed *StringEquiv* on both datasets using F1-measures and this is consistent with results at the OAEI challenge and previous works [2]. Also, *hybrid* outperformed its components as had been expected [22].

Table 2: Performances on OAEI 2016 conference track (classes only)

Approach	Precision	Recall	F1-measure
StringEquiv	0.878	0.498	0.635
edna	0.880	0.537	0.667
WordEmb	0.881	0.544	0.673
Hybrid	0.880	0.564	0.687
$Rafcom_W$	0.889	0.680	0.770
$Rafcom_G$	**0.891**	**0.695**	**0.781**

Table 3: Performances on EuroVoc dataset (EuroVoc–GEMET alignment)

Approach	Precision	Recall	F1-measure
StringEquiv	0.580	0.746	0.653
edna	0.572	**0.776**	0.659
WordEmb	0.581	0.746	0.653
Hybrid	0.581	0.768	0.662
$Rafcom_W$	**0.714**	0.632	**0.671**
$Rafcom_G$	**0.714**	0.629	0.669

Figure 3 shows results of alignment systems on the Conference dataset at the OAEI challenge ordered by F1-measure. Although the systems may have competed under a different circumstance, our results are promising when compared with the best systems at the challenge.

Matcher	Threshold	Precision	F.5-measure	F1-measure	F2-measure	Recall
CroMatch	0	0.78	0.77	0.76	0.75	0.74
AML	0	0.83	0.8	0.76	0.72	0.7
LogMap	0	0.84	0.79	0.73	0.67	0.64
XMap	0	0.86	0.8	0.73	0.67	0.63
LogMapBio	0	0.8	0.76	0.71	0.67	0.64
DKPAOMLite	0	0.82	0.76	0.69	0.63	0.59
DKPAOM	0	0.82	0.76	0.69	0.63	0.59
edna	0	0.88	0.78	0.67	0.59	0.54
NAISC	0.98	0.85	0.77	0.67	0.59	0.55
FCAMap	0	0.75	0.72	0.67	0.63	0.61
LogMapLt	0	0.84	0.76	0.66	0.58	0.54
StringEquiv	0	0.88	0.76	0.64	0.55	0.5
Lily	0	0.59	0.6	0.61	0.62	0.63
LPHOM	0.76	0.89	0.71	0.55	0.45	0.4
Alin	0	0.89	0.65	0.46	0.36	0.31
LYAM	0.97	0.48	0.36	0.26	0.21	0.18

Fig. 3: Performance of alignment systems on OAEI 2016 conference track (classes only)[15].

Influence of similarity methods in discovering alignment types The easiest correspondences to discover are exact string matches. Both *hybrid* and *isub* can discover such correspondences. There are observed differences between similarity approaches when concept labels do not match as shown in Table 4. The correspondence "edas#Academic_Event" ≡ "ekaw#Scientific_Event" was found using *hybrid* because "academic" and "scientific" were embedded in similar vector space for an overall similarity of 0.84. "conference#Track-workshop_chair" ≡ "ekaw#Workshop_Chair" was discovered using *isub*. ISUB similarity puts greater emphasis on common substrings resulting in high similarity of 0.91. The similarity between this pair is 0.6 using the Levenshtein distance approach. The word "conference" appeared multiple times in conference# ontology resulting in a low tf-idf weight. The correspondence "conference#Conference_document" ≡ "ekaw#Document" has a high similarity of 0.94 using *vsm* highlighting the reduced importance of "conference". Also interesting is the comparison between "edas#Paper" and "iasted#Submission" which returned low similarity scores for all direct comparisons. The concepts have relations "edas#Document" ≺ "edas#Paper" and "iasted#Document" ≺ "iasted#Submission". Comparing their semantic neighbourhoods using *context* rightly identifies the pair as alignment candidates with 0.76 similarity.

Influence of feature categories We dropped feature categories during classification of candidate alignments to analyse how the features influenced performance. Precision and recall values were observed for each group of feature cate-

[15] `http://oaei.ontologymatching.org/2016/conference/eval.html`

Table 4: Similarity values for some correspondences discovered

Source concept vs Target concept	**Similarity approaches**			
	hybrid	**isub**	**vsm**	**context**
conference#Paper vs confOf#Paper	**1.0**	**1.0**	**1.0**	0.28
edas#Academic_Event vs ekaw#Scientific_Event	**0.84**	0.61	0.34	0.72
conference#Track-workshop_chair vs ekaw#Workshop_Chair	0.56	**0.91**	0.42	0.25
conference#Conference_document vs ekaw#Document	0.57	0.81	**0.94**	0.33
edas#Paper vs iasted#Submission	0.18	0.0	0.0	**0.76**

gories as shown in Figure 4. Previous experiment configurations were reused and performances were based on 10-fold cross-validation on the Conference dataset.

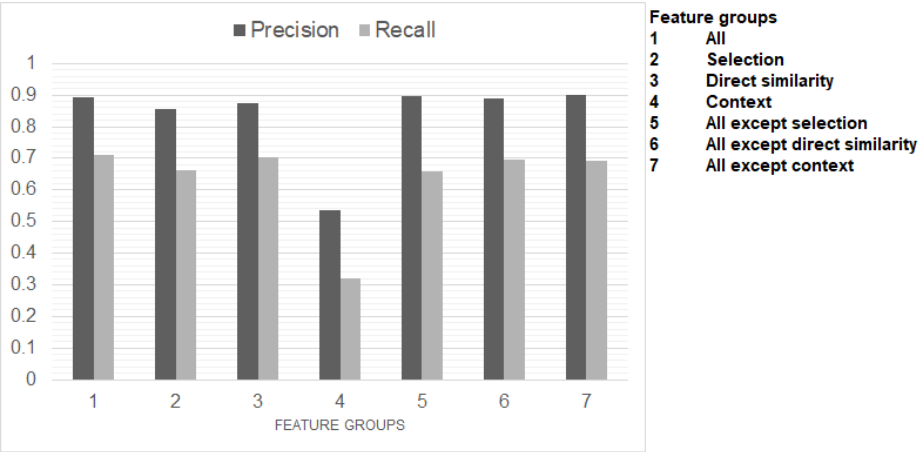

Fig. 4: Impact of excluding features categories.

Classification using all features (group 1) was best but only marginally better than dropping the context features (group 7). Context features contributed least to performance and this is further highlighted by weak performance when

context features alone (group 4) are used for classification. We attribute the weak performance on context features to insufficient data. Analysis showed that only 3% of the candidate alignments identified using context similarity were actual alignments. Accordingly, the classifier model did not learn to effectively use context information due to a significant class imbalance in training data. Also interesting is the slight difference between using direct similarity features alone (group 3) and dropping the direct similarity features (group 6). This suggests that some similarity features are redundant for classifying candidate alignments.

5 Conclusion and future work

We introduced a classifier-based approach for ontology alignment which uses a hybrid of string-based similarity features, semantic similarity features, and semantic context features. Word embedding was used to generate semantic features for a random forest classifier in addition to other novel similarity features. Our experiments showed promising results and outperformed previous known approach which incorporates word embedding. Also, comparison with best-performing alignment systems at the OAEI challenge show that it can outperform state-of-the-art systems.

Future work will investigate a systematic determination of similarity thresholds for selecting candidate alignments and how to deal with class imbalances in generating the classifier model. Also, the ability to transfer a trained model to a different domain will be explored. This is particularly useful in the initial stages of alignment where there are no reference alignments with which to generate a classifier model.

Acknowledgement

This work was supported in part by the British Geological Survey (BGS) through the BGS University Funding Initiative (BUFI S291). We are grateful for the valuable comments of our reviewers.

References

1. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
2. Cheatham, M., Hitzler, P.: String similarity metrics for ontology alignment. In: International Semantic Web Conference. pp. 294–309. Springer (2013)
3. Cruz, I.F., Antonelli, F.P., Stroe, C.: AgreementMaker: Efficient matching for large real-world schemas and ontologies. Proceedings of the VLDB Endowment **2**(2), 1586–1589 (2009)
4. David, J., Euzenat, J., Scharffe, F., Trojahn dos Santos, C.: The alignment API 4.0. Semantic web **2**(1), 3–10 (2011)
5. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.: Learning to match ontologies on the semantic web. The VLDB Journal **12**(4), 303–319 (2003)
6. Euzenat, J., Shvaiko, P., et al.: Ontology matching, vol. 333. Springer (2007)

7. Gulić, M., Vrdoljak, B., Banek, M.: Cromatcher: An ontology matching system based on automated weighted aggregation and iterative final alignment. Web Semantics: Science, Services and Agents on the World Wide Web **41**, 50–71 (2016)
8. Husein, I.G., Akbar, S., Sitohang, B., Azizah, F.N.: Review of ontology matching with background knowledge. In: Data and Software Engineering (ICoDSE), 2016 International Conference on. pp. 1–6. IEEE (2016)
9. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology alignment for linked open data. In: The Semantic Web–ISWC 2010, pp. 402–417. Springer (2010)
10. Li, J., Tang, J., Li, Y., Luo, Q.: RIMOM: A dynamic multistrategy ontology alignment framework. Knowledge and Data Engineering, IEEE Transactions on **21**(8), 1218–1232 (2009)
11. Li, Y., McLean, D., Bandar, Z.A., O'shea, J.D., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. IEEE transactions on knowledge and data engineering **18**(8), 1138–1150 (2006)
12. Lin, F., Sandkuhl, K.: A survey of exploiting wordnet in ontology matching. In: IFIP International Conference on Artificial Intelligence in Theory and Practice. pp. 341–350. Springer (2008)
13. Martínez-Romero, M., Vázquez-Naya, J.M., Nóvoa, F.J., Vázquez, G., Pereira, J.: A genetic algorithms-based approach for optimizing similarity aggregation in ontology matching. In: International Work-Conference on Artificial Neural Networks. pp. 435–444. Springer (2013)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
15. Monge, A.E., Elkan, C., et al.: The field matching problem: Algorithms and applications. In: KDD. pp. 267–270 (1996)
16. Ngo, D., Bellahsene, Z.: YAM++: A multi-strategy based approach for ontology matching task. In: Knowledge Engineering and Knowledge Management, pp. 421–425. Springer (2012)
17. Ngo, D., Bellahsene, Z., Todorov, K.: Opening the black box of ontology matching. In: Extended Semantic Web Conference. pp. 16–30. Springer (2013)
18. Otero-Cerdeira, L., Rodríguez-Martínez, F.J., Gómez-Rodríguez, A.: Ontology matching: A literature review. Expert Systems with Applications **42**(2), 949–971 (2015)
19. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. IEEE Transactions on knowledge and data engineering **25**(1), 158–176 (2013)
20. Stoilos, G., Stamou, G., Kollias, S.: A string metric for ontology alignment. In: International Semantic Web Conference. pp. 624–637. Springer (2005)
21. Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: International Semantic Web Conference. pp. 628–644. Springer (2017)
22. 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. Springer (2014)