Aligning Experientially Grounded Ontologies using Language Games

Michael Anslow and Michael Rovatsos
University of Edinburgh
Edinburgh, United Kingdom
M.Anslow@sms.ed.ac.uk and mrovatso@inf.ed.ac.uk

June 22, 2015

Abstract

Ontology alignment is essential to enable communication in a multi-agent system where agents have heterogeneous ontologies. We use language games as a decentralised iterative ontology alignment solution in a multi-agent system where ontologies are grounded in measurements taken in a dynamic environment. Rather than attempting to ground ontologies through physical interaction, we design language game strategies that involve exchanging descriptions of the environment as graph patterns and interpreting descriptions using graph matching. These methods rely on structural similarity as evidence for ontology alignment. We compare various language game strategies with respect to communication overhead and alignment success and provide preliminary results which show that ontology alignment using language games that rely on descriptions alone can result in perfect alignments with only modest communication overhead. However, this requires that environmental dynamics are reasoned about when providing descriptions and that partial matching of descriptions is used when there are inconsistencies between descriptions and local knowledge.

1 Introduction

Successful communication in a multi-agent system is paramount to successful coordination. To this end, ontologies make the semantics of a language explicit, in a machine readable form, that facilitate in the interpretation of communication. However, when ontologies are heterogeneous, an alignment between them must be found. In this paper, we explore graph-based ontology matching solutions to this problem, where structural similarity between ontologies serves as evidence for ontology alignment. We do this in the context of a multi-agent simulation where ontologies are grounded in measurements of a dynamic environment.

We adapt 'language games' popularised by Steels [1999] as a coordinated communication process that serves as a decentralised, iterative ontology alignment solution. Agents perform communication tasks to distinguish between different elements of their environment by selecting, communicating and reasoning about graph patterns and performing graph matching. Our language games are novel in that they are based on finding overlapping descriptions of a shared environment without relying on alignment through physical interaction alone.

We reduce language games for the purpose of ontological alignment to three subproblems, *target selection*, *context selection* and *correspondence induction*. Target selection consists of selecting a label that agents will attempt to discover an alignment for. Context selection consists of selecting a graph pattern whose structure distinguishes the selected target label from other labels. Correspondence induction consists of inducing correspondences between ontologies based on reasoning about structural similarity between local and communicated knowledge. We use a fixed target selection strategy in this paper and instead focus on context selection and correspondence induction. In particular we explore the quality of different solutions to these problems with respect to correctness of alignments and communication overhead. We believe that graph-based structural similarity can resolve practical, task oriented, ontology alignment problems if we assume a sufficiently distinctive environment, that agents structure their knowledge in the same way, and that agents coexist within and have overlapping knowledge of, their environment.

We only address instance matching in this paper and provide fixed and correct alignments between other elements of ontologies such as concepts and relations. We explore two forms of ontological heterogeneity, coverage and terminological mismatch, as described by Euzenat and Shvaiko [2013] in chapter 2. Difference in coverage of ontologies results from perceiving the environment at the same level of granularity, but, as a result of incomplete information, ontologies represent a partial perspective of a dynamic environment. Terminological mismatch is caused by autonomous labelling of instances and is inevitable given that agents experience their environment in isolation and discover entities independently.

The approach detailed in this paper is applicable to any instance matching problem where agents conceptualise a shared environment at the same level of granularity and use the same concept and relationship labels. One could apply our approach when aligning human constructed ontologies, however, human judgement is variable and as such it is unlikely that humans would create ontologies that match these assumptions. We have not addressed a real-world ontology alignment problem, instead, we favour exploration of a well-defined alignment problem that is generated in a pseudo-random way to ensure robustness over pseudo-random instantiations of this problem. We regard our approach as a first step towards further principled and methodical exploration of language games along the dimensions of ontology heterogeneity, agent behaviour and communication requirements.

We provide preliminary experimental results in a simulated grid-world-like scenario which show that ontology alignment using language games that rely on descriptions alone can result in near perfect alignments. However, this requires that environmental dynamics are reasoned about when providing descriptions and that partial matching of descriptions is used when there are inconsistencies between descriptions and local knowledge.

The remainder of this paper is structured as follows: In section 2, we contextualise our work against related literature. In section 3, we provide a formal definition of the ontology alignment problem within a multi-agent systems context. In section 4, we describe our proposed solution. Section 5 outlines our experimental methodology. We then compare the performance of various language game strategies in section 6. Finally, we summarise our findings and suggest possible extensions to this work in sections 7 and 8, respectively.

2 Related Work

Ontology matching [Euzenat and Shvaiko [2013]] is the automatic/semi-automatic discovery of semantic correspondences between heterogeneous ontologies. Existing agent-based ontology matching approaches involve some form of interaction between agents where agents negotiate the meaning of the correspondences between ontologies [Davidovsky *et al.* [2012]]. The ontologies that these techniques are applied to are typically Semantic Web ontologies, as initially described by Fensel *et al.* [2001]. In general, both agent-based and non agent based approaches to ontology matching rely on the assumption that ontologies are constructed by humans. This allows for a rich plethora of natural language processing tools and resources to be used. However, when ontologies are learnt autonomously from non-human perceptions and labelled in an artificial language, this assumption does not hold.

The 'symbol grounding problem' as described by Harnad [1990] is the problem of how symbols 'acquire' meaning. In robotics and intelligent systems, this problem is that of grounding symbols in data measured by physical sensors. There are two variants of this: 'physical symbol grounding', described by Vogt [2002], which consists of individual agents grounding symbols by interacting with the environment and 'social symbol grounding', the multi-agent extension to this described by Cangelosi [2006], in which agents negotiate the meaning of independently physically grounded symbols. In this work we explore social symbol grounding at a high level of abstraction without taking account of the complexities of low-level signal processing. We believe that anchoring frameworks, as described by Coradeschi and Saffiotti [2003], provide a plausible solution to the low-level counterpart of this abstraction and therefore, we assume a mapping from low-level signals to symbolic labels is readily available.

Though our work is influenced by language games, it is distinct from existing work that use this technique in a number of ways: Agents do not plan actions to discover overlapping measurements, instead, agent behaviour is fixed and overlapping measurements are coincidental, instead agents align ontologies by discovering structural similarity between knowledge; Language games are driven by a need to communicate, as such, language learning is not an end in itself; A shared language is not the result of inventing labels together, instead, agents have an existing language before attempting to align ontologies that corresponds to their conceptualisation of the environment; finally, language games typically only focus on strategies to discover equivalences between elements of an ontology, we also infer disjunction between labels when particular equivalences can be ruled out that are used to constrain interpretations in subsequent language games.

Unlike the work of McNeill and Bundy [2007] we do not explore feedback from a task for the diagnosis and correction of incorrect alignments. Our techniques are evidence based, and as such, prone to error; solutions to correcting incorrect alignments are beyond the scope of this paper.

3 Formal framework

Agents $A = \{1, 2, \cdots, n\}$ exist within an environment. Each agent maintains a conceptualisation of their environment as an ontology. We define ontology as follows:

Definition 3.1. An *ontology* is a tuple $O = \langle \mathcal{C}, \mathcal{I}, \mathcal{T}, \mathcal{D}, \mathcal{R}, \mathfrak{R} \rangle$ where: $\mathcal{C}$ is the set of concepts (classes), $\mathcal{I}$ is the set of individuals/instances, $\mathcal{T}$ is the set of data types, $\mathcal{D}$ is the set of data values, $\mathcal{R}$ is the set of relations, $\mathfrak{R} : (\mathcal{C} \cup \mathcal{I} \cup \mathcal{T} \cup \mathcal{D})^2 \rightarrow \wp(R)$ is a function indicating which binary relations hold among $\mathcal{C}, \mathcal{I}, \mathcal{T}, \mathcal{D}$.

This is similar to the definition used by Euzenat and Shvaiko [2013] in chapter 2, however we do not differentiate between particular classes of relations such as subsumption, equivalence and exclusion as our focus is only on the structure of an ontology rather than reasoning about what particular relationships entail.

Example 3.1 describes an ontology and figure 1 provides a graphical depiction of this ontology. These are representative fragments of the ontologies used in our experiments described in section 5.

Example 3.1. $\mathcal{C} = \{Location, Agent, C_1\}$, $\mathcal{I} = \{A, B, C, D, E\}$, $\mathcal{R} = \{Connected, InLocation, MemberOf, HasValue, HasType\}$, $\mathcal{T} = \{Boolean\}$. As $\mathcal{T}$ only contains $Boolean$, $\mathcal{D} = \{True, False\}$. $\mathfrak{R}$ defines: what concept an instance is a members of ($\mathfrak{R}(D, Agent) = \{MemberOf\}$), which location an instance is in ($\mathfrak{R}(D, C) = \{Inlocation\}$), which locations are connected ($\mathfrak{R}(A, B) = \{Connected\}$), what property an instance has ($\mathfrak{R}(A, True) = \{HasValue\}$) and the type of a data value ($\mathfrak{R}(True, Boolean) = \{HasType\}$).

The set of possible ontologies is $\mathcal{O}$. The environment, $\mathcal{E}$, is itself an ontology in this set. It is at the abstraction of this ontology that agents perceive, reason and communicate about their environment. Agents 'measure' their environment at each time step according to a measurement function,

$$\mu_i \colon \mathcal{O} \rightarrow \mathcal{O} \quad (1)$$

where $\mu_i(\mathcal{E}) = \hat{\mathcal{E}}$ is the measurement received by agent i. $\hat{\mathcal{E}}$ is a sub ontology of $\mathcal{E}$ such that $\hat{\mathcal{C}} \subseteq \mathcal{C}, \hat{\mathcal{I}} \subseteq \mathcal{I}, \hat{\mathcal{T}} \subseteq \mathcal{T}, \hat{\mathcal{D}} \subseteq$

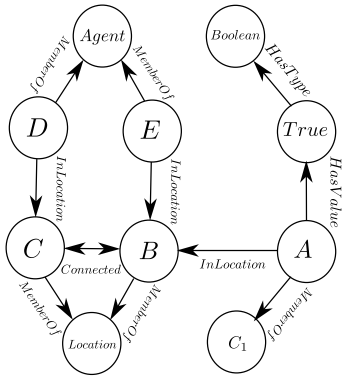

Figure 1: A graphical depiction of an ontology. This ontology depicts three instances in different locations, two of these instances are members of concept Agent and one of an arbitrarily generated concept C_1. The instance of concept C_1 has a property $True$ of type $Boolean$.

$\mathcal{D}$ and $\hat{\mathfrak{R}}$ is a sub function of $\mathfrak{R}$. μ defines how complete each agent's measurements are of $\mathcal{E}$, for example, if $\mu_i(\mathcal{E}) = \mathcal{E}$ then agents have complete measurements.

Given measurements, agents independently learn their own conceptualisation of their environment as an ontology, where O_i is the ontology of agent i. To communicate about ontologies, ontologies are associated with a 'labelling' function,

$$\ell\colon \mathcal{C} \cup \mathcal{I} \cup \mathcal{T} \cup \mathcal{D} \cup \mathcal{R} \rightarrow \mathcal{L}, \tag{2}$$

that associates elements of the ontology with a countably infinite set of labels, $\mathcal{L}$.

We assume that the labelling functions for local ontologies and the environment are surjective in general and bijective for instances. Though elements of $\mathcal{D}$ are not uniquely labelled (e.g. $True$ and $False$ are used for multiple entities), we require that the ontological entities they label can be distinguished from other entities by the entities' relational context. To enforce this, values are always properties of instances that are uniquely labelled. This allows for a value to be determined by the relation it has with an instance.

With these provisions, we can now proceed with introducing our definition of the ontology alignment problem. To facilitate communication between agent i and agent j, agent i must discover an alignment, $\phi \subset \mathcal{L}_i \times \mathcal{L}_j \times \Theta$ where $\mathcal{L}_i$ consists of labels local to agent i and $\mathcal{L}_j$ consists of labels received from j and Θ is a set of semantic relations that can hold between labels. An element of an alignment is called a *correspondence* and is a tuple $\langle l, l', \theta \rangle$ that implies that θ holds between l and l'. We use Φ to denote all agent alignments and the notation $\Phi_{i,j}$ to denote the alignment that agent i has with agent j.

Ontology alignment is only used where communication cannot be understood. This creates the *focus* of the ontology alignment problem. This focus is influenced by: predefined communication rules that dictate what agents attempt to communicate about; fixed action policies that influence the 'behaviour' of agents within the environment; a fixed, correct, alignment for all non instance entities of ontologies that is a subset of all agents' alignments. An agent's behaviour affects what they measure and, as a consequence, their knowledge, which in turn, according to pre-defined communication rules, affects what is communicated. The fixed alignment specifies known and unknown alignments, the latter of which constitute the labels that an agent will not understand when communicated to them prior to ontology alignment methods.

For the purpose of evaluation, we generate gold standard alignments between all agents Φ^*, where $\Phi^*_{i,j}$ is the gold standard alignment between agent i and j. This is created by keeping track of the labels that each agent assigns to $\mathcal{E}$, allowing for a gold standard alignment to be constructed directly from ground truth. Given this gold standard alignment, we can then define our ontology alignment problem as follows: given agents i and j, and their ontologies O and O' respectively, find an alignment $\phi \in \Phi_{i,j}$ from O to O' such that $\phi = \Phi^*_{i,j}$.

4 Proposed solution method

Our solution to this ontology alignment problem is based on using 'language games' defined as follows:

Definition 4.1. A *language game* is a coordinated communication process between two agents i and j, called the comprehender and the explicator respectively, that generates correspondences between $\mathcal{L}_i$ and $\mathcal{L}_j$. A language game consists of three steps:

- **Target Selection**: The comprehender chooses some target label $l_{target} \in \mathcal{L}_j$ that they cannot understand and which they were exposed to in previous communication.
- **Context Selection**: The explicator provides 'context' for l_{target} to distinguish it from other labels.
- **Correspondence Induction**: Given this context, the comprehender infers a correspondence, $\langle l_{local}, l_{target}, \theta \rangle$, by induction, where θ is a semantic relation that holds between l_{source} and l_{target} from a set Θ of possible semantic relations.

We consider each step of a language game to be a *strategy*. In this paper we use a fixed strategy for target selection and focus on context selection and correspondence induction. Agents communicate messages to each other, where the content of a message is a graph pattern (described later in this section). Language games follow a simple protocol shown in figure 2 that dictates the state of communication. This protocol distinguishes between two stages of communication: operational communication, that is communication between agents that is intended to be understood and explication communication, that consists of communication to align ontologies and facilitate future operational communication.

All context selection strategies that we consider consist of the *neighbourhood* of the element that l_{target} refers to. This neighbourhood consists of any element from $\mathcal{I} \cup \mathcal{C} \cup \mathcal{T} \cup \mathcal{D}$ that can be reached by traversing from the element corresponding to l_{target} along relations defined by $\mathfrak{R}$. The relations that are traversed along are also included in the neighbourhood, preserving structural information. We restrict the traversed relations to not include $MemberOf$ or $HasType$ relationships so that traversal is localised. Without this restriction, the neighbourhood of an instance or data value along

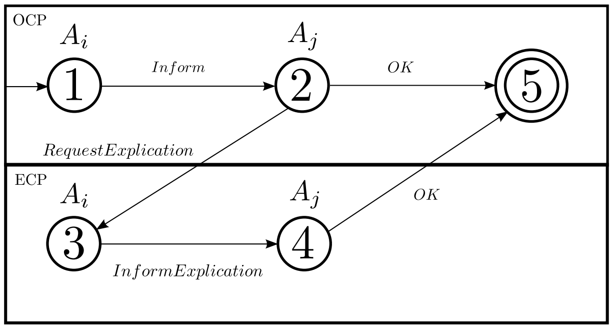

Figure 2: *Operational communication protocol* (OCP): Agent i sends an $Inform$ message to inform agent j about changes in $\mathcal{E}$. If Agent j cannot correctly translate this $Inform$ message, agent j requests explication as a $RequestExplication$ message. *Explication communication protocol* (ECP): Agent i replies with an $InformExplication$ message containing the explication from their content selection strategy. This is a simplified version of the protocol suggested by van Diggelen *et al.* [2006].

two traversed relations could be all instances or data values that share a concept or data type. After the neighbourhood is selected, $MemberOf$ and $HasType$ relations and their concepts and data types are included. Our assumption is that structural similarity/dissimilarity of neighbourhoods serve as evidence for similarity/dissimilarity between elements of those neighbourhoods, in particular, the target element that corresponds to the target label selected in the target selection strategy.

Though this is a very general structure-based similarity measure, in this paper, we are only concerned with applying it to instance matching. As such, the set of semantic relations Θ only consists of equivalence ($=$), that indicates that l_{local} and l_{target} refers to the same instance in the environment, and disjunction ($\perp$), that indicates that l_{local} and l_{target} do not refer to the same instance in the environment.

To facilitate reasoning about structural similarity, we represent ontologies as vertex and edge labelled directed multigraphs graphs as follows:

Definition 4.2. A *vertex and edge labelled directed multigraph* is a tuple $G = (V, E, \Sigma_V, \Sigma_E, \ell, s, t)$ where V is a set of vertices, E is a multiset of ordered pairs from $V \times V$, Σ_V is a set of labels for vertices, Σ_E is the set of labels for edges, ℓ is a labelling function $\ell : V \cup E \to \Sigma_V \cup \Sigma_E$; $s : E \to V$ assigns each edge to its source vertex; and $t : E \to V$ assigns each edge to its target.

We represent agent i's ontology O_i as a graph knowledge base $\mathcal{K}_i$ where $V(\mathcal{K}_i)$ is a set of vertices corresponding to elements of $\mathcal{C} \cup \mathcal{I} \cup \mathcal{T} \cup \mathcal{D}$, $E(\mathcal{K}_i)$ is a set of edges derived from $\mathfrak{R}$ and t and s are defined such that they respect the ordering of pairs in $E(\mathcal{K}_i)$, i.e., $t(\langle v, v' \rangle) = v'$ and $s(\langle v, v' \rangle) = v$.

Agents communicate about their local knowledge by exchanging 'graph patterns' as defined in Barceló *et al.* [2011]:

Definition 4.3. A *graph pattern* π is a tuple with the same elements as those in definition 4.2 except $V = V_{const} \cup V_{var}$,$E = E_{const} \cup E_{var}$,$\Sigma_V = REG(\Sigma_{V_{const}} \cup \Sigma_{V_{var}})$,$\Sigma_E = REG(\Sigma_{E_{const}} \cup \Sigma_{E_{var}})$, indicating that vertices and edges can represent either constants or variables and a regular language over vertex and edge labels denoted by $REG(\Gamma)$ which denotes the set of non-empty regular languages over Γ. We denote π_Σ as the graph pattern labelled with $\Sigma_V \cup \Sigma_E$.

An example of a graph pattern is shown in figure 3.

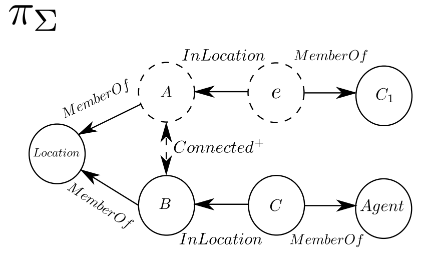

Figure 3: π_Σ expresses that there is an agent instance labelled 'C' that is within one or more connected relations of a Location labelled either 'A' or 'B' that has some instance denoted by variable 'e' of type C_1 in that location.

Further to the notion of a graph pattern, is a graph pattern query. This is a pair $Q = (\pi, \bar{x})$ where π is a graph pattern and $\bar{x}$ is a tuple of elements from $V(\pi)$. This is similar to a conjunctive query where $\bar{x}$ is the head of the query, containing distinguished variables and π is the body of the query, limiting what $\bar{x}$ can be bound to. Given a knowledge base $\mathcal{K}$ and a graph query $Q = (\pi, \bar{x})$ with $|\bar{x}| = k$, the answer to Q on $\mathcal{K}$ is:

$$Q(\mathcal{K}) = \{\bar{v} \in V^k | \mathcal{K} \models \pi[\bar{v}/\bar{x}]\}. \tag{3}$$

Here $\pi[\bar{v}/\bar{x}]$ is the result of substituting $\bar{v}$ for $\bar{x}$ in the pattern π. $\bar{x}$ can consist of any vertices in π that are constants or variables, while its substitution $\bar{v}$ consists of constants from $\mathcal{K}$ constrained by the graph structure in π. We refer to π as the *context* for $\bar{x}$, as π serves to distinguish vertices in $\bar{x}$ from other vertices by constraining how it can map onto a knowledge base.

In our language games, the explicator provides context for l_{target} as a graph pattern query where $\bar{x}$ is a single vertex corresponding to l_{target} and π contextualises l_{target}. The comprehender matches this context against their local knowledge base, finding possible valuations for $\bar{x}$ and hence also for l_{target}. When $|Q(\mathcal{K})| > 1$ the answer to the query, and hence the context provided by the explicator, is ambiguous. The higher the cardinality of $Q(\mathcal{K})$, the more ambiguous the context is. An unambiguous graph query is then one where $|Q(\mathcal{K})| = 1$. It is also possible that $|Q(\mathcal{K})| = 0$, indicating that the context provided by the explicator does not overlap with the comprehender's knowledge. This is expected to occur given that agents have heterogeneous knowledge. Reasoning about ambiguity features prominently in our language games strategies described later.

Before the comprehender matches context from the explicator, the comprehender first translates the context according to a mapping function between sets of labels,

$$map_{i,j} : \mathcal{L}_j \nrightarrow \mathcal{L}_i \tag{4}$$

where $map_{i,j}$ is agent i's mapping function for agent j such that $map_{i,j}(l') = l \iff \langle l, l', = \rangle \in \phi_{i,j}$ where

$l' \in \mathcal{L}_j$, $l \in \mathcal{L}_i$. $map_{i,j}$ is a partial function as agent i does not have a complete mapping from agent j's labels to their own in general. A graph query is then translated as follows: if a mapping for a label belonging to a constant vertex is defined, this label is substituted by its mapping. Otherwise, the vertex corresponding to the constant label is moved to V_{var} making it a variable. The label given to this vertex depends on whether there are known disjunctions for the constant label or not according to alignment $\phi_{i,j}$. If there are no known disjunctions, the label is given a unique variable label. If there are known disjunction, a regular expression of possible alternatives for the constant is created indicating that the label could be one of any label of the same concept for which a disjunction semantic relation does not hold. For example, a translation for a vertex might be $A|B|C$, indicating that the vertex label is either A or B or C.

An example of the graph matching that occurs in a language game is given in figure 4.

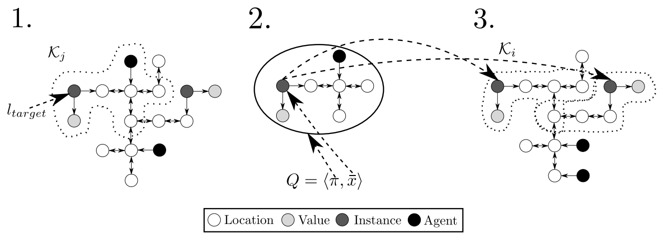
1.
2.
3.
$\mathcal{K}_j$
l_{target}
$Q = \langle \bar{\pi}, \bar{x} \rangle$
$\mathcal{K}_i$
○ Location ◐ Value ● Instance ● Agent

Figure 4: Graph matching in a language game. For clarity, these graphs omit vertices coresponding to members of $\mathcal{C}$ and $\mathcal{T}$ as well as $MemberOf$ and $HasType$ relations. 1. The explicator selects the neighbourhood of a vertex corresponding to l_{target}. 2. This neighbourhood is communicated to the comprehender as a graph query. At this point, the graph query is translated by the comprehender according to their current mapping and alignment with the explicator. 3. The comprehender finds all valuations for this graph query against their knowledge base. In this case, $|Q(\mathcal{K})| = 2$ using partial matching and $|Q(\mathcal{K})| = 0$ using exact matching.

Reasoning about environmental dynamics

To reason about environmental dynamics, agents must maintain a model of 'change' of their environment. An edge has 'changed' with respect to local knowledge if: there is evidence that an observed edge (those edges that they have observed in previous time steps), present in the knowledge base in the last time step, is no longer present, or, if an edge that was not present in the knowledge base in the previous time step is now present. An edge is 'unchanged' if it existed in the knowledge base in the previous time step and still exists in the knowledge base. Each agent maintains a data set for each edge they have observed. This data set is a set of tuples of the form $\langle x, y \rangle$, where y is a label indicating 'changed' or 'unchanged' and x is the elapsed time since the last observed change, e.g., $\langle 5, changed \rangle$ for an observed edge indicates that the edge had changed after 5 time steps. This serves as a training set for a support vector machine [Cortes and Vapnik [1995]] which is used to classify whether an edge that is not currently observed is likely to have changed after a given time. We use this notion of change and this classification technique whenever updating knowledge and reasoning about uncertain local knowledge respectively.

4.1 Language game strategies

In this section we begin by formalising language game strategies in general and followed this by describing particular strategies used in our experiments. We omit a general description of 'target selection' as this is hard-coded in our experiments. Correspondence induction and context selection are defined as follows:

Definition 4.4. *Correspondence induction.* Given the comprehender's knowledge base $\mathcal{K}_i$, l_{target}, context Q from the explicator such that l_{target} is the label of a vertex in $V(\pi)$ where $\pi \in Q$, Θ and the existing alignment of the comprehender with the explicator ϕ, induce a set of correspondences $\phi' \subseteq \mathcal{L}_i \times l_{target} \times \Theta$ if Q provides enough evidence that $\phi' \subseteq \Phi^*_{i,j}$ and update ϕ with these.

Definition 4.5. *Context selection.* Given the explicator's knowledge base $\mathcal{K}_j$, $l_{target} \in \mathcal{L}_j$, the existing alignment of the explicator with the comprehender ϕ and a correspondence induction strategy, send a message to the comprehender with a graph pattern Q such that l_{target} is in $\bar{x}$ where $\bar{x} \in Q$ and such that the comprehender, using the correspondence induction strategy with Q, induces correct correspondences.

The difficulty of these sub-problems stem from the distributed nature of these language games. Agents do not have access to each other's knowledge and even if they did, their knowledge is labelled differently. As such, any solution to this problem relies on the assumption that structural similarity between a translated query and a knowledge base is enough evidence for induction of correct correspondences. Also note that, though we refer to the gold standard alignment Φ^* in our definitions, this is of course inaccessible to agents when reasoning.

Target selection strategy

When agent i receives an operational message from agent j that cannot be translated (i.e. $map_{i,j}$ is undefined for any label in the message), agent i sends an ExplicationRequest message to agent j for the labels with an undefined mapping. Agent i then discards the operational message. Labels are no longer selected for target selection if the explication by the explicator results in no matches ($Q(\mathcal{K}_i) = 0$).

Correspondence induction strategies

The following correspondence induction strategies take as input the parameters described in definition 4.4. The graph queries in correspondence induction are first translated before serving as input for these strategies.

Exact Match: Correspondences are induced as follows: if $|Q(\mathcal{K})| = 1$, induce $\langle \ell(v)$ where $v \in Q(\mathcal{K})$, $l_{target}, = \rangle$; if$|Q(\mathcal{K})| > 1$ induce $\{\langle \ell(v), l_{target}, \perp \rangle | v \notin Q(\mathcal{K}) \wedge v \in V(\mathcal{K})\}$ indicating that l_{target} is disjoint ($\perp$) from some local labels. This is essentially induction based on graph isomorphism where regular expressions must also match.

Partial Match: finds the maximum common sub graph containing a vertex for l_{target} between the query and the knowledge base. The consequence of this is that edges become optional and so can be removed from the query if they do not match. If there are multiple maximum common sub graphs that bind l_{target} to the same vertex, only one of these is selected at random for each possible binding of l_{target}. Induction is then handled in the same way as in the exact match strategy.

Context selection strategies

The following strategies take as input the parameters described in definition 4.5.

K-increasing: We define an algorithm $kcon$ that, given a vertex $v \in V(\mathcal{K})$ and a natural number k, returns the neighbourhood within k edges of v. The k-increasing strategy begins with $k = 0$ and generates a query $Q = \langle v, v \rangle$ where of course v is the only subgraph within 0 edges of v. k is then increased by 1 for each subsequent request of v. The value k associated with requests for explication of a vertex is independent between vertices and requests from different agents. This essentially expands the neighbourhood of vertex v where $l_{target} = \ell(v)$ each time a request is made.

K-selection the explicator chooses the size of the neighbourhood to share by applying the k-increasing strategy against their own knowledge base. The neighbourhood selected is the lowest value of k that is locally unambiguous (where $|Q(\mathcal{K})| = 1$). Before executing the query, the context selected by the k-increasing strategy is translated in reverse: only labels that the explicator believes the comprehender understands are included in the context. Intuitively this answers the hypothetical question: what context would the explicator need to provide to themself to disambiguate a label?

Uncertainty removal We consider variations of k-increasing and k-selection that remove uncertain edges from graph patterns. Uncertainty removal is applied to the graph generated by $kcon$. We also explore an extreme version of uncertainty removal in which all dynamic edges are removed, leaving only static edges.

5 Experimentation

The environment is generated pseudo-randomly: there is a randomly generated component and some predefined relational structure. The environment we use for experimentation is a grid world, where cells are locations, there are agents and instances in cells and there are arbitrarily many connections between locations. The generation of $\mathcal{E}$ is parametrised by a vector $\mathbb{N}^m_{>0}$ where each parameter is used to specifying the number of vertices in the generated graph. This vector corresponds to $\langle$locations, agents, otherInstances, otherConcepts$\rangle$. This allows us to easily specify randomly generated ontologies with only a few parameters. The ontology we consider is:

$O = \langle \mathcal{C} = \{Location, Agent, C_1, C_2, \cdots, C_{\mathbf{E}_4}\}$,
$\mathcal{I} = \{I_1, I_2, \cdots I_{\mathbf{E}_1+\mathbf{E}_2+\mathbf{E}_3\cdot\mathbf{E}_4}\}$, $\mathcal{T} = \{Boolean\}$,
$\mathcal{D} = \{D_1, D_2, \cdots, D_{\mathbf{E}_3\cdot\mathbf{E}_4}\}$,
$\mathcal{R} = \{Connected, InLocation, MemberOf, HasType, HasValue\}, \mathfrak{R} \rangle$

The instances $I_{\mathbf{E}_1+\mathbf{E}_2+1}, I_{\mathbf{E}_1+\mathbf{E}_2+2}, \cdots, I_{\mathbf{E}_1+\mathbf{E}_2+\mathbf{E}_3\cdot\mathbf{E}_4}$, of classes $C_1, C_2, \cdots, C_{\mathbf{E}_4}$, are the possible target instances of our language games. Data values in $\mathcal{D}$ are properties of these instances where: data values are not shared between instances. There are initially an even number of true and false instances.

The relations are defined by $\mathfrak{R}$ as follows:

$MemberOf$ relations are created such that $\forall i \in \mathcal{I} \exists c \in \mathcal{C}$ s.t $\mathfrak{R}(i, c) = \{MemberOf\}$. In particular:

- $\forall location \in \{\mathcal{I}_1, \mathcal{I}_2, \cdots, \mathcal{I}_{\mathbf{E}_1}\}$, $\mathfrak{R}(location, Location) = \{MemberOf\}$.
- $\forall agent \in \{\mathcal{I}_{\mathbf{E}_1+1}, \mathcal{I}_{\mathbf{E}_1+2}, \cdots, \mathcal{I}_{\mathbf{E}_1+\mathbf{E}_2}\}$ $\mathfrak{R}(agent, Agent) = \{MemberOf\}$.
- Each target instance has a $MemberOf$ relation with a single concept from $C_1, C_2, \cdots, C_{\mathbf{E}_4}$ such that there are an even number of instances per concept.

$HasType$ relations are created such that all elements of $\mathcal{D}$ have the type $Boolean$. $InLocation$ relations are created incrementally between non location instances and a randomly selected location that does not already contain an instance of the same type. If two instances of the same type are in the same location, they would not be distinguishable from one another. $Connected$ relations are randomly generated between $Location$ instances using a variation of Erdős-Rényi $G(n, p)$ [Gilbert [1959]] random graph model where vertices (n) are $Location$ instances and edges are $Connected$ relations that hold from one location to any other location with a probability of p. To ensure that locations are fully connected, we connect a random vertex in one component to a random vertex in another component until the graph is fully connected. An example of a generated environment is shown in figure 1. This is generated with parameters $\langle 2, 2, 1, 1 \rangle$.

5.1 Environment dynamics

Only data values and the location of agents change in the environment. When the environment is created, there are initially an even number of $True$ and $False$ data values. Values $d \in \mathcal{D}$ then alternate between $True$ and $False$ according to a parameter $\rho \in (0, 1]$ where each $d \in \mathcal{D}$ alternates value with a probability ρ at each time step. Agents create plans to move stochastically in the environment. When agent i does not have a plan, it selects a location at random to travel to. It uses A* search to plan a path along $Connected$ relations from its current location to its target location. At each time step it moves along one $Connected$ relation into a new location. When agent i arrives at its target location, agent i re-plans in the same way.

5.2 Measurements

Measurement $\hat{\mathcal{E}} = \mu^t_i(\mathcal{E})$ is received by agent i at time t. The set of vertices received by agent i are: A vertex representing agent i, the location agent i is in, any instances in that location (including other agents), any data values and data types of these instances, all locations along one $Connected$ relation from their current location and the instances in any of these locations but not those instances' data types or data values.

5.3 Operational communication

Agents can communicate with each other from any location and at any time. They communicate whenever they observe an edge change containing a target instance. The content of an operational communication message is the edge that has changed.

5.4 Updating $\mathcal{K}$

At the start of each experiment, we provide agents with a snapshot of the the complete environment. Agents essentially duplicate this snapshot of the environment as their local knowledge. We do not address the problem of deciding when an agent has explored their environment enough to begin disambiguation from descriptions. They then spend a fixed amount of time learning the dynamics of the environment from measurements. During this time, differences in measurements and behaviour results in different knowledge.

There are only two assumed common knowledge rules used by agents to detect change: 1. An agent cannot be in two locations at once therefore if an agent is seen in a new location, it no longer holds that it is in the old location. 2. There can be only one data value that is a property of an instance with a particular relation name, therefore a new value for this property overwrites an old value. Given these definitions of change, the way in which agents update their knowledge from successfully translated operational messages and measurements is the same: new edges are added if they do not occur in $\mathcal{K}$, and inconsistent edges are removed.

6 Experimental results

We compare pairs of context selection and correspondence induction strategies with respect to correctness of alignments and the amount of context required to achieve this. To measure the amount of context sent, we count the number of edges sent in explication messages, excluding edges that indicate class membership, i.e., $HasType$ edges. We then average the number of edges across the number of explication messages sent.

To measure correctness of alignments, we use semantic precision and recall described by Euzenat [2007]. Given an alignment $\phi \in \Phi_{i,j}$ and the gold standard reference alignment $\phi^* \in \Phi^*_{i,j}$, semantic precision and recall is calculated as $P(\phi, \phi*) = \frac{C(\phi) \cap C(\phi*)}{C(\phi)}$ and $R(\phi, \phi*) = \frac{C(\phi) \cap C(\phi*)}{C(\phi*)}$ where $C(\cdot)$ is the deductive closure of an alignment under its entailments, i.e., all alignments that can be deduced from other alignments. We use $F\text{-}score$ as a combined measurement of precision and recall defined as $F\text{-}score(\phi, \phi*) = 2 \cdot \frac{P(\phi,\phi*) \cdot R(\phi,\phi*)}{P(\phi,\phi*)+R(\phi,\phi*)}$. We then use the mean $F\text{-}score$ across all alignments of all agents excluding the fixed known alignments. Semantic precision and recall is often not possible to compute in the general case. However, our entailments are simple: If an equivalence (=) between two labels that refer to instances of the same class is known, a disjoint ($\perp$) semantic relation is deduced between these two labels and all other labels of instances of the same class.

The results of our experiments are shown in figure 5. The parameters used in all experiments are : $\rho = 0.1$, $\langle 100, 10, 5, 4, 4 \rangle$ as environment generation parameters, and $n = 100$, $p = 0.1$ as the random graph parameters. Agents learn dynamics of their environment for 1000 steps before communicating and all experiments are run over 10 repetitions. A repetition is complete when all agents have found correct or incorrect equivalence correspondences for all instances, or when they have exhausted attempts at finding correspondences. For the later case we set a limit of 10 requests for explication for an instance target.

Our results show that exact matching results in a lower $F\text{-}score$ than partial matching (exact-kinc vs. part-kinc). However, removal of uncertain context by the explicator improves these scores dramatically (exact-kinc vs. exact-kincrem). This is because the removal of uncertain context reduces the structural difference between the explication context and knowledge of the comprehender. In the case that agents only communicate static context, agents achieve optimal $F\text{-}score$s. This demonstrates that the environment is simple enough, and the agent strategies powerful enough to resolve the ontology alignment problem through static context alone. Moreover, the amount of context provided to do so is quite modest; only 5 edges per target for the best performing pairs of strategies (exact-kselnd).

We expected inclusion of some dynamic context to perform better than communicating static context. For example, if agents are in the same location and attempt to communicate about a target in that location, context that includes the agents in the description should result in a correct correspondence with less context than omitting the agents from the description. This suggests that agents' knowledge of dynamic elements of the environment are still too different to be used successfully as context under our uncertainty removal approach.

K-selection strategies result in worse $F\text{-}score$s than k-increasing strategies in general. This is because, parameter k in the k-increasing strategy is essentially fine-tuned through trial and error by the comprehender via repeated requests for explication. Furthermore, assumptions made by k-selection strategies based on local knowledge may not hold for other agents, resulting in repeatedly sending context that cannot result in a match (part-ksel vs.part-kinc). However, when the assumption about what context is needed is correct, the same $F\text{-}score$ can be achieved with much less context (exact-kselnd vs. exact-kincnd).

The best performing strategies in our results achieve a perfect $F\text{-}score$ with between 5.4 and 9.9 edges of context per target node. This is quite a modest communication overhead requirement to enable correct interpretation of messages.

7 Discussion

The context selection strategies that we have explored in this paper focus on finding context that is both unambiguous and shared between agents. In future work, we plan to extend context selection strategies by both identifying and exploiting salient features of the environment and including approximate dynamic information in selected context. One can identify salient parts of the environment by statistical analysis of graphs and use this information to bias context selection strategies towards more salient parts of the environment. Our results have shown that simply removing uncertain dynamic context only goes so far in improving alignment success. Rather than removing uncertain context all-together, bounding uncertainty by providing approximate context may be beneficial. For example, if the location of an agent is uncertain, rather than excluding the agent from explication, the explicator can include the agent's approximate location as a regular expression over $Connected$ relations bound by their

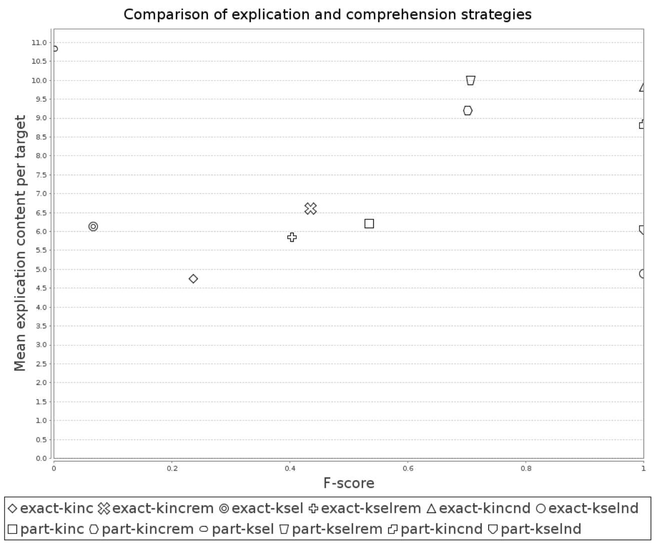

Figure 5: $F\text{-}score$ and explication content communicated for pairs of correspondence induction and context selection strategies. Legends abbreviations: exact = exact matching, part = partial matching, kinc = k-increasing, kincrem = k-increasing with uncertainty removal, ksel = k-selection, kselrem = k-selection with uncertainty removal, kselnd = k-selection with no dynamics, kincnd = k-increasing with no dynamics.

maximum expected distance travelled since they were last observed.

As well as extending context selection strategies, we also plan to address target selection strategies. When operational communication messages contains multiple misunderstood labels, or more generally, when an agent has a pool of possible misunderstood labels to select from, the comprehender must choose a sequence of target labels as the focus of language games. Further to this, the comprehender can select multiple targets in a single explication request, requiring that the explicator disambiguates all of these targets in a single explication. In future work we plan to explore language game selection strategies with respect to these problems.

8 Conclusion

In this paper, we proposed a novel combination of language games and graph-based knowledge representation as a solution to decentralised ontology matching between agents situated in a shared environment where ontologies are representations of agents' beliefs about their environment. To this end, we defined a language game as a sequence of three strategies: target selection, correspondence induction and context selection. We compared the performance of various correspondence induction and context selection strategies given a fixed target selection strategy.

Our results show that structural similarity alone can help align ontologies that are at the same level of granularity without agents utilising grounding through physical interaction with only a modest communication overhead. However, environmental dynamics and incomplete measurements that result in different local knowledge must be reasoned about for this to be possible. We have also shown that the shortcomings of a correspondence induction strategy can be ameliorated by the choice of context selection strategy and vice versa.

In future work we plan to explore more complex language game strategies. In particular, context selection strategies that identify and reason about salient features of the environment and the inclusion of approximate dynamic information as context, and target selection strategies, where agents must select sequences of targets for language games and where a single language game can involve multiple targets.

Acknowledgments

The research reported in this paper is supported by an EPSRC Industrial CASE Award PhD Studentship with Selex ES Ltd.

References

Pablo Barceló, Leonid Libkin, and Juan L Reutter. Querying graph patterns. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 199–210. ACM, 2011.

Angelo Cangelosi. The grounding and sharing of symbols. *Pragmatics & Cognition*, 14(2):275–285, 2006.

Silvia Coradeschi and Alessandro Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2):85–96, 2003.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Maxim Davidovsky, Vadim Ermolayev, and Vyacheslav Tolok. A survey on agent-based ontology alignment. In *ICAART (2)*, pages 355–361. Citeseer, 2012.

Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2nd edition, 2013.

Jérôme Euzenat. Semantic precision and recall for ontology alignment evaluation. In *IJCAI*, pages 348–353, 2007.

Dieter Fensel, DL McGuiness, Ellen Schulten, Wee Keong Ng, Ge Peng Lim, and Guanghao Yan. Ontologies and electronic commerce. *Intelligent Systems, IEEE*, 16(1):8–14, 2001.

Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, pages 1141–1144, 1959.

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990.

Fiona McNeill and Alan Bundy. Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(3):1–35, 2007.

L. Steels. *The Talking Heads Experiment. Volume 1. Words and Meanings*. Laboratorium, Antwerpen, 1999.

Jurriaan van Diggelen, Edwin D de Jong, and Marco A Wiering. Strategies for ontology negotiation: Finding the right level of generality. In *Agent Communication II*, pages 164–180. Springer, 2006.

Paul Vogt. The physical symbol grounding problem. *Cognitive Systems Research*, 3(3):429–457, 2002.