It makes sense ...and reference: A bag-of-words model for meaning grounding and sharing in multi-agent systems

Theodosia Togia and Fiona McNeill
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom
{ttogia, fmcneill}@inf.ed.ac.uk

Abstract
In a semantic web-like multi-agent environment ontology mismatch is inevitable: we can't realistically expect agents created at different times and places by different people to commit to one unchanging universal ontology. But what happens when agents attempt to interact? Can we benefit from ontology matching techniques to prevent communication breakdown? In this paper we present the Semantic Matcher, a system created to handle semantic heterogeneity (i.e. the use of different words for similar meanings), which is one of the most common mismatches across existing web-based ontologies. We claim that independently created agents with different ontologies are bound to miscommunicate because word meanings are private and not grounded in the real world. We propose the solution of achieving reference by means of sense; a common argument within Philosophy of Language. In our work senses are simulated by bags of words, constructed for each lexical item known to a service requesting agent and for selected lexical items known to a service providing agent in a given interaction. In the end we show that adopting this notion of 'sense' for words in formal ontologies allows for successful matching and points to a research direction where the vast amount of data available on the web can be used to facilitate agent communication without assuming a universal ontology.

1 Introduction
In open multi-agent systems agents can request and provide services in order to automatically carry out complex tasks. These agents, called requesters and providers respectively\(^1\), have ontologies which are quite often based on a variant of the BDI model (Beliefs, Desires, Intentions) (see [Wooldridge, 2009] for an introduction): they have beliefs (i.e. facts in their knowledge base), desires (i.e. goals; facts that they would like to make true, that is to add among their beliefs) and intentions (i.e. actions whose effects amount to them fulfilling their desires). In order to bring about a desired state of affairs, service requesting agents need to follow a course of action, which is decided by planning.

A requester will typically have frequent and fleeting interactions with a potentially large number of providers that it has never talked to before. Therefore, interaction will be unpredictable before run-time and communication will usually be unsuccessful given the expected ontological mismatches. A solution to this problem was proposed by McNeill (see [McNeill and Bundy, 2007]), who built the Ontology Repair System (ORS), the first known system that attempts ontology matching on the fly, that is during agent interaction. Similarly, the Semantic Matcher, described in the current paper, is - to our knowledge - the first example of on-line semantic matching. Since the Semantic Matcher is embedded into ORS, it is important to briefly describe the overall system before we move on to later sections.

ORS has been designed as a plug-in to a service requesting agent that can consult the system in order to make its ontology conform to the provider’s view of the world. In a multi-agent environment ontology matching has to be fast, therefore ORS only ‘fixes’ parts of the requester’s ontology that are relevant to a given encounter. In fact, dealing with incomplete information will be very common as providers might not be willing to reveal their full ontology to a service requesting agent or to any ORS-like system.

The lifecycle of a service requesting operation is as follows:

1. The requester uses its ontology to form plans, which consist of actions that usually require interaction with other agents;
2. The requester begins to execute the plan by contacting a suitable service provider to perform the first action;
3. If this action succeeds, the requester continues with the rest of the plan. If not, it calls ORS to diagnose and fix the problem:
   - ORS analyses the communication so far to identify possible mismatches. This may lead to a precise diagnosis or a ‘best guess’ diagnosis, or it may prompt the requester to further question the

\(^1\)In previous work (e.g. [McNeill and Bundy, 2007; Togia, 2010]) we have referred to them as Planning Agents and Service Providing Agents respectively.
provider in an attempt to locate the source of the problem;  
- Once a diagnosis has been made (in the worst case, this may just be marking some part of the requester’s ontology as unusable in conjunction with the provider), ORS repairs the requester’s ontology on this basis;  

4. The requester replans with its improved ontology and starts again.

One assumption of ORS is that agents in the future will have first-order ontologies, which are capable of supporting planning and can have mismatches of many kinds (e.g. differences in predicate arity, differences in pre-conditions for actions etc). The original ORS can successfully deal with some of these mismatches (see [Togia, 2010] for a detailed list) but always taking for granted that agents share the same words and understand them in the same way, thus ignoring semantic heterogeneity for the sake of simplicity. In this paper we demonstrate how this problem was handled with the creation of the Semantic Matcher, an ORS module that initiates negotiation of meaning between agents that have different vocabularies.

The following section explains the problem of semantic mismatch in agent communication situations and places it in the context of previous work. Section 3 discusses the philosophical and practical challenges associated with deriving the meaning of terms in ontologies. Section 4 explains our method of using a bag-of-words model for matching terms in formal ontologies on the fly and describes the architecture of our Semantic Matcher. Section 5 presents our initial results. Section ?? describes ways in which this work can be extended and Section 6 concludes the paper.

2 Semantic matching in multi-agent systems
Two agents trying to communicate on the web are very likely to have ontologies that differ both structurally and semantically. Structural heterogeneity is dealt with by the original ORS under the simplifying assumption that words and their meanings are shared between the two agents. Once this assumption is removed, the system needs to align words before it performs any structural repairs.

Semantic matching is necessary in situations where communication is hindered as a result of agents’ lexicalising

---

2We assume these must be directed yes/no/don’t know or instantiation questions rather than request for large parts of the provider’s ontology to be revealed.
3The current implementation assumes that the provider is always right and disposes of the requester’s original ontology. We are currently investigating more sophisticated ways of using this information.
4Examples of structural mismatch are different number of arguments for equivalent predicates, different number of preconditions or effects for equivalent axioms, differences in quantifiers, implication and many others (see [Togia, 2010]). Semantic mismatches are differences in wording: for instance, the requester might represent UK currency as ‘GBP pounds’ while to the provider the same notion might be known as ‘UKCurrencySterling’.

similar concepts in a different way. For example, imagine a requester that contacts a provider in order to buy a book. The provider is willing to perform this action on some conditions and starts by asking the requester if its ‘credit_card_balance’ is at least £5. The requester has never heard of this word before but has the same concept under the label ‘moneyAvailable’. Our system has to perform semantic matching and suggest that ‘moneyAvailable’ should be changed into ‘credit_card_balance’ in order for the transaction to proceed.

The design of our system has benefited from previous work done in the area of Ontology Matching but, as we shall see, had to diverge significantly. One notable piece of reasearch is presented in Giunchiglia and Shvaiko’s article Semantic Matching [Giunchiglia and Shvaiko, 2003], where the authors describe a process in which a ‘match’ operator takes as input two graph-like constructs (e.g. ontologies, web directory structures etc.) and produces as output a mapping between semantically equivalent nodes by looking at their ‘parents’, ‘sisters’ and ‘children’. This enables nodes like ‘Europe’ and ‘pictures’ to be matched as equivalent (synonymous) if their ancestors are ‘Images’ and ‘Europe’ respectively, given that both nodes mean ‘pictures of Europe’. Another work worth mentioning is a system built by Qu and his colleagues [Qu et al., 2006], which provided the inspiration for the design of our on-line Semantic Matcher. This system computes correspondences between nodes of RDF graphs, which are Uniform Resource Identifiers (URIs), by constructing ‘virtual documents’ for each one of them. The term ‘document’ comes from Information Retrieval and means a multiset of words that represents a web page. In the system described, ‘virtual documents’ representing nodes are compared for similarity using the vector space model [Salton et al., 1975]. The bags are generated from the tokenised URI (i.e. name of node) but also from ‘neighbouring information’, that is names of other nodes connected to it in the graph. Examples of many other node matching techniques can be found in [Euzenat and Shvaiko, 2007].

For our system previous ontology matching methods could not be followed to the letter. Some reasons are: 1) They all deal with simple taxonomies or at best Description Logics. However, in our case ontologies are based on First Order Logic, whose semi-decidability prevented us from attempting to traverse their structure and performing complex computations. 2) To our knowledge, no previous work exists that performs semantic matching on the fly, which case computational time is a serious consideration. 3) Our system has to deal with incomplete information because it is reasonable to assume that service providers will not be willing to reveal their full ontology to service requesting agents because of privacy concerns. ORS only has access to the ontology of the requester, so it cannot align two ontologies but rather map unknown terms from the provider’s ontology (revealed during interaction) to the best-matching terms from the requester’s ontology.

Apart from the above implementation challenges, our system also has to meet an important theoretical challenge: it has to tackle the root of the semantic mismatch problem. In the next section we claim that what inhibits meaning sharing
among agents is the fact that words in agents’ ontologies are not grounded to the objective world without human interpretation. Later we will demonstrate how our system enables ontology terms to make reference to real-world entities by creating sense, that is a ‘mental representation’ for each word in the agent’s ‘mind’.

3 Sense and reference in agent interaction

Humans use language in order to describe reality. Words, phrases and other grammatical constructions are only useful to the extent that they can make statements about the world around us. In fact, any definition that attempts to assign meaning ignoring the actual world is bound to cause infinite regress because words (and whatever is made up of words) are symbols and if they rely solely on other symbols for their meaning, at least some primitive symbols have to be grounded. In current Philosophy of Language research linguistic meaning is usually accommodated in the framework of Truth-Conditional Semantics, where meaning is reference to the world: the meaning of a proper name is the individual named [Kripke, 1972], the meaning of a noun is a set of entities that it designates, the meaning of a verb is either a set of entities or a set of tuples of entities and the meaning of a sentence is a set of situations that make it true. When words are combined, the meaning of the construction is a combination (usually intersection) of the sets they point to.

Human communication is possible if the speakers involved have a shared understanding of the linguistic symbols used, that is if they all associate the symbols with the same entities or sets in the world. So, reference is essential for communication. However, restricting semantic assignment to reference only is not immune to problems because 1) the referent might not exist, 2) words might have some extra meaning apart from their referent, 3) reference can only be achieved by means of a mental representation, that is a meaning ‘in the head’: The third reason is very important for our work and can be easily understood if we think about the meaning of nouns. For example, the meaning of ‘cat’ is the set of cats in the world. This is perfectly acceptable semantics but humans are unlikely to see or have seen all the cats in the world. So, when they see a cat that they have never seen before, they need some rules that help them decide if this entity belongs to the set of cats or not. These rules can be of many kinds, for example, individually necessary and jointly sufficient conditions encoded as a ‘concept’, that is mental representation. There are many theories that try to accommodate the structure of concepts. For an extensive overview the reader can be referred to [Laurence and Margolis, 1999].

The distinction between the ‘mental’ and the ‘physical’ facet of meaning was first made by Gotlob Frege [Frege, 1892], who called them sense and reference respectively. A few decades later, Carnap [Carnap, 1947] introduced the terms intension and extension, which are widely used in Philosophy of Language. A common way of visualising the fact that sense determines reference is Ogden and Richards’ [Ogden and Richards, 1923] ‘meaning triangle’. In Figure 3 the linguistic symbol “cat” on the left stands for the set of entities that the word extends to (i.e. all the cats in the world), but this relation is only indirect – hence the dotted line – because what mediates is the ‘thought’, that is the sense CAT of the word.

The bottom line is that to achieve communication we need reference and to achieve reference we need sense. But what happens in the case of agents?

Agents use ‘words’ (predicates or constants) in order to form ‘sentences’ (formulas9). Their ontologies are written in languages that have clearly defined semantics with symbols, combinatorial rules and a universe of discourse which represents the real world. Sets of entities from this universe give meaning to the predicates and ultimately to formulas. However, it is important to mention that the grounding of the ontology symbols to the actual world is performed by means of an interpretation function, which is where subjectivity enters the scene: ontology words have the meaning that humans give to them. In other words, it is the sense in the human head that determines reference to the real world. This is acceptable from a formal semantics point of view, but it does nothing to prevent communication breakdown between agents. If we want agents to communicate as if they were human, we need to make them fix the reference themselves. In other words, they need to be the ones that understand the meaning of their terms in order to share it with other agents. Therefore, we will need to create mental representations (senses) for every word in the agent’s ‘head’ as shown in Figure 3.

As we will see in the next section, senses are represented as bags of words which are created automatically by aggregating information from various sources inside the ontology. Each word in the service requesting agent’s ontology will have a bag associated with it, which will represent the agent’s understanding of the word, that is a sense which will help the agent determine the word’s reference in the world. Comparing senses from the requestor’s and the provider’s ontology is a way to simulate negotiation of meaning and help the agent communicate.

---

5To be more precise, linguistic ‘symbols’ are actually signs, that is they are conventionally associated with their referrent.

6e.g. ‘Santa Claus’

7e.g. If the person named could exhaust the meaning of a proper name then a sentence like ‘Superman is Clark Kent’ would be a tautology.

8i.e. predicates instantiated with constants and perhaps combined with other formulas in order to encode something that the agent believes to be true.

9or sets of tuples of entities.
The system has a theoretical justification, but it also solves previously unsolvable practical problems, namely online semantic matching with limited access to the service provider’s ontology. The following section breaks it down into its components and explains the design decisions behind them.

4 Implementation

Our system is a search engine whose ‘queries’ and ‘documents’ are senses. As a thought experiment imagine a ‘Google of meanings’, where we can input a mental representation and get as output a ranked list of similar mental representations.

Our model of semantic matching involves incorporating bags of words in formal ontologies, in order to achieve some semantic grounding - i.e. relations between the words in the ontology and the objective world. The Semantic Matcher in ORS is a search engine that tries to find the ‘best match’ for the provider’s lexeme among the candidate lexemes in the requester’s ontology. The provider’s lexeme is expanded into a bag of words that make up its intensional meaning (i.e. sense). This bag serves as a query to the search engine. The requester’s candidate lexemes are also bags or words, acting as a collection of documents which will be ranked from the most to the least relevant after the search has been performed.

The architecture of the Semantic Matcher is illustrated in Figure 4. The Semantic Matcher as a search engine can be broken down into the following three components:

1. **Training the text acquisition model**
   The Text Acquisition Model is a set of databases created by our WordNet and SUMO parsers (see Figure 4), taking as input the lexical resource Wordnet [Gross and Miller, 2005], a database of SUMO-WordNet mappings [Niles and Pease, 2003] and a collection of 645 ontology files (different versions of 38 ontologies that extend SUMO). The text acquisition model only makes use of Wordnet synonyms, ignoring hyponyms and instances, since this tends to disorient us from the lexeme’s meaning. The 645 ontology files provide us with natural language documentation and superclass or type information (i.e. what a lexeme is a subclass or an instance of) for some SUMO lexemes, which is necessary since SUMO is a modular ontology.

   The above databases were parsed to create a set of new easily readable databases. WordNet and SUMO-WordNet mappings were scanned with regular expressions using the WordNet parser and the ontology files were parsed with the SUMO parser (which is able to deal with SUO-KIF and is robust to errors such as unbalanced parentheses or quotes).

   The text acquisition model that these databases form is a collection of resources (later read as lookup tables; dictionaries) that will determine what information can enter the bag for each lexeme in the requester’s ontology. These files have been created once and will only have to be re-computed if the ontologies or WordNet version have to be updated. Their format is very easily processable so as to decrease the computational time of the next stage.

2. **Sense Creation and Term Weighting**
   This module takes as input the requester’s ontology and the databases created in the previous stage and returns a collection of bags of words, each one of which represents the intensional meaning (‘sense’) of a candi-

---

10something like what we would submit as input to Google; not to be confused with a formal query written in languages such as SQL, providerRQL etc.


12the language that SUMO is written in
date lexeme. Each bag contains weighted words, that is words with a co-efficient of importance.

We use the term sense creation to cover both text acquisition, where some text relevant to the lexeme’s meaning is extracted from the databases, and text transformation, where this text is converted into words: the two sub-processes of creating the bags of words. Acquisition and transformation take place many times for every lexeme, since bags are filled incrementally.

The text aquired from the databases is put through a process of tokenisation (essentially, turned into words, e.g., camel case words separated into individual words), stopping (removal of semantically vacuous words) and stemming (reducing words to their stem so that it is easier to identify repetition of words, and therefore gauge their significance). Next, every word in every bag is assigned a weight, indicating how well that word represents the bag. This is done using the tf-idf (term frequency - inverse document frequency) weighting scheme [Robertson and Jones, 1976].

3. Query Processing Whilst the previous two stages can occur off-line, before interaction commences, this stage must be done online, during interaction, because it is only during interaction that the unknown lexemes are revealed. Once interaction failure occurs, ORS performs a diagnosis and, in the case that an unknown lexeme is identified, calls the semantic matcher to identify which of the candidate lexemes is the best match for this unknown lexeme.

5 Evaluation

The system was evaluated using different versions of the SUMO ontology and its sub-ontologies from the Sigmakee repository\textsuperscript{13}. When terms are changed between SUMO versions (e.g. "Corn" becomes "Maize"), we have an objective way of measuring the performance of the matcher because we can safely regard terms and their renamings as synonyms and compare these pairings with our system’s prediction. Initial results are encouraging, with 578by the system, 19The outcome of this implementation cannot be compared against previous work because, to our knowledge, it is the first attempt to incorporate semantic matching in an agent communication system that has minimal access to one of the ontologies, but given the goals that ORS wants to achieve, we believe that these results are very encouraging. However, because of the sparse data available in the repository, it would be too early to jump to conclusions. We are currently evaluating this work more fully. In particular, we are investigating the possibility performing large-scale evaluation using pairs of terms from manually matched ontologies, which will be compared to the pairs predicted by the system. We will not need to use matched first-order ontologies, which are after all hard to find. Only the pairs of terms will suffice. We hope that our full evaluation will confirm our initial positive results.

6 Future goals

In the above sections we showed that a bag-of-words model can simulate word meaning in ontologies and facilitate communication between agents with semantically heterogeneous ontologies. However, we believe that perhaps the most important contribution of this work is that it paves the way for a new approach in ontology matching where senses are not just created from within the ontology but also discovered on the web and therefore take advantage of the vast amount of data available.

A bag-of-words model is structurally similar to a broad folksonomy [?]. Folksonomy is a bottom-up, not centrally controlled classification system in which structure emerges out of the practice of users labelling digital resources (‘objects’) with keywords (‘tags’). Vander Wal distinguishes between broad folksonomies and narrow folksonomies [?]. The former are created when a particular object can be tagged by different people so the same tag can be assigned more than once. For example a Delicious\textsuperscript{14} bookmark about, say, chocolate can have the word ‘recipes’ assigned to it 600 times, the word ‘chocolate’ 578 times, the word ‘food’ 423 times and so on. This pattern reveals some trends as to what vocabularies are generally considered appropriate to describe this resource. Narrow folksonomies, on the other hand, are formed in systems where one object can be labelled only by its author with distinct tags. For example, a Flickr\textsuperscript{15} user can submit a photograph and annotate it with keywords such as ‘surfing’, ‘waves’, ‘beach’ and ‘summer’. If it is made publicly available, it can be found by other users who search for photos about ‘surfing’ or ‘waves’ and so on.

Folksonomy is a way to annotate digital resources on the web for ease of retrieval and categorisation. Bringing folksonomy into formal ontologies is an attempt to annotate physical resources, that is things in the actual world. It should be noted that in our work so far folksonomies have been constructed from sources (e.g. comments in the ontology) that might reveal the ontology engineer’s intended meaning. Therefore, they are not products of collaborative tagging. One of our future goals is to use tag data from online sources in order to help create senses in the agent’s ‘head’\textsuperscript{16}.

The idea of statistically approximating senses has been extensively discussed by Halpin (e.g. [Halpin, 2009]) in the context of semantic web architecture. In all these works the author argues that “the meaning of any expression, including URLs [equivalent to our ‘lexemes’ in ontologies], are grounded out not just in their formal truth values or referents, but in their linguistically-constructed ‘sense’”. Halpin’s ‘sense’ is built on late-Wittgenstein’s [Wittgenstein, 1953] idea of socially constructed meaning: the meaning a URI (or an ontology word in our case) is determined by users. This is obvious in natural language, where words can change meanings through the years because they are used differently. To approximate this social aspect of meaning, Halpin and Lavrenko [?] propose relevance feedback as a way to col-

\textsuperscript{13}http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/KBs/
\textsuperscript{14}http://www.delicious.com/
\textsuperscript{15}http://www.flickr.com/
\textsuperscript{16}For instance, tags that often co-occur with the tag ‘cat’ on Delicious might be used to simulate the meaning of the word ‘cat’. 
lect data that indicates what meanings people assign to URIs. For example, in a semantic search engine, a query term (e.g. ‘dissicated coconut’) typed by a user can help construct the sense of the URI which the user found relevant.

This line of research is worth mentioning because it can point to further work within the area of formal ontologies and agent interaction. In particular, our work can be extended to take social meaning into account and throw more words into the ‘bags’ for ontology words. Since semantic search engines only work with URIs for now, we can use tags co-occurring with these words on sites like Delicious, where the power of collaborative tagging can give us an intuition as to how people assign meaning to words.

7 Conclusions

This paper briefly introduced our work on statistically approximating the meaning of words in formal ontologies in order to perform matching on the fly, whenever the need becomes apparent. We believe these ideas could be a major step forward in the problem of ontology matching in an agent communication environment, and in providing symbol grounding for ontology terms. Furthermore, they can provide a framework for the design of matchers which exploit the large amount of tag data available on the web. For instance, one of our future goals is to extend the Semantic Matcher so that it takes social meaning (i.e. how users conceptualise words) into account. To achieve this, large databases of tags such as Delicious or perhaps users’ input to semantic search engines will prove to be of central importance.

The Semantic Matcher was created on the basis of Information Retrieval principles, treating senses for words in the ontology as ‘bags of words’. This was not just an engineering decision but also a proposal for incorporating unordered sets of words into the semantics of formal ontologies in order to achieve symbol grounding.

Full details of the theory on which this work is based, together with full descriptions of the implementation and evaluation of our original work can be found in [Togia, 2010].

References


17or with part of these words; e.g. ‘cheesecake’ instead of ‘LemonCheesecakeDessert’