Data Min Knowl Disc (2007) 15:349–381
DOI 10.1007/s10618-007-0073-y

Tree-Traversing Ant Algorithm for term clustering based on featureless similarities

Wilson Wong · Wei Liu · Mohammed Bennamoun

Received: 20 February 2007 / Accepted: 13 April 2007 / Published online: 8 June 2007

Abstract Many conventional methods for concepts formation in ontology learning have relied on the use of predefined templates and rules, and static resources such as WordNet. Such approaches are not scalable, difficult to port between different domains and incapable of handling knowledge fluctuations. Their results are far from desirable, either. In this paper, we propose a new ant-based clustering algorithm, *Tree-Traversing Ant* (TTA), for concepts formation as part of an ontology learning system. With the help of *Normalized Google Distance (NGD)* and *n° of Wikipedia (n°W)* as measures for similarity and distance between terms, we attempt to achieve an adaptable clustering method that is highly scalable and portable across domains. Evaluations with an seven datasets show promising results with an average lexical overlap of 97% and ontological improvement of 48%. At the same time, the evaluations demonstrated several advantages that are not simultaneously present in standard ant-based and other conventional clustering methods.

Keywords Ontology learning · Text mining · Term clustering · Concept discovery · Cluster analysis · Featureless similarity measures

Responsible editor: M. J. Zaki.

W. Wong · W. Liu (✉) · M. Bennamoun
School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, WA 6009, Australia
e-mail: wei@csse.uwa.edu.au

W. Wong
e-mail: wilson@csse.uwa.edu.au

M. Bennamoun
e-mail: bennamou@csse.uwa.edu.au

1 Introduction

Ontology has become indispensable in modern information systems to provide inter-operable semantics. Increasing demand on ontologies makes labor-intensive creation more and more undesirable, if not impossible. Exacerbating the situation is the knowledge fluctuation resulted from the ever growing information source, both online and offline, which makes ontology maintenance even more difficult. Since the late nineties, more and more researchers started looking for solutions to relieve knowledge engineers from the increasingly acute situations. One of the main research thrusts with high potential of success is to construct and maintain ontology automatically or semi-automatically from text kept in electronic format. Ontology learning from text is the process of identifying concepts and relations from natural language text, and using them to construct and maintain an ontology. In ontology learning, terms are the lexical realizations of important concepts for characterizing a domain. Consequently, the task of grouping together variants of terms to form concepts and separating unrelated ones (known as *term clustering*) constitutes a crucial fundamental step in ontology learning (Fig. 1).

Unlike documents (Steinbach et al. 2000), webpages (Choi and yao 2005), and pixels in image segmentation and object recognition (Jain et al. 1999),

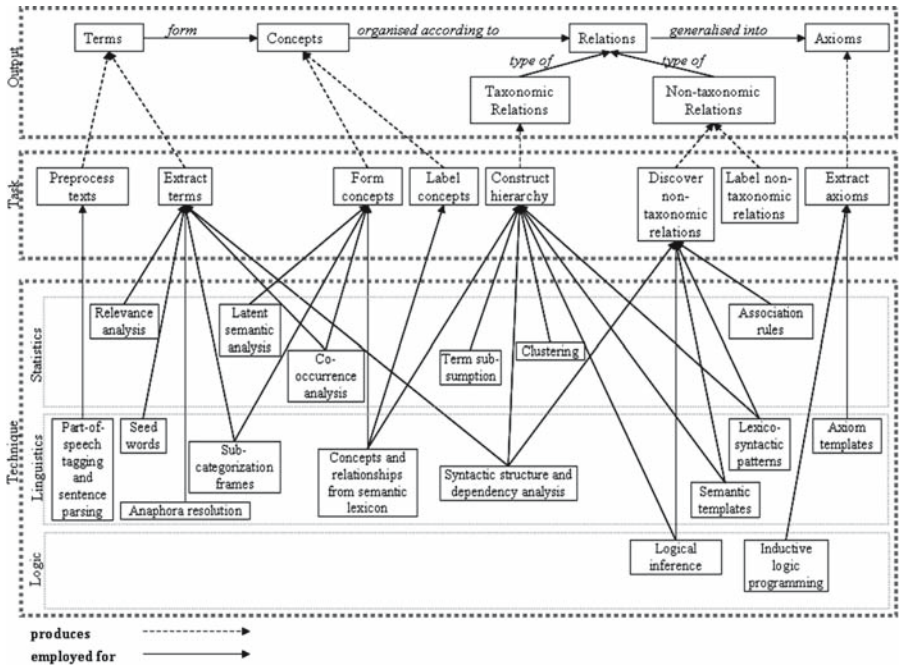

Fig. 1 Techniques, tasks, and output in ontology learning. Note that clustering has been largely employed for constructing hierarchies whereas concepts are formed using templates, patterns, co-occurrence analysis, and semantic lexicons

terms alone are lexically featureless. Similarity of objects can be established by feature analysis based on visible (e.g. physical and behavioral) traits. Unfortunately, using object names (i.e. terms) alone, similarity depends on something less tangible, namely, background knowledge which humans acquired through their senses over the years. The absence of features requires certain adjustments to be made with regard to the methods for term clustering. One of the most evident adaptation required is the use of context and other linguistic evidences as features for the computation of similarity. A recent survey in ontology learning (Gomez-Perez and Manzano-macho 2003) reveals that all reviewed systems which apply clustering methods rely on the contextual cues surrounding the terms as features. The large collection of documents, and predefined patterns and templates that are required for the extraction of contextual cues makes the portability of such ontology learning systems difficult. Consequently, non-feature similarity measures are fast becoming a necessity for term clustering, especially when there are no features to rely on. Along the same line of thought, Lagus et al. (1996) state *"In principle a document might be encoded as a histogram of its words... symbolic words as such retain no information of their relatedness"*. In addition to the problems associated with feature extraction in term clustering, much work is still required with respect to the clustering algorithm itself. Even though most conventional clustering algorithms appeared to be more established, researchers (Handl et al. 2003) have shown that certain commonly adopted algorithms such as K-means and average-link agglomerative yield mediocre results in comparison with the ant-based algorithms, which is a relatively new paradigm. Handl et al. (2003) demonstrated certain desirable properties in ant-based algorithms such as the tolerance to different cluster sizes, and the ability to identify the number of clusters. Despite such advantages, the potentials of ant-based algorithms remained relatively unexplored for possible applications in ontology learning.

In this paper, we propose to employ the well-established Normalized Google Distance (NGD) together with a new ant-based algorithm called *Tree-Traversing Ant* (TTA) for clustering terms in ontology learning. TTA was designed as an attempt to fuse the strengths of standard ant-based methods with certain advantages of conventional clustering methods. In addition, we introduce a second-pass for refining the results produced by the TTAs using *NGD*. During this second-pass, the TTAs will employ a new measure called *n° of Wikipedia (n°W)* for quantifying the distance between two terms based on the cross-linking of Wikipedia articles. Evaluations using seven datasets show promising results, and revealed several advantages which are not simultaneously present in existing clustering algorithms. In Sect. 2, we give an introduction to the existing techniques employed for term clustering in ontology learning. In Sect. 3, a summary of the mathematics behind the *NGD*, and the introduction to standard ant-based clustering and its enhancements to-date are presented. In Sect. 4, we present the TTA, and how *NGD* and *n°W* are employed to support term clustering. In Sect. 5, we summarize the results and findings from our evaluations. Finally, we conclude this paper with an outlook to future work in Sect. 6.

2 Existing techniques for term clustering

Faure and Nedellec (1998) presented a *corpus-based conceptual clustering* method as part of an ontology learning system called ASIUM. The clustering method is designed for aggregating basic classes based on a distance measure inspired by the Hamming distance. The basic classes are formed prior to clustering in a phase for extracting *sub-categorization frames* (Faure and Nedellec 2000). Terms that appear in at least two different occasions with the same verb, and the same preposition or syntactic role, can be regarded as semantically similar such that they can substituted with one another in a particular context. Such terms are extracted in the form of selection restriction as part of the sub-categorization frames. These semantically similar terms will form the basic classes. The basic classes form the lowest level of the ontology and are successively aggregated to construct a hierarchy from bottom-up. Each time, only two basic classes are compared. The clustering begins by computing the distance between all pairs of basic classes and aggregate those with distance less than the user-defined threshold. The distance between two classes containing the same words with the same frequencies is 0. On the other hand, a pair of classes without a single common word have a distance 1. In other words, the terms in the basic classes act as features, allowing for inter-class comparison. The measure for distance is defined as

$$distance(C_1, C_2) = 1 - \left(\frac{\sum FC_1 \times \frac{N_{comm}}{card(C_1)} + \sum FC_2 \times \frac{N_{comm}}{card(C_2)}}{\sum_{i=1}^{card(C_1)} f(word_{iC_1}) + \sum_{i=1}^{card(C_2)} f(word_{iC_2})} \right)$$

where $card(C_1)$ and $card(C_2)$ is the number of words in (i.e. the size of) C_1 and C_2 respectively, and N_{comm} is the number of words common to both C_1 and C_2. $\sum FC_1$ and $\sum FC_2$ is the sum of frequencies of the words in C_1 and C_2 which also occur in C_2 and C_1, respectively. $f(word_{iC_1})$ and $f(word_{iC_2})$ is the frequency of the ith word of class C_1 and C_2, respectively.

Maedche and Volz (2001) presented a *bottom-up hierarchical clustering* method that functions as part of the ontology learning system Text-to-Onto. This clustering method, designed for clustering terms, rely on an all-knowing *oracle*, denoted by H, which is capable of returning possible hypernyms for a given term. In other words, the performance of the clustering algorithm has an upper-bound limited by the ability of the oracle to know all possible hypernyms for a term. The oracle is constructed with the help of WordNet and *lexico-syntactic patterns* (Cimiano and Stab 2005). During the clustering phase, the algorithm would be fed with a list of terms and the similarity between each pair is computed using a cosine measure. For this purpose, the syntactic surface dependencies of each term, in the form of modifiers of various types of phrases, are extracted and used as the features for that term. The algorithm is an extremely long list of nested `if-else` statements. For the sake of brevity, it suffices to know that the algorithm attempts to examine the hypernym–hyponym relations between all pairs of terms as it decides to place a term as a hypernym

of the other or vise versa, or places two terms under a new parent. Each time information about the hypernymy relation between two terms is required, the oracle is consulted. The projection $H(t)$ will return a set of tuples (x, y) where x is a hypernym for term t and y is the number of times the algorithm has found evidence for it.

Shamsfard and Barforoush (2002) presented two clustering algorithms as part of the ontology learning system Hasti. Concepts have to be formed prior to the clustering phase. It suffices to know that the process of creating the concepts and extracting relations that are used as features for clustering involve a knowledge extractor where *"the knowledge extractor is a combination of logical, template driven and semantic analysis methods"* (Shamsfard and Barforoush 2004). In *concept-based clustering*, a similarity matrix, consisting of the similarity for all possible pairs of concepts is computed. Then the pair with the maximum similarity, also greater than the merge-threshold, will be chosen to form a new super concept. In this method, each intermediate (i.e. non-leaf) node in the conceptual hierarchy has at most two children, but the hierarchy is not a binary tree as each node may have more than one father. As for *relation-based clustering*, only non-taxonomic relations are considered. For every concept c, a set of assertions about the non-taxonomic relations $NF(c)$ that c has with other concepts is identified. In other words, these relations can be seen as features allowing for the concepts to be merged according to what they share. If at least one related concept is common between assertions about that relation, then the set of other concepts (called merge-set) will be good candidates for merging. After all the relations have been examined, a list of merge-set is obtained. The merge-set with the highest similarity between its members is chosen for merging. In both clustering algorithms, semantic similarity is employed and is defined as

$$similarity(a,b) = \sum_{j=1}^{maxlevel} \left(\sum_{i=1}^{card(cm)} \left(W_{cm(i).r} + \sum_{k=1}^{valence(cm(i).r)} W_{cm(i).arg(k)} \right) \right) \times L_j$$

where $cm = Nf(a) \cap Nf(b)$ is the intersection between the sets of assertions (i.e. common relations) about a and b, and *card(cm)* is the cardinality of cm. $W_{cm(i).r}$ is the weight for each common relation and $\sum_{k=1}^{valence(cm(i).r)} W_{cm(i).arg(k)}$ is the sum of the weights of all terms related to the common relations cm. L_j is the level constant assigned to each similarity level which decreases as the level increases. The main aspect of the similarity measure is the common features between two concepts a and b. The common features are the ones in the intersection between the set of non-taxonomic assertions, $Nf(a) \cap Nf(b)$ between the two concepts. Each common feature $cm(i).r$ together with the corresponding weight $W_{cm(i).r}$ and the weight of the related terms will be accumulated. In other words, the more features two concepts have in common, the higher the similarity between them.

Regardless of how the existing techniques described in this section are named, they shared a common point, namely, the reliance on some linguistic

(e.g. subcategorisation frames, lexico-syntactic patterns) or predefined semantic (e.g. WordNet) resources as features. These features are necessary for the computation of similarity using conventional measures and clustering algorithms. The ease of scalability and portability across domains, and the resources required for feature extraction are among the few questions our new clustering technique attempts to overcome. In addition, the new clustering technique fuses the strengths of recent innovations such as ant-based algorithms and the notion of featureless similarity measures that have yet to benefit the techniques in ontology learning.

3 Background

3.1 Normalized Google Distance

Normalized Google Distance (NGD) computes the similarity between objects based on their names using only the Google search engine. *NGD* is a non-feature similarity which attempts to capture every effective distance (e.g. Hamming distance, Euclidean distance, and edit distances) into a single metric. *NGD* combines the notion of Kolmogorov Complexity (Grunwald and vitanyi 2003), Shannon-Fano coding (Lelewer and Hirschberg 1987) and the Google search engine.

The basis of *NGD* begins with the idea of the shortest binary program that is capable of producing a string x as output. Kolmogorov Complexity of the string x, $K(x)$ is just the length, in binary bits, of that program. Extending this notion to include an additional string y, we will have the Information Distance (Bennett et al. 1998) where $E(x, y)$ is the length of the shortest binary program that can produce x given y, and y given x. It was shown (Bennett et al. 1998) that

$$E(x, y) = K(x, y) - min\{K(x), K(y)\} \tag{1}$$

where $E(x, x) = 0, E(x, y) > 0$ for $x \neq y$, and $E(x, y) = E(y, x)$. Next, for every other computable distances D that are non-negative and symmetric, there will be a binary program, given string x and y, which has length (in binary bits) equal to $D(x, y)$. Formally,

$$E(x, y) \leq D(x, y) + c_D$$

where c_D is a constant that depends on the distance D but not x and y. $E(x, y)$ is called universal because it acts as the lower bound for all computable distances. In other words, if two strings x and y are close according to some distance D, then they are at least as close according to E (Cilibrasi and Vitanyi 2005). Since all computable distances compare the *closeness* of strings through the quantification of certain features that the strings have in common, we can consider that information distance determines the distance between two strings according to the feature in which they are most similar.

By normalizing information distance, we would have $NID(x, y) \in (0, 1)$ where 0 means the two strings are the same and 1 being completely different in the sense that they share no features. The normalized information distance is defined as:

$$NID(x, y) = \frac{K(x, y) - min\{K(x), K(y)\}}{max\{K(x), K(y)\}}$$

Nonetheless, referring back to Kolmogorov Complexity and Eq. 1, the non-computability of $K(x)$ inherently results in the non-computability of $NID(x, y)$. Fortunately, an approximation of K can be achieved using real compression programs (Vitanyi 2005). If C is a compressor, then $C(x)$ denotes the length of the compressed version of string x. Approximating $K(x)$ with $C(x)$ will result in:

$$NCD(x, y) = \frac{C(x, y) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}}$$

The derivation of *NGD* continues by observing the working behind compressors. Compressors encode source words x into code words x' such that the length $|x| < |x'|$. We can consider these code words from the perspective of Shannon-Fano coding. Shannon-Fano coding encodes a source word x using a code word that has the length $\log \frac{1}{p(x)}$. $p(x)$ can be thought of as a probability mass function that maps each source word x to the code that achieves optimal compression of x. In Shannon-Fano coding, $p(x) = \frac{n_x}{N}$ captures the probability of encountering source word x in a text or a stream of data from a source, where n_x is the occurrence of x and N is the total number of source words in the same text. Cilibrasi and Vitanyi (2005) extensively discussed the use of compressors for *NCD* and concluded that the inability of these compressors to take into consideration external knowledge during compression makes them inadequate. Instead, the authors proposed to make use of a source that "... *stands out as the most inclusive summary of statistical information*" (Cilibrasi and Vitanyi 2005), namely, the World Wide Web. More specifically, the authors proposed the use of the Google search engine to devise a probability mass function that reflects knowledge and constructs a Shannon-Fano code alike. It appears that the Google's equivalence of Shannon-Fano code, known as *Google code*, has length defined by Cilibrasi and Vitanyi (2006)

$$G(x) = \log \frac{1}{g(x)}$$

$$G(x, y) = \log \frac{1}{g(x, y)}$$

where $g(x) = |\mathbf{x}|/N$ and $g(x, y) = |\mathbf{x} \cap \mathbf{y}|/N$ are the new probability mass function that capture the probability of occurrences of search terms x and y. $\mathbf{x}$ is the

set of web pages returned by Google containing the single search term x (i.e. singleton set) and similarly, $\mathbf{x} \cap \mathbf{y}$ is the set of web pages returned by Google containing both search term x and y (i.e. doubleton set). N is the summation of all unique singleton and doubleton sets.

Consequently, the Google search engine can be considered as a compressor for encoding search terms (i.e. source words) x to produce the *meaning* (i.e. compressed code words) that has the length $G(x)$. By rewriting the *NCD*, we will obtain the new *NGD* defined as:

$$NGD(x, y) = \frac{G(x, y) - min\{G(x), G(y)\}}{max\{G(x), G(y)\}} \tag{2}$$

In a nutshell, *NGD* is an approximation of *NCD* and hence *NID* to overcome the non-computability of Kolmogorov Complexity by employing the Google search engine as a compressor to generate Google code based on the Shannon-Fano coding. From the perspective of term clustering, *NGD* provides an innovative starting point to demonstrate the advantages that featureless similarity measures can offer. In our new term clustering technique, we take such an innovation one step further by employing *NGD* in a new clustering technique that combines the strengths from both conventional and ant-based algorithms.

3.2 Ant-based clustering

The idea of ant-based clustering was first proposed by Deneubourg et al. (1991) in 1991 as part of an attempt to explain the different types of emergent technologies inspired by nature. During simulation, the ants are represented as agents that move around the environment, a square grid, in random. Objects are randomly placed in this environment and the ants can pick-up the object, move and drop them. These three basic operations are influenced by the distribution of the objects. Objects that are surrounded or isolated by dissimilar ones are more likely to be picked up and are later dropped elsewhere in the surrounding of more similar ones. The probability of picking up and dropping of objects are influenced by the probabilities:

$$P_{pick}(i) = \left(\frac{k_p}{k_p + f(i)}\right)^2$$

$$P_{drop}(i) = \left(\frac{f(i)}{k_d + f(i)}\right)^2$$

where $f(i)$ is an estimation of the distribution density of the objects in the ants' immediate environment (i.e. local neighborhood) with respect to the object that the ants is considering to pick or drop. The choice of $f(i)$ varies depending on the cost and other factors related to the environment and the data items. As $f(i)$

decreases below k_p, the probability of picking up the object is very high, and the opposite occurs when $f(i)$ exceeds k_p. As for the probability of dropping an object, high $f(i)$ exceeding k_d will induce the ants to give up the object while low $f(i)$, which is less than k_d, will encourage the ants to hold onto the object. The combination of these three simple operations and the heuristics behind them gave birth to the notion of basic ants for clustering, also known as *standard ant clustering algorithm* (*SACA*).

Gutowitz (1993) examined the basic ants described by Deneubourg et al. and proposed a variant ant known as *complexity-seeking ants*. Such ants are capable of sensing local complexity and are inclined to work in regions of high interest (i.e. high complexity). Regions with high complexity are determined using a local measure that assesses the neighboring cells and counts the number of pairs of contrasting cells (i.e. occupied or empty). Neighborhoods with all empty or all occupied immediate cells have zero complexity while regions with checkboard patterns have high complexity. Hence, these modified ants are able to accomplish their task faster because they are more inclined to manipulate objects in regions with higher complexity (Vizine et al. 2005).

Lumer and Faieta (1994) further extended and improved the idea of ant-based clustering in terms of the *numerical aspects of the algorithm and the convergence time*. The authors represented the objects in terms of numerical vectors and the distance between the vectors is computed using Euclidean distance. Hence, given that $\delta(i,j) \in [0,1]$ as the Euclidean distance between object i (i.e. i is the location of the object in the center of the neighborhood) and every other neighboring objects j, the neighborhood function $f(i)$ is defined by the authors as:

$$f(i) = \begin{cases} \frac{1}{s^2} \sum_j 1 - \frac{\delta(i,j)}{\alpha} & \text{if } f(i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where s^2 is the size of the local neighborhood, and $\alpha \in [0,1]$ is the constant for scaling the distance among objects. In other words, an ant will have to consider the average similarity of object i with respect to all other objects j in the local neighborhood before performing an operation (i.e. pickup or drop). As the value of $f(i)$ is obtained by averaging the total similarities with the number of neighboring cells, s^2, empty cells which do not contribute to the overall similarity must be penalized. In addition, the radius of perception (i.e. the extent to which objects are taken into consideration for $f(i)$) of each ant at the center of the local neighborhood is given by $\frac{s-1}{2}$. The algorithm behind clustering using the basic ant *SACA* is defined in Algorithm 1.

Handl and Meyer introduced several enhancements to make ant-based clustering more efficient. The first was the concept of *eager ants* where idle phases are avoided by having the ants to immediately pickup objects as soon as existing ones are dropped. The second was the notion of *stagnant control*. There are occasions in ant-based clustering when ants are occupied or blocked due to objects that are difficult to dispose. In such cases, the ants will be forced to drop

Algorithm 1 Basic ant-based clustering defined by Handl et al. (2006)

```
1: begin
2: INITIALIZATION PHASE
3: Randomly scatter data items on the toroidal grid
4: for each j in 1 to #agents do
5:    i := random_select (remaining_items)
6:    pick_up(agent(j), i)
7:    g := random_select (remaining_empty_grid_locations)
8:    place_agent (agent(j), g)
9: end for
10: MAIN LOOP
11: for each it_ctr in 1 to #iterations do
12:    j := random_select (all_agents)
13:    step(agent(j), stepsize)
14:    i := carried_item (agent(j))
15:    drop := drop_item? (f(i))
16:    if drop = TRUE then
17:       while pick = FALSE do
18:          i := random_select (free_data_items)
19:          pick := pick_item? (f(i))
20:       end while
21:    end if
22: end for
```

whatever they are carrying after a certain number of unsuccessful drops. In a different paper (Handl et al. 2003), the authors have also demonstrated that the ant-based algorithm triumphs in several aspects:

- tolerance to different cluster size
- ability to identify the number of clusters
- performance increases with the size of the datasets
- graceful degradation in the face of overlapping clusters.

Nonetheless, the authors have also highlighted two shortcomings of ant-based clustering, namely, the inability to distinguish more refined clusters within coarser level ones, and the inability to specify the number of clusters can be seen as a disadvantage when the users have precise ideas about it.

Vizine et al. (2005) proposed an *adaptive ant clustering algorithm* (A^2CA) that improves upon the algorithm by Lumer and Faieta. The authors introduced two major modifications, namely, *progressive vision scheme* and the *use of pheromones* on grid cells. The progressive vision scheme allows the dynamic adjustment of s^2. Whenever an ant perceives a larger cluster, it will increase its radius of perception from the original $\frac{s-1}{2}$ to the new $\frac{s'-1}{2}$. The second enhancement allows ants to mark regions that are recently constructed or under construction. The pheromones will attract other ants resulting in increases of probability of deconstruction of relatively smaller regions, and increases the probability of dropping objects at denser clusters.

Ant-based algorithms have been employed to cluster objects that can be represented using numerical vectors. Similar to conventional algorithms, the similarity or distance measures used by existing ant-based algorithms are still

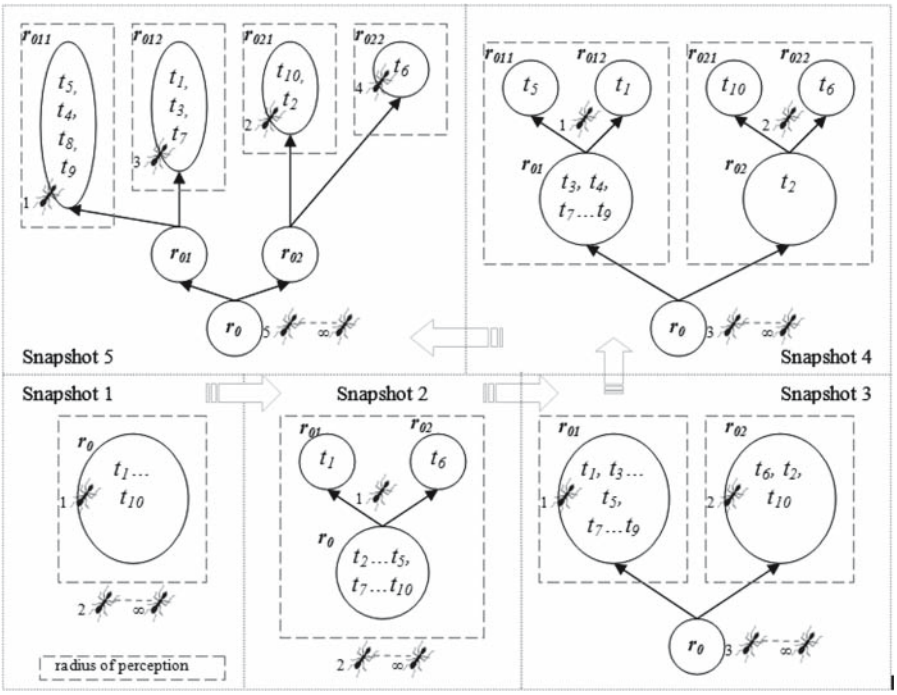

Fig. 2 Example of TTA at work

feature-based. Consequently, they share similar problems like portability across domains. In addition, despite the strengths of standard ant-based algorithms, two disadvantages have been identified. In our new technique, we make use of the known strengths of standard ant-based algorithms and some desirable traits from conventional ones for clustering terms using featureless similarity.

4 The proposed Tree-Traversing Ants

Unlike the standard ant-based clustering where the ants walk on a toroidal grid, the structure employed for clustering by the TTAs are a dynamic tree. The dynamic tree begins with one root node r_0 consisting of all terms $T = \{t_1, \ldots, t_n\}$, and will further branch out to new sub-nodes as required. In other words, we begin with $r_0 = \{t_1, \ldots, t_n\}$. For example, the first snapshot in Fig. 2 shows the start of the algorithm with the root node r_0 initialized with the terms $t_1, \ldots, t_{n=10}$. Essentially, each node in the tree is a set of terms $r_u = \{t_1, \ldots, t_q\}$. The size of each new sub-nodes $|r_u|$ reduces as less and less terms are assigned to them in the process of creating nodes with higher intra-node similarity.

The processing starts with only one ant, while a theoretically infinite number of ants awaits to *work* at each of the new sub-node created for each node. In the third snapshot in Fig. 2, while the original ant moves on to work at the left

sub-node r_{01}, a new ant proceeds to process the right sub-node r_{02}. The exact number of new sub-nodes that can be *grown* for each main node (i.e branching factor) in this version of TTA is two. In other words, for each main node r_m, we have got the sub-nodes r_{m1}, r_{m2}. As the TTAs are no different from the standard ants in terms of the number of objects that they can carry at any given time and the inability for direct communication, pheromones may be required for leaving behind trails and as a positive feedback mechanism from its environment. Similar to some of the enhanced ants, TTAs too are endowed with the ability of *short-term memory* for remembering similarities and distances acquired through its senses. The TTAs are equipped with two types of senses, namely, *NGD* and *n° of Wikipedia (n°W)*.

The standard ants has a radius of perception defined in terms of cells immediately surrounding the ants. The radius of perception of TTAs covers all terms in the two sub-nodes created for each main node. The main node is defined as the node originally consisting of terms to be sorted to the two sub-nodes. It is also worth pointing out that the vision of the TTAs changes as they move from the main node to any of the sub-nodes for further processing once the main node runs out of terms. In such cases, the sub-node that the ant has moved to will become the main node, and so on. For example, in the second and third snapshot of Fig. 2, *ant*1 moves from r_0 to the new left sub-node r_{01} once the root node has run out of terms. While *ant*1 was on r_0 (as in snapshot 2), its vision (denoted by the dotted box) covers both r_{01} and r_{02}. Once *ant*1 has moved to the r_{01} (as in snapshot 3 and 4), its radius of perception has changed to the subsequent two new sub-nodes r_{011} and r_{012}, created for the main node r_{01}.

The TTAs work in a two-pass approach for term clustering. During the first-pass, the TTAs recursively break nodes into sub-nodes and relocate terms until the *ideal* clusters are achieved. The resulting tree of clusters created in the first-pass are often good enough to reflect the natural clusters. Nonetheless, discrepancies do occur, not due to the TTAs, but rather due to certain oddities in the co-occurrences of terms on the World Wide Web that manifest itself through *NGD*. Accordingly, a second-pass is proposed for relocating terms which are misplaced due to *NGD*. *n°W* is employed for this purpose. The second-pass can be regarded as a refinement phase for producing clusters with higher quality.

4.1 First-pass using Normalized Google Distance

The proposed clustering algorithm begins processing at the root node which consists of all n terms $r_0 = \{t_1, \ldots, t_n\}$. Each term can be considered as an element in the node. The TTA will randomly pick a term, and proceed on to *sense* its similarity with every other terms on that same node. The TTA will repeat this for all n terms until the similarity of all possible pair of terms have been *memorized*. The similarity between two terms t_x and t_y is defined as:

$$s(t_x, t_y) = 1 - NGD(t_x, t_y)\alpha \tag{4}$$

where $NGD(t_x, t_y)$ is the distance between term t_x and t_y estimated using the original NGD defined at 2. α is a constant for scaling the distance between the two terms. The algorithm will grow two new sub-nodes to accommodate the two least similar terms t_a and t_b. The TTA will move the first term t_a from the main node r_m to the first sub-node while emitting pheromones that trace back to t_b in the process. The TTA will then follow the trail pheromones back to the second term t_b to move it to the second sub-node.

The second snapshot in Fig. 2 shows two new sub-nodes r_{01} and r_{02} with the ant having moved the term t_1 to r_1 and the least similar term t_6 to r_{02}. Nonetheless, prior to the creation of the new sub-nodes and the moving of the two least similar terms to the new sub-nodes, an *ideal intra-node similarity condition* must be tested. The operation of moving the two least similar terms from the main node to create and initialize new sub-nodes can be regarded as a partitioning process. Eventually, each leaf node will end up with only one term if the TTAs do not know when to stop. For this reason, we adopt an *ideal intra-node similarity threshold* s_T for controlling the extent of branching out. Whenever a TTA senses that the similarity between the two least similar term exceeds s_T, no further sub-nodes will be created and the current main node will be left as it is. A high similarity (higher than s_T) between the two most dissimilar terms in the current main node provide a simple but effective indication that the intra-node similarity has reached an ideal stage. More refined factors such as the mean and standard deviation of intra-node similarity are possible but have not been considered here due to time constraint.

If the similarity between the two most dissimilar terms is still less than s_T, further branching out will be performed. The TTA will repeatedly picks up the remaining terms on the current main node one by one and senses their similarities with every other terms which are already located in the sub-nodes. Formally, the probability of picking up term t_i by the TTA in the first-pass is defined as:

$$P^1_{pick}(t_i) = \begin{cases} 1 & \text{if } t_i \in r_m \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where r_m is the set of terms in the main node. In other words, the probability of picking up terms by the TTAs is always 1 as long as there are still terms remaining in the main node.

Each sub-node r_u of the main node r_m can be seen as a neighborhood in which a TTA senses for the similarities between the term t_i it picked up from r_m with every other terms $t_j \in r_u$. Each term $t_i \in r_m$ will be moved to neighborhood r_u that has the term $t_j \in r_u$ with the highest similarity with t_i. In other words, a TTA considers multiple neighborhoods prior to dropping a term. Snapshot 3 in Fig. 2 illustrates the corresponding two sub-nodes r_{01} and r_{02} that have been filled with all the terms which are used to be located at the main node r_0. The standard neighborhood function $f(i)$ defined in Eq. 3 represents the density of the neighborhood as the average of the similarities between t_i with every other term in its immediate surrounding (i.e. local neighborhood) confined by s^2.

Unlike the senses of basic ants which cover the surrounding cells s^2, the extent to which TTAs perceive cover all terms in two sub-nodes (i.e. multiple neighborhoods) corresponding to the immediate main node. Accordingly, instead of estimating $f(i)$ as the averaged similarity defined over s^2 terms surrounding the ant, the new neighborhood function $f_{TTA}(t_i, u)$ is defined as the maximum similarity between term $t_i \in r_m$ and the neighborhood (i.e. sub-nodes) r_u. The maximum similarity between t_i and r_u is the highest similarity between t_i and all other terms $t_j \in r_u$. Formally, we define the density of neighborhood r_u with respect to term t_i during the first-pass as:

$$f^1_{TTA}(t_i, r_u) = \text{maximum of } s(t_i, t_j) \text{ w.r.t } t_j \in r_u \tag{6}$$

where the similarity between the two terms $s(t_i, t_j)$ is computed using Eq. 4.

Besides deciding on whether to drop an object or not, like in the case of basic ants, TTAs have to decide on one additional issue, namely, where to drop. A TTA will decide on where to drop a term based on the $f_{TTA}(t_i, r_u)$ that it has memorized for all sub-nodes r_u of the main node r_m. Formally, the decision on whether to drop term $t_i \in r_m$ on sub-node r_v depends on:

$$P^1_{drop}(t_i, r_v) = \begin{cases} 1 & \text{if } (f^1_{TTA}(t_i, r_v) = \text{maximum of } f^1_{TTA}(t_i, r_u) \\ & \text{w.r.t. } r_u \in \{r_{m1}, r_{m2}\}) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The current version of the algorithm for clustering using TTAs is implemented in two parts, one as the main function while the other one is a recursive function. The main function is defined in Algorithm 2 while the recursive function for the first-pass elaborated in this subsection is reported in Algorithm 3.

Algorithm 2 Main function

```
1: input A list of terms, T = {t1, ...tn}.
2: Create an initial tree with a root node r0 containing n terms.
3: Define the ideal intra-node similarity threshold sT and δT.
4: //first-pass using NGD
5: ant := new_ant()
6: ant.ant_traverse(r0, r0)
7: //second-pass using n°W
8: leafnodes := ant.pickup_trail()//return all leaf nodes marked by pheromones
9: for each r_next ∈ leafnodes do
10:    ant.ant_refine(leafnodes, r_next)
11: end for
```

4.2 n° of Wikipedia: a new distance metric

The use of *NGD* for quantifying the similarity between two objects in terms of their names is by no means perfect. We will highlight certain discrepancies that

Algorithm 3 Function ant_traverse(r_m, r_0) using NGD

if $|r_m| = 1$ **then**
  leave_trail(r_m, r_0)//leave trail from current leave node to root node. for use in second-pass
  return //only one term left. return to root
end if
$\{t_a, t_b\}$:= find_most_dissimilar_terms(r_m)
if $s(t_a, t_b) > s_T$ **then**
  leave_trail(r_m, r_0)//leave trail from current leave node to root node. for use in second-pass
  return //ideal cluster has been achieved. return to root node
else
  $\{r_{m1}, r_{m2}\}$:= grow_sub_nodes(r_m)
  move_terms($\{t_a, t_b\}$, $\{r_{m1}, r_{m2}\}$)
  for each term $t_i \in r_m$ **do**
    pick(t_i)//based on Eq. 5
    for each $r_u \in \{r_{m1}, r_{m2}\}$ **do**
      for each term $t_j \in r_u$ **do**
        $s(t_i, t_j)$:= sense_similarity(t_i, t_j) //based on Eq. 4
        remember_similarity($s(t_i, t_j)$)
      end for
      $f^1_{TTA}(t_i, r_u)$:= sense_neighborhood() //based on Eq. 6
      remember_neighbourhood($f^1_{TTA}(t_i, r_u)$)
    end for
    $\{\forall u, f^1_{TTA}(t_i, r_u)\}$:= recall_neighborhood()
    r_v := decide_drop($\{\forall u, f^1_{TTA}(t_i, r_u)\}$)// based on Eq. 7
    drop($\{t_i\}$, $\{r_v\}$)
  end for
end if
ant_{m1} := new_ant()
ant_{m1}.ant_traverse(r_{m1}, r_0)//repeat the process recursively for each sub-node
ant_{m2} := new_ant()
ant_{m2}.ant_traverse(r_{m2}, r_0)//repeat the process recursively for each sub-node

surfaced during our initial experiments in the following section. The initial tree of clusters generated by the TTAs using *NGD* demonstrated promising results. Nonetheless, we reckoned that better results could be generated if we would allow the TTAs to visit the nodes again for the purpose of refinement. Instead of using *NGD*, we proposed a different way to gauge the similarity between terms.

Google can be regarded as the gateway to the huge volume of documents on the World Wide Web. The sheer size of Google index has enables the *NGD* to reliably estimate the usage of terms in the society. The page count provided by Google is the essence of *NGD* in the attempt to compute the similarity between two terms based on the mutual information that they both share at the compressed level. Other than the frequency of occurrences of terms, it is rather difficult to reliably make use of the other aspects of documents on the World Wide Web through Google search engine. There is no restriction on the topic and the content of each document, and the hyperlinks between them do not adhere to any particular categorical indices. As for Wikipedia, its number of articles is less than a fraction of what Google indexes. Nonetheless, the restrictions imposed on the authoring of articles and the organization of its

articles provide us with a possibly new way of looking at similarity between terms.

n° of Wikipedia (n°W) (Wong et al. 2006) is inspired by a game for *Wikipedians*. *6° of Wikipedia*[1] is a task set out to study the character of Wikipedia in terms of the similarity between its articles. An article in Wikipedia can be regarded as an entry of encyclopedic information describing a particular topic. The articles are organized using categorical indices which eventually leads to the highest level, namely, *"Categories"*.[2] Each article can appear under more than one category, making the organization of articles in Wikipedia to appear more as a directed acyclic graph with a root node, instead of a pure tree structure.[3] The huge volume of articles in Wikipedia, the organization of articles in a graph structure, the open-source nature of its articles, and the availability of the articles in electronic form makes it the ideal candidate for our endeavor.

We define Wikipedia as a directed graph $W := (V, E)$. W is essentially a network of linked-articles where $V = \{a_1, \ldots, a_\omega\}$ is the set of articles. We limit the vertices to English articles. At the moment, $\omega = |V|$ is reported to be $1,384,729$,[4] making it the largest encyclopedia[5] in merely five years since its conception. The inter-connections between articles are represented as the set of ordered pairs of vertices E. At the moment, the edges are uniformly assigned with weight 1. Each article can be considered as an elaboration of a particular event, entity or abstract idea. In other words, an article in Wikipedia can be regarded as a manifestation of the information encoded in the terms. Consequently, we can represent each term t_i using the corresponding article a_i in Wikipedia. Hence, finding the distance between two terms t_i,t_j can be reduced to the discovery of how closely situated are the two corresponding articles a_i,a_j in the Wikipedia categorical indices. The problem of finding the *degree of separation* between two articles can be addressed in terms of the single-source shortest path problem. Since the weights are all positive, we have resorted to Dijkstra's Algorithm for finding the shortest-path between two vertices (i.e. articles). The benefits of various other existing algorithms for the shortest-path problem have yet to be scrutinized and considered due to time-constraint. Formally, the distance between term t_x and t_y given by $n°W$ is defined as

$$\delta(t_x, t_y) = n°W(a_x, a_y) = \sum_{k=1}^{|SP|} c_{e_k} \tag{8}$$

where $n°W(a_x, a_y)$ is the *degree of separation* between the articles a_x and a_y which corresponds to the term t_x and t_y, respectively. The *degree of separation* is computed as the sum of the cost of all edges along the shortest path between

[1] http://en.wikipedia.org/wiki/Six_Degrees_of_Wikipedia

[2] http://en.wikipedia.org/wiki/Category:Categories

[3] http://en.wikipedia.org/wiki/Wikipedia:Categorization#Categories_do_not_form_a_tree

[4] http://en.wikipedia.org/wiki/Wikipedia:Size_comparisons

[5] http://en.wikipedia.org/wiki/Wikipedia:Largest_encyclopedia

articles a_x and a_y in the graph of Wikipedia articles W. SP is the set of edges along the shortest path and e_k is the kth edge or element in set SP. $|SP|$ is the number of edges along the shortest path and c_{e_k} is the cost associated with the kth edge. It is also worth pointing out that $\delta(t_x, t_y) \geq 1$ for $t_x \neq t_y$ but no upper bound can be ascertained. Although there is a hypothesis stating[6] that no two articles in Wikipedia are separated by more than six degrees, no certainties can be made as there are some Wikipedians who have shown that certain articles can be separated by up to eight steps.[7] This is the reason why we adopted the name *n° of Wikipedia* instead of *6° of Wikipedia*.

4.3 Second-pass using n° of Wikipedia

At the end of the first-pass of creating the sub-nodes, there will be at most n leaf nodes where each term in the initial set of all terms T end up in individual nodes (i.e. clusters). There are only two possibilities for such extreme cases. The first is when the *ideal intra-node similarity threshold* s_T is set too high while the second is when all the terms are extremely unrelated. In normal cases, most of the terms will be nicely grouped into nodes with intra-node similarities exceeding s_T. Only a small number of terms will be isolated into individual nodes. We refer to these terms as *isolated terms*. There are two possibilities of isolated terms in normal cases, one is that the term has been displaced during the first-pass due to discrepancies related to *NGD*, or the term is in fact an outlier. During the returning of the TTAs to the root node at the end of the first-pass (as in line 2 and line 7 of Algorithm 3), trail pheromones will be released to mark the path from root node to leaf nodes. For the purpose of relocating the isolated terms to possibly more suitable nodes, one TTA will be allocated to work in the second-pass. The single TTA will jump to the leaf nodes following the pheromone trails. At each leaf node r_l, the probability of picking up a term t_i during the second-pass is 1 if the leaf node has only one term (isolated term):

$$P^2_{pick}(t_i) = \begin{cases} 1 & \text{if } |r_l| = 1 \wedge t_i \in r_l \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

After picking up an isolated term, the single TTA will continue to jump from one leaf node to the next. In each leaf node, the TTA will determine whether that particular leaf node (i.e. neighborhood) r_l is the most suitable one to house the isolated term t_i it is carrying based on the average distance between t_i and all other existing terms in r_l. Formally, the density of neighborhood r_l with respect to the isolated term t_i during the second-pass is defined as:

$$f^2_{TTA}(t_i, r_l) = \frac{\sum_{j=1}^{|r_l|} \delta(t_i, t_j)}{|r_l|} \tag{10}$$

[6] http://tools.wikimedia.de/sixdeg/index.jsp

[7] http://en.wikipedia.org/wiki/Six_Degrees_of_Wikipedia

where $|r_l|$ is the number of terms in the leaf node r_l and the distance between the two terms t_i and t_j is computed using Eq. 8.

This process of sensing the distance of the isolated term with all other terms in each leaf node is carried out for all leaf nodes. The probability of the TTA dropping the isolated term t_i on the most suitable leaf node r_v is evaluated once the TTA returns to the original leaf node of t_i. Back at the original leaf node of t_i, the TTA will recall the neighborhood density $f^2_{TTA}(t_i, r_l)$ that it has memorized for all neighborhoods (i.e leaf nodes). In the case of t_i being an outlier, it will have very high average distances with all other leaf nodes. Without a threshold, the TTA will still relocate the outlier to the leaf node that has the minimum average distance, which by comparison with all other cases of non-outliers, is still considered extremely high. In short, the TTA will drop the isolated term t_i which it is carrying on the leaf node r_v if all terms in r_v collectively yields the minimum average distance with t_i that satisfies the *outlier discrimination threshold* δ_T. Formally,

$$P^2_{drop}(t_i, r_v) = \begin{cases} 1 & \text{if } (f^2_{TTA}(t_i, r_v) = \text{minimum of } f^2_{TTA}(t_i, r_l) \text{ w.r.t. } r_l \in L) \\ & \wedge\ (f^2_{TTA}(t_i, r_v) \leq \delta_T) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where L is the set of all leaf nodes.

Unlike the first-pass where the number of TTAs employed increases at the same rate as the creation of sub-nodes, we only need one TTA in the second-pass. Multiple TTAs revisiting the leaf nodes will introduce problems related to concurrency control such as the communication between TTAs and the possibility of conflict due to the sharing of the same address space (i.e. tree structure). For example, assume that we have an isolated term t_1 about to be relocated by a TTA to a better leaf node that has the least average distance. At the same time, another TTA is already on its way to drop a different isolated term t_2 in the same node as t_1. What will happen when the second TTA arrived at the leaf node that used to contain t_1 but discovered that it is now empty? The use of multiple TTAs will result in such predicaments that will eventually lead to a sub-optimal condition in the second-pass when all the TTAs, each holding an isolated term, never gets to drop them. After the TTA has visited all the leaf nodes and has failed to drop the isolated term, the term will be returned to its original location. The failure to drop the isolated term in a more suitable node indicates that it is an outlier.

Referring back to the example in Fig. 2, let's assume that snapshot 5 represents the end of the first-pass where the intra-node similarity of all nodes have satisfied s_T. While all other leaf nodes, namely r_{011}, r_{012}, and r_{021} consist of more than one term, leaf node r_{022} contains only one term t_6. Hence, at the end of the first-pass, all TTAs, namely, *ant*1, *ant*2, *ant*3 and *ant*4 will all retreat back to the root node r_0. For the purpose of the second-pass, only one TTA will be deployed to relocate the isolated term t_6 from r_{022} to either leaf node r_{011},

r_{012} or r_{021} depending on the average distances of these leaf nodes with respect to t_6.

The algorithm for the second-pass using $n°W$ is described in Algorithm 4. Unlike the *ant_traverse*() function in Algorithm 3 where each new sub-node is processed as a separate iteration of *ant_traverse*() using an independent TTA, there is only one TTA required throughout the second-pass.

Algorithm 4 Function ant_refine(*leafnodes*, r_u) using n°W

```
1: if |r_u| = 1 then
2:    //current leaf node has isolated term t_i
3:    pick(t_i)//based on Eq. 9
4:    for each r_l ∈ leafnodes do
5:       for each term t_j in current leaf node r_l do
6:          //jump from one leaf node to the next to sense neighborhood density
7:          δ(t_i, t_j) := sense_distance(t_i, t_j)//based on Eq. 8
8:          remember_distance(δ(t_i, t_j))
9:       end for
10:      f^2_TTA(t_i, r_l) := sense_neighbourhood()//based on Eq. 10
11:      remember_neighbourhood(f^2_TTA(t_i, r_l))
12:   end for
13:   //back to original leaf node of term t_i after visiting all other leaves
14:   {∀l, f^2_TTA(t_i, r_l)} := recall_neighbourhood()
15:   r_v := decide_drop({∀l, f^2_TTA(t_i, r_l)})// based on Eq. 11
16:   if r_v not null then
17:      drop({t_i},{r_v})//drop at ideal leaf node
18:   else
19:      drop({t_i},{r_u})//outlier. no ideal leaf node. drop back at original leaf
20:   end if
21: end if
```

5 Evaluations and discussions

There are three ways of assessing ontology learning approaches, namely, evaluation as part of a working application, empirical evaluation by domains experts, and evaluation using gold standards (i.e. reference-based evaluation). For reference-based evaluations, different measures are available (Dellschaft and Staab 2006) for evaluating the quality of the different intermediate outputs of ontology learning (i.e. layers on an ontology) such as terms, concepts, and relationships. At the lexical term layer, the extraction performance can be measured using *Lexical Precision* (*LP*) and *Lexical Recall* (*LR*). *LP* and *LR* are employed to assess the relevance of the extracted terms (i.e. $correct_e$) by comparing them against a gold standard corpus. *LP* and *LR* are defined as (Sabou et al. 2005):

$$LP = \frac{|correct_e|}{|all_e|} \tag{12}$$

$$LR = \frac{|correct_e|}{|all_c|} \tag{13}$$

where all_c is the set of all terms in the corpus and all_e is the set of extracted terms.

At the concept hierarchy layer, we assess the degree to which the concepts and taxonomic relationships in an ontology covers a certain domain of interest. For comparing the domain coverage, evaluations can be performed at two different levels, namely, lexical and conceptual. At the conceptual level, taxonomic structures are taken into consideration. At the lexical level, only the sets of terms representing the concepts are compared. In this paper, we will only focus on the evaluation of the concept hierarchy layer at the lexical level since we have yet to verify the taxonomic structure discovered by the TTAs. For the purpose of our experiments, we will employ three existing metrics. The first is known as *Lexical Overlap* (LO) for evaluating the intersection between the discovered concepts (C_d) and the recommended (i.e. manually created) concepts (C_m) (Maedche and Staab 2002). The manually created concepts can be regarded as the reference for our evaluations. LO is defined as:

$$LO = \frac{|C_d \cap C_m|}{|C_m|} \tag{14}$$

Some minor changes were made in terms of how the intersection between the set of recommended clusters and discovered clusters (i.e. $C_d \cap C_m$) should be computed. The normal way of having exact lexical matching of the concept identifiers cannot be applied to our experiments. Due to the ability of TTA in discovering concepts with varying level of granularity depending on s_T, we have to put into consideration the possibility of sub-clusters that collectively correspond to some recommended clusters. For this reason, such related sub-clusters that are automatically discovered can be collectively considered as a valid representation of the corresponding recommended clusters. For our evaluations, the presence of discovered sub-clusters that correspond to some recommended clusters are considered as a valid intersection. In other words, given that $C_d = \{c_1, \ldots, c_n\}$ and $C_m = \{c_x\}$ where $c_x \notin C_d$, then

$$|C_d \cap C_m| = 1 \text{ if } c_1 \cup \ldots \cup c_n = c_x$$

The second and third metric are meant to account for new concepts that are absent during the manual creation, and for concepts which exist during manual creation but were not discovered, respectively. The second metric is referred to as *Ontological Improvement* (OI) while the third metric is known as *Ontological Loss* (OL). They are defined as (Sabou et al. 2005):

$$OI = \frac{|C_d - C_m|}{|C_m|} \tag{15}$$

$$OL = \frac{|C_m - C_d|}{|C_m|} \tag{16}$$

Ontology learning is an incremental process that involves the continuous maintenance of ontology every time new terms are added. As such, we do not see clustering large datasets as a problem. At the moment, we are employing seven datasets to assess the quality of the discovered clusters using the three metrics described above. The origin of the datasets and some brief descriptions are provided below:

- Three of the datasets used for our experiments were obtained from the UCI Machine Learning Repository.[8] They are WINE_15T, MUSHROOM_16T and DISEASE_20T. These datasets have to be modified for our purpose because they contain mainly numerical attributes that are meant for use with feature-based similarities.
- We also use the original Animals dataset (i.e. ANIMAL_16T) proposed for use with Self-Organizing Maps (SOMs) by Ritter and Kohonen (1989).
- We constructed the remaining three datasets called ANIMAL GOOGLE_16T, MIX_31T, and MIX_60T. ANIMALGOOGLE_16T is similar to the ANIMAL_16T dataset except for a single replacement using the term *"Google"*. The other two MIX dataset consist of a good mixture of terms from a large number of domains.

Table 1 summarizes the datasets employed for our experiments. The column C_m are the recommended clusters and C_d are clusters automatically discovered using TTA. Table 2 summarizes the evaluation of TTA using the three metrics for all ten experiments. The high performance in terms of LO shows the good domain coverage of the discovered clusters. The occasionally high OI demonstrates the ability of TTA in highlighting new, interesting concepts that were ignored during manual creation of the recommended clusters.

During the experiments, snapshots were produced to show the results in two parts: results after the first-pass using NGD, and results after the second-pass using $n^{\circ}W$. The first experiment uses WINE_15T. The original dataset has 178 nameless instances spread out over three clusters. Each instances has 13 attributes for use with feature-based similarity measures. We augment the dataset by introducing famous names in the Wine domain and remove their numerical attributes. We maintained the three clusters namely, *"white"*, *"red"*, and *"mix"*. *"Mix"* is actually referring to wines that were named after famous wine regions around the world. Such wines can either be red or white. As shown in Fig. 3, setting $s_T = 0.92$ produces five clusters. Clusters A and D are actually sub-clusters for the recommended cluster *"red"*, while Clusters C and E are sub-clusters for the recommended cluster *"white"*. Cluster B corresponds exactly to the recommended cluster *"mix"*. The second experiment uses MUSHROOM_16T. The original dataset has 8,124 nameless instances spread out over two clusters. Each instances has 22 nominal attributes for use with feature-based similarity

[8] http://www.ics.uci.edu/~mlearn/MLRepository.html

Table 1 Summary of the datasets employed for experiments.

Experiment	Datasets	Number of terms	Number of recommended clusters C_m	Number of discovered clusters (obtained through experiments), C_d	Granularity (determined using S_r)
1	WINE_15T	15	3 [red, white, mix]	5 [red_non_noble, white_non_noble, mix, red_nible, white_noble]	0.92
2	MUSHROOM_16T	16	2[edible, poisonous]	3 [edible_ east_asian, poisonous, edible_western]	0.89
3	DISEASE_20T	20	4 [skin, blood, cardio-vascular, digestive]	7 [cardiovascular_ inflame_vein, skin, blood_anaemia, blood_low_ leukocyte_platelet, blood_etc, digestive, cardiovascu-lar_inflame_clot_vein]	0.86
4	ANIMAL_16T	16	2 [bird, mammal]	2 [bird, mammal	0.60
5	ANIMAL_16t	16	2 [bird, mammal]	5 [bird, mammal_hoofed, mammal_ prey_feline, mammal_prey_ canine, mammal_ kept_as_pet]	0.72
6	ANIMAL GOOGLE_16T	16	3 [bird, mammal, google]	2[animal, google]	0.58
7	ANIMAL GOOGLE_16T	16	3 [bird, mammal, google]	3 [bird, mammal, google]	0.60
8	ANIMAL GOOGLE_16T	16	3 [bird, mammal, google]	5 [bird, mammal_prey, mammal_hoofed, mammal_ kept_as_pet, google]	0.72
9	MIX_31T	31	8[actor_ actress, musician, country, politics, transport, finance_ account, internet, food]	8 [actor_actress, musician, country, politics, transport, finance_account, internet, food]	0.70

Table 1 continued

Experiment	Datasets	Number of terms	Number of recommended clusters C_m	Number of discovered clusters (obtained through experiments), C_d	Granularity (determined using S_r)
10	MIX_60T	60	12 [transport, country_city, marsupial, mammal_prey, bird, finance_account, computing_hardware, food, beverage, politician, plant, herb]	20 [herb, food_pastry, food_italian, computing_hardware, politician, country_city_france, country_city_etc, plant_eucalyptus marsupial, finance_account, transport_four_more_wheel, plant_organ, beverage, bird_prey, bird_etc, transport_two_wheel, mammal_prey, plant_acacia	0.76

Column C_m are the recommended clusters and C_d are clusters automatically discovered using TTA.

Table 2 Summary of the evaluation results for all ten experiments using the three metrics LO, OI and OL

Experiment	*Datasets*	C_m	C_d	$C_d \bigcap C_m$ *(common)*	C_m-C_d *(missed)*	C_d-C_m *(new)*	*LO*	*OI*	*OL*
1	WINE_15T	3	5	3	0	2	100%	67%	0%
2	MUSHROOM_16T	2	3	2	0	1	100%	50%	0%
3	DISEASE_20T	4	7	4	0	3	100%	75%	0%
4	ANIMAL_16T	2	2	2	0	0	100%	0%	0%
5	ANIMAL_16T	2	5	2	0	3	100%	150%	0%
6	ANIMAL GOOGLE_16T	3	2	2	1	0	67%	0%	33%
7	ANIMAL GOOGLE_16T	3	3	3	0	0	100%	0%	0%
8	ANIMAL GOOGLE_16T	3	5	3	0	2	100%	67%	0%
9	MIX_31T	8	8	8	0	0	100%	0%	0%
10	MIX_60T	12	20	12	0	8	100%	67%	0%
Average							97%	48%	3%

measures. We augment the dataset by introducing names of mushrooms that fit into one of the two recommended clusters, namely, *"edible"* and *"poisonous"*. As shown in Fig. 4, setting $s_T = 0.89$ produces four clusters. Cluster A corresponds exactly to the recommended cluster *"poisonous"*. The remaining three clusters where actually sub-clusters of the recommended cluster *"edible"*. Cluster B contains edible mushrooms prominent in East Asia while Clusters C and D comprise of mushrooms found mostly in North America and Europe, and are

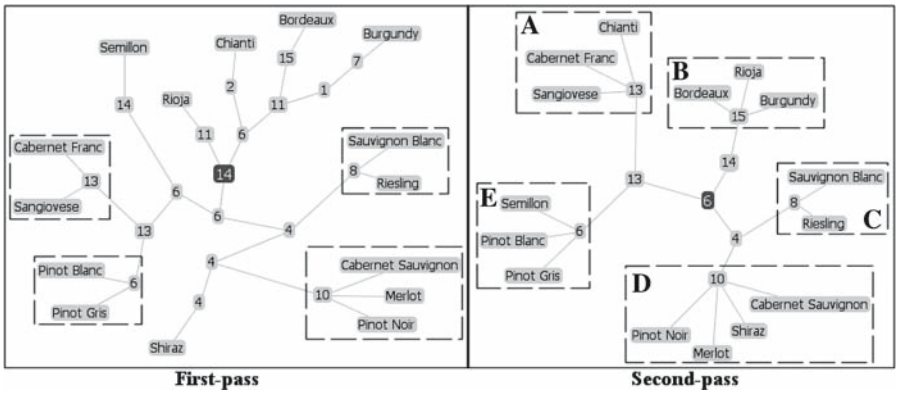

Fig. 3 Experiment using 15 terms from the Wine domain. Setting $s_T = 0.92$ results in five clusters. Cluster A is simply red wine grapes or red wines, while Cluster E represents white wine grapes or white wines. Cluster B represents wines named after famous regions around the world and they can either be red, white, or rose. Cluster C represents white noble grapes for producing great wines. Cluster D represents red noble grapes. Even though not common, but Shiraz was occasionally admitted to this group

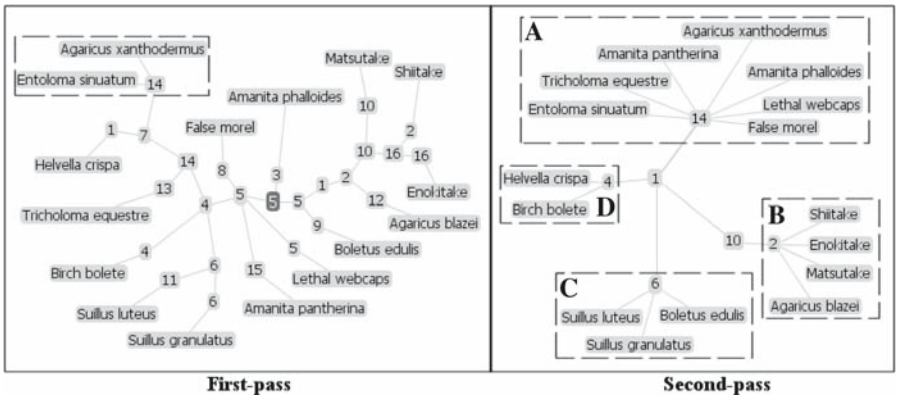

Fig. 4 Experiment using 16 terms from the Mushroom domain. Setting $s_T = 0.89$ results in four clusters. Cluster A represents poisonous mushrooms. Cluster B comprises of edible mushrooms which are prominent in East Asian cuisine except for Agaricus Blazei. Nonetheless, this mushroom was included in this cluster due to its high content of beta glucan for potential use in cancer treatment, just like Shiitake. Moreover, China is the major exporter of Agaricus Blazei, also known as Himematsutake, further relating this mushroom to East Asia. Cluster C and D comprises of edible mushrooms found mainly in Europe and North America, and are more prominent in Western cuisines

prominent in Western cuisines. Similarly, the third experiment was conducted using DISEASE_20T. At $s_T = 0.86$ TTA discovered hidden sub-clusters within the four recommended clusters, namely, *"skin", "blood", "cardiovascular",* and *"digestion"*. In relation to this, Handl et al. (2006) highlighted a shortcoming in their evaluation of ant-based clustering algorithm. The authors state that the algorithm *"... only manages to identify these upper-level structures and fails to*

further distinguish between groups of data within them". In other words, unlike existing ant-based algorithms, the first three experiments demonstrated that our TTA has the *ability to further distinguish hidden structures within clusters*.

The fourth and fifth experiments were conducted using ANIMAL_16T dataset. This dataset has been employed to evaluate both the *standard ant-based clustering* (*SACA*) and an improved version called A^2CA by Vizine et al. (2005). The original dataset consists of 16 named instances, each representing an animal using binary feature attributes. Both *SACA* and A^2CA discovered two natural clusters, one for *"mammal"* while the other for *"bird"*. While *SACA* was inconsistent in its results, A^2CA yielded 100% recall rate over 10 runs. The authors of A^2CA stated that the dataset can also be represented as three recommended clusters. In the spirit of the evaluation by Vizine et al., we performed the clustering of the 16 animals using TTA over 10 runs. In our case, no features were used. Just like all experiments in this paper, the 16 animals were clustered based on their names. As shown in the fourth experiment in Fig. 5, by setting $s_T = 0.60$, the TTAs automatically discovered the two recommended clusters after the second-pass: *"bird"* and *"mammal"*. While the ant-based methods are known for their intrinsic capability in identifying clusters automatically, conventional clustering methods (e.g. K-means, average link agglomerative clustering) rely on the specification of the number of clusters (Handl et al. 2006). The inability to control the desired number of natural clusters can be troublesome. According to Vizine et al. (2005), *"in most cases, they generate a number of clusters that is much larger that the natural number of clusters"*. Unlike both extremes, TTA has the *flexibility in regards to the discovery of clusters*. The granularity and number of discovered clusters in TTA can be adjusted by simply modifying the threshold s_T. By setting higher s_T, the number of discovered clusters for ANIMAL_16T has been increased to five as shown in Fig. 6. A lower value

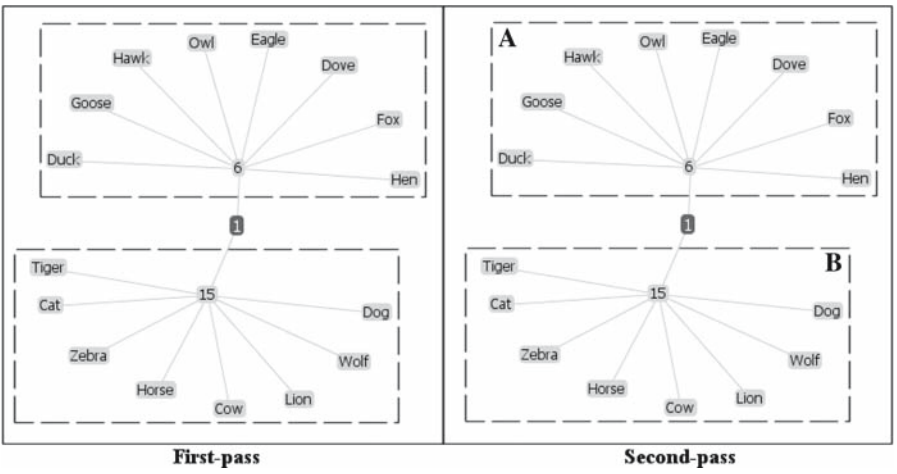

Fig. 5 Experiment using 16 terms from the Animal domain. Setting $s_T = 0.60$ results in two clusters. Cluster A comprises of birds and Cluster B represents mammals

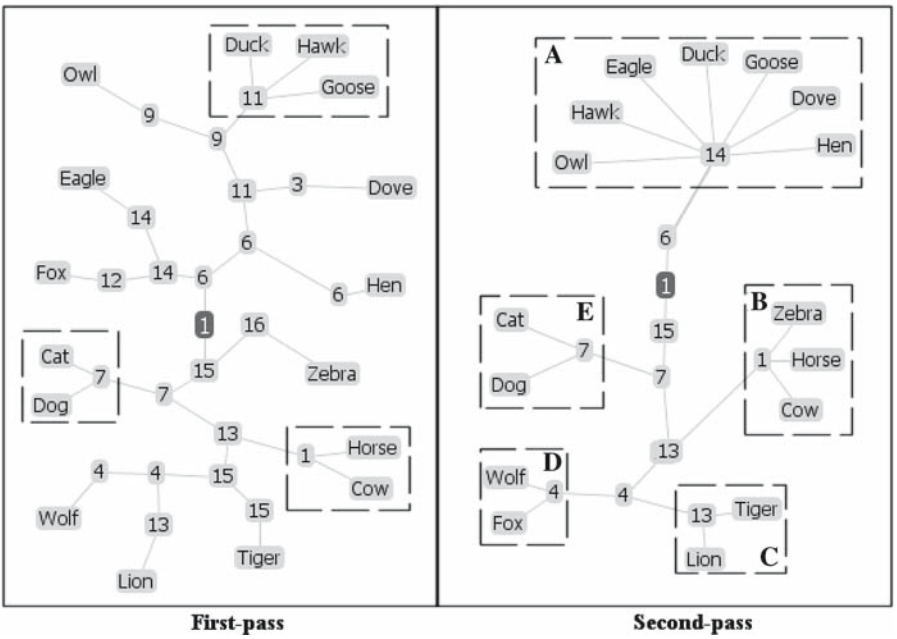

Fig. 6 Experiment using 16 terms from the Animal domain (the same dataset from the experiment in Figure 5). Setting $s_T = 0.72$ results in five clusters. Cluster A represents birds. Cluster B includes hoofed mammals (i.e. ungulates). Cluster C corresponds to predatory feline while Cluster D represents predatory canine. Cluster E constitutes animals kept as pet

of the desired *ideal intra-node similarity* s_T results in less branching out and hence less clusters, making the elements in the final leaf nodes more loosely coupled. Conversely, setting higher s_T produces more tightly coupled terms where the similarities between elements in leaf nodes are very high. In the fifth experiment depicted in Fig. 6, the value s_T was raised to 0.72 and more refined clusters were discovered: *"bird", "mammal_hoofed", "mammal_kept_as_pet", "predatory_canine",* and *"predatory_feline".*

The next three experiments were conducted using the ANIMAL-GOOGLE_16T dataset. These three experiments were meant to reveal another advantage of TTA through the presence of an *outlier* namely the term *"Google".* An outlier can be simply considered as a term that does not fit into any of the clusters. According to Berkhin (2002), *"Legitimately, outliers can be viewed as legitimate records having abnormal behaviour. In general, clustering techniques do not distinguish between the two..."* In Fig. 7, TTA successfully isolated the term *"Google"* while discovering clusters at different levels of granularities based on the different settings of s_T. As similar terms will be clustered into the same node, outliers will eventually be singled out as individual terms in leaf nodes. Consequently, unlike some conventional methods such as K-means (Yao and Choi 2003), clustering using TTA is not susceptible to poor results due to outliers. In fact, there are two ways of looking at the term *"Google",* one as

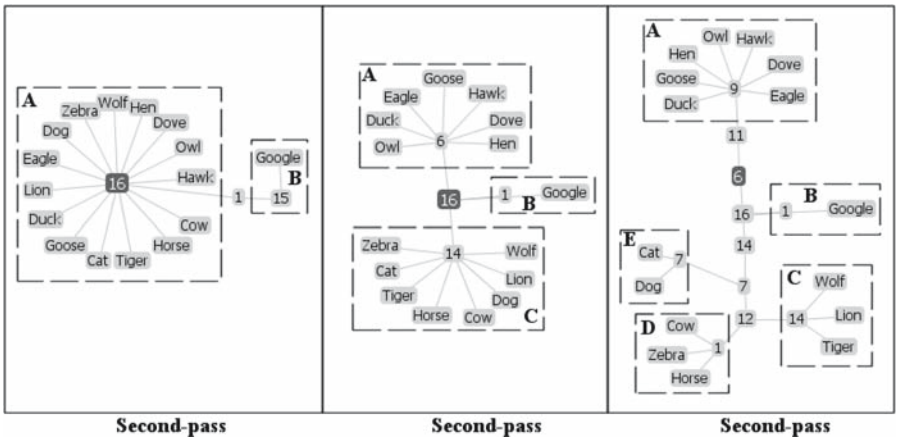

Fig. 7 Experiment using 15 terms from the Animal domain plus an additional term *"Google"*. Setting $s_T = 0.58$ (left screenshot), $s_T = 0.60$ (middle screenshot) and $s_T = 0.72$ (right screenshot) result in two clusters, three clusters, and five clusters, respectively. In the left screenshot, Cluster A acts as the parent for the two recommended clusters *"bird"* and *"mammal"*, while Cluster B includes the term *"Google"*. In the middle screenshot, the recommended clusters *"bird"* and *"mammal"* were clearly reflected through Cluster A and C, respectively. By setting s_T higher, we dissected the recommended cluster *"mammal"* to obtain the discovered sub-clusters C, D, and E as shown in the right screenshot

an outlier as described above, or the second as an extremely small cluster with one term. Either way, the term *"Google"* demonstrates two abilities of TTA: *capable of identifying and isolating outliers*, and *tolerance to differing cluster sizes* like its predecessors. Handl et al. (2006) have shown through experiments that certain conventional clustering methods such as K-means and one-dimensional self-organizing maps perform poorly in the face of increasing deviations between cluster sizes.

The last two experiments were conducted using MIX_31T and MIX_60T. Figure 8 shows the results after the first-pass and second-pass using 31 terms while Fig. 9 shows the final results using 60 terms. Similar to the previous experiments, the first-pass resulted in a number of clusters plus some isolated terms. The second-pass aims to relocate these isolated terms to the most appropriate clusters. Despite the rise in the number of terms from 31 to 60, all the clusters formed by the TTAs after the second-pass correspond precisely to their occurrences in real-life (i.e. natural clusters). With the absolute consistency of the results over 10 runs, these two experiments yield 100% recalls just like the previous experiments. Consequently, we can claim that TTA is *able to produce consistent results*, unlike the standard ant-based clustering where the solution does not stabilize and fail to converge. For example, in the evaluation by Vizine et al. (2005), the standard ant-based clustering were inconsistent in their performance over the 10 runs using the ANIMAL_16T dataset. This is a very common problem in ant-based clustering when *"they constantly construct and*

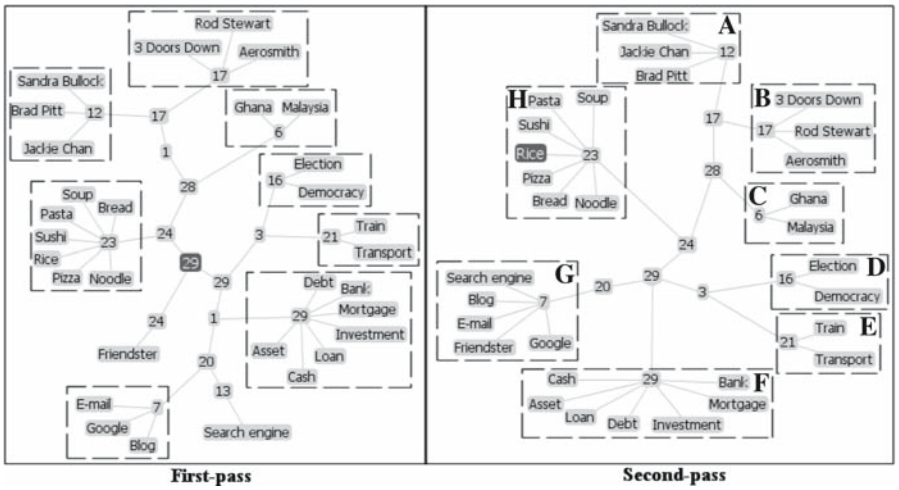

Fig. 8 Experiment using 31 terms from various domains. Setting $s_T = 0.70$ results in eight clusters. Cluster A represents actors and actresses. Cluster B represents musicians. Cluster C represents countries. Cluster D represents politics-related notions. Cluster E is transport. Cluster F includes finance and accounting matters. Cluster G constitutes technology and services on the Internet. Cluster H represents food

deconstruct clusters during the iterative procedure of adaptation" (Vizine et al. 2005).

There is also another advantage of the TTAs that is not found in the standard ants namely the ability to identify taxonomic relationships between clusters. Referring to all the 10 experiments conducted, we noticed that there are implicit hierarchical information that connects the discovered clusters. For example, referring to the most recent experiment in Fig. 8, the two discovered Clusters A (which contains *"Sandra Bullock", "Jackie Chan", "Brad Pitt"*) and B (which contains *"3 Doors Down", "Aerosmith", "Rod Stewart"*) after the second-pass share the same parent node. We can employ the graph of Wikipedia articles W to find the *most common parent* of the two natural clusters and label it with the category name provided by Wikipedia. In our case, we can label the parent node of the two natural clusters as *"Entertainers"*. In fact, the labels of the natural clusters themselves can be named using the same approach. For example, the terms in the discovered cluster B (which contains *"3 Doors Down", "Aerosmith","Rod Stewart"*) fall under the same category *"American musicians"* in Wikipedia and hence, we can accordingly label the cluster using that category name. In other words, clustering using TTA with the help of *NGD* and $n^{\circ}W$ does not only produce flexible and consistent natural clusters, but is also *able to identify implicit taxonomic relationships between clusters*. Nonetheless, we would like to point out that not all hierarchies of natural clusters formed by the TTAs correspond to real-life hierarchical relationships. More research is required to properly validate this capability of the TTA.

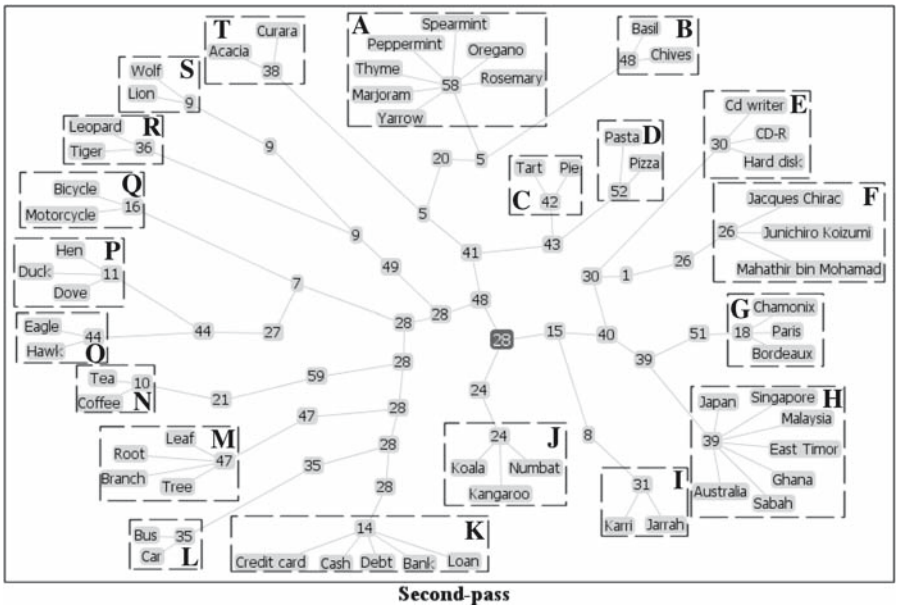

Fig. 9 Experiment using 60 terms from various domains. Setting $s_T = 0.76$ results in 20 clusters. Cluster A and B represent herbs. Cluster C comprises of pastry dishes while Cluster D represents dishes of Italian origin. Cluster E represents computing hardware. Cluster F is a group of politicians. Cluster G represents cities or towns in France while Cluster H includes countries and states other than France. Cluster I constitutes trees of the genus Eucalyptus. Cluster J represents marsupials. Cluster K represents finance and accounting matters. Cluster L represents transports with four or more wheels. Cluster M represents plant organs. Cluster N represents beverages. Cluster O represents predatory birds. Cluster P represents birds other than predatory birds. Cluster Q represents two-wheeled transports. Cluster R and S represent predatory mammals. Cluster T represents trees of the genus Acacia

One can notice that in all the experiments of this section, the quality of the overall output of clustering using TTA was less desirable if we were to only rely on the result of the first-pass. As pointed out earlier, the second-pass is necessary to produce naturally occurring clusters. The results after the first-pass usually contain isolated terms due to discrepancies in *NGD*. This is mainly due to the appearance of words and popularity of word pairs that are *not natural*. For example, given the words *"Fox", "Wolf"* and *"Entertainment",* the first two should go together naturally. Unfortunately, due to the popularity of the name *"Fox Entertainment",* a search in Google using the pair *"Fox"* and *"Wolf"* will generate lower page count as compared to *"Fox"* and *"Entertainment"*. A lower page count will have adverse effects on Eq. 2, resulting in lower similarity. Using Eq. 4, *"Fox"* and *"Entertainment"* achieve a similarity of 0.7488 while *"Fox"* and *"Wolf"* yield a lower similarity of 0.7364. Despite such shortcomings related to *NGD* and probably, $n^{\circ}W$, these two huge collection of online documents offer TTA the ability to handle technical or everyday terms of any domain regardless of whether they have been around for some time or merely beginning to evolve into common use on the Internet. As visible throughout all of our experiments,

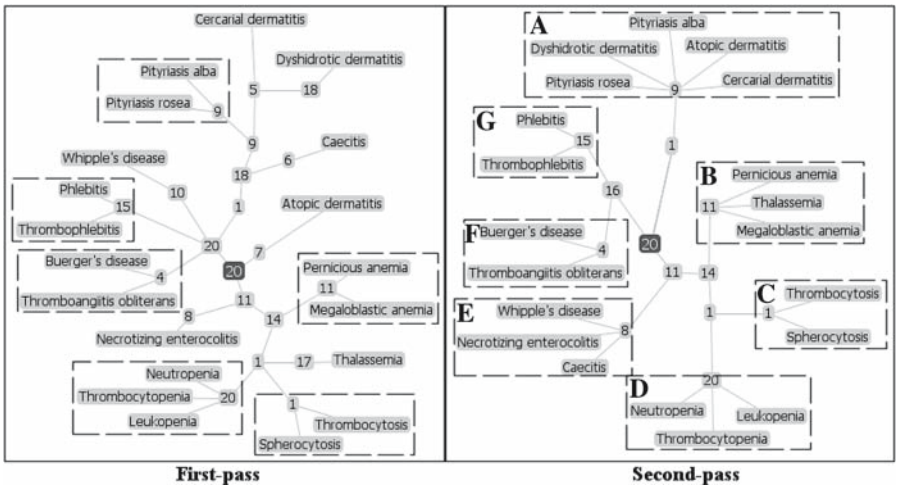

Fig. 10 Experiment using 20 terms from the Disease domain. Setting $s_T = 0.86$ results in seven clusters. Cluster A represents skin diseases. Cluster B represents a class of blood disorders known as anaemia. Cluster C represents other kinds of blood disorders. Cluster D represents blood disorders characterized by the relatively low count of leukocytes (i.e. white blood cells) or platelets. Cluster E represents digestive diseases. Cluster F represents cardiovascular diseases characterized by both the inflammation and thrombosis (i.e. clotting) of arteries and veins. Cluster G represents cardiovascular diseases characterized by the inflammation of veins

the TTAs successfully cluster objects based only on their names, regardless of how common or rare the usage of such names are. Due to the mere reliance on names or nouns for clustering, some readers may question TTAs' ability in handling various linguistic issues such as synonyms and word senses. Looking back at Fig. 10, the term *"Buerger's disease"* and *"Thromboangiitis obliterans"* are actually synonyms referring to the acute inflammation and thrombosis (clotting) of arteries and veins of the hands and feet. In the context of the experiment in Fig. 3, the term *"Bordeaux"* was treated as *Bordeaux wine* instead of the *city of Bordeaux* and successfully clustered together with other wines from other famous regions such as *"Burgundy"*. In another experiment in Fig. 9, the similar term *"Bordeaux"* was automatically disambiguated and treated as a port-city in the Southwest of France instead. The TTA then automatically cluster this term together with other cities in France such as *"Chamonix"* and *"Paris"*. In short, TTA poses the *inherent capability of coping with synonyms, word senses and the fluctuation in terms usage*.

The quality of the clustering results is very much dependent on the choice of s_T and to a lesser extent, δ_T. Nonetheless, as an effective rule-of-thumb, s_T should be set as high as possible. Higher s_T will result in more leaf nodes with each having possibly a smaller number of terms that are tightly coupled together. High s_T will also enable the isolation of potential outliers. The isolated terms and outliers generated by a high s_T can then be further refined in the second-pass. The ideal range of s_T derived through our experiments is within 0.6–0.9. Setting s_T too low will result in very coarse clusters like the ones

shown in Fig. 5 where potentially hidden clusters are left uncovered. Regarding the value of δ_T, it is usually set inversely proportional to s_T. As can be witnessed from our evaluations, as we set s_T higher, we decreases the value of δ_T. The reason behind the choices of these two threshold values can be explained as follows: as we lower s_T, it will results in coarser clusters with loosely coupled terms. The intra-node distance of such clusters are inevitably higher compared to the much finer clusters because the terms in these coarse clusters are more likely to be less similar. In order for the second-pass to function appropriately during the relocation of isolated terms and the isolation of outliers, δ_T has to be set comparatively higher. Besides, lower s_T will not provide the adequate discriminative ability for the TTAs to distinguish or pick out the outliers. Another interesting point about s_T is that by setting it to the maximum (i.e. 1.0), it results in a divisive clustering effect. In divisive clustering, the process starts with one, all-inclusive cluster and at each step, splits the cluster until only singleton clusters of individual term remain (Steinbach et al. 2000). When $s_T = 1.0$, the TTAs will continue to branch out from every current main nodes until there is only one term left for all the leaf nodes.

6 Conclusion and future work

In this paper, we presented a research that seeks to introduce a decentralized multi-agent system for term clustering in ontology learning. Unlike document clustering or other forms of clustering in pattern recognition, clustering terms in ontology learning requires a different approach. The most evident adjustment required in term clustering is the measure of similarity and distance. Existing approaches employed for term clustering in many ontology learning systems remain confined within the realm of conventional clustering methods and feature-based similarity measures. As there are no explicit features attached to terms, unlike an image of a man that has heights and weights, these existing approaches have come to rely on contextual cues surrounding the terms. Not only that these approaches yield results that have yet to reach a level we desire, they are expensive in terms of the need for an extremely large collection of domain documents to reliably extract contextual cues for the pre-computation of similarity matrices. In addition the static resources employed by these systems such as WordNet, patterns and templates make the approach even more difficult to port across domains.

Consequently, we have proposed the innovative use of *featureless* similarity based on *Normalized Google Distance (NGD)* and *n° of Wikipedia (n°W)*. The use of the two similarity measures as part of a new hybrid clustering algorithm called *Tree-Traversing Ant* (TTA) demonstrated excellent results during our evaluations. Standard ant-based methods exhibit certain characteristics that have been shown to be useful and superior compared to conventional clustering methods. The TTA is the result of an attempt to inherit these strengths while avoiding some inherent drawbacks. In the process, certain advantages from the conventional divisive clustering were incorporated, resulting in the appearance

of a hybrid between ant-based and conventional algorithm. Seven of the most notable strength of the TTA with *NGD* and $n^{\circ}W$ are:

- Able to further distinguish hidden structures within clusters;
- Flexible in regards to the discovery of clusters;
- Capable of identifying and isolating outliers;
- Tolerance to differing cluster sizes;
- Able to produce consistent results;
- Able to identify implicit taxonomic relationships between clusters; and
- Inherent capability of coping with synonyms, word senses and the fluctuation in terms usage.

Nonetheless, much work is still required in certain aspects. One of the main future work we have in plan is to ascertain the validity and make good use of the implicit hierarchical relationships discovered by the TTAs. In addition, the next issue that interests us is the formalization of the automatic labeling of the natural clusters and the nodes in the hierarchy using Wikipedia. Labeling has always been a hard problem in clustering especially documents and term clustering. We are also keen on conducting more studies on the relationship between the two most important thresholds in TTA, namely, s_T and δ_T. If possible, we intend to find ways for the automatic adjustment of these threshold values to maximize the quality of our output.

Acknowledgements This research was supported by the Australian Endeavour International Postgraduate Research Scholarship, and a Research Grant 2006 from the University of Western Australia.

References

Bennett C, Gacs P, Li M, Vitanyi P, Zurek W (1998) Information distance. IEEE Trans Inform Theory 44(4): 1407–1423

Berkhin P (2002) Survey of clustering data mining techniques. Technical report. Accrue Software

Choi B, Yao Z (2005) Web page classification. In: Chu W, Lin T (eds) Foundations and advances in data mining. Springer-Verlag

Cilibrasi R, Vitanyi P (2005) Automatic meaning discovery using google. http://xxx.lanl.gov/abs/cs.CL/0412098

Cilibrasi R, Vitanyi P (2006) Automatic extraction of meaning from the web. In: Proceedings of the IEEE international symposium on information theory, Seattle, USA

Cimiano P, Staab S (2005) Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In: Proceedings of the workshop on learning and extending lexical ontologies with machine learning methods, Bonn, Germany

Dellschaft K, Staab S (2006) On how to perform a gold standard based evaluation of ontology learning. In: Proceedings of the 5th international semantic web conference (ISWC)

Deneubourg J, Goss S, Franks N, Sendova-Franks A, Detrain C, Chretien L (1991) The dynamics of collective sorting: robot-like ants and ant-like robots. In: Proceedings of the 1st international conference on simulation of adaptive behavior: from animals to Animats, France

Faure D, Nedellec C (1998) A corpus-based conceptual clustering method for verb frames and ontology acquisition. In: Proceedings of the 1st international conference on language resources and evaluation (LREC), Granada, Spain

Faure D, Poibeau T (2000) First experiments of using semantic knowledge learned by asium for information extraction task using intex. In: Proceedings of the 1st Workshop on Ontology Learning, Berlin, Germany

Gomez-Perez A, Manzano-Macho D (2003) A survey of ontology learning methods and techniques. Deliverable 1.5, OntoWeb Consortium

Grunwald P, Vitanyi P (2003) Kolmogorov complexity and information theory. J Logic Language(and Information) 12(4): 497–529

Gutowitz H (1993) Complexity-seeking ants. In: Proceedings of the 3rd European conference on artificial life.

Handl J, Meyer B (2002) Improved ant-based clustering and sorting. In: Proceedings of the 7th international conference on parallel problem solving from nature

Handl J, Knowles J, Dorigo M (2003) Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. Technical Report TR/IRIDIA/2003-24, Universite Libre de Bruxelles

Handl J, Knowles J, Dorigo M (2006) Ant-based clustering and topographic mapping. Artif Life 12(1): 35–61

Jain A, Murty M, Flynn P (1999) Data clustering: a review. ACM Comput Survey 31(3): 264–323

Lagus K, Honkela T, Kaski S, Kohonen T (1996) Self-organizing maps of document collections: A new approach to interactive exploration. In: Proceedings of the 2nd international conference on knowledge discovery and data mining

Lelewer D, Hirschberg D (1987) Data compression. ACM Comput Surveys 19(3): 261–296

Lumer E, Faieta B (1994) Diversity and adaptation in populations of clustering ants. In: Proceedings of the 3rd international conference on simulation of adaptive behavior: from animals to animats 3

Maedche A, Staab S (2002) Measuring similarity between ontologies. In: Proceedings of the European conference on knowledge acquisition and management (EKAW), Madrid, Spain

Maedche A, Volz R (2001) The ontology extraction & maintenance framework: text-to-onto. In: Proceedings of the IEEE international conference on data mining, California, USA

Ritter H, Kohonen T (1989) Self-organizing semantic maps. Biol Cybernet 61(1): 241–254

Sabou M, Wroe C, Goble C, Mishne G (2005) Learning domain ontologies for web service descriptions: an experiment in bioinformatics. In: Proceedings of the 14th international conference on World Wide Web

Shamsfard M, Barforoush A (2002) An introduction to hasti: an ontology learning system. In: Proceedings of the 7th Iranian conference on electrical engineering, Tehran, Iran

Shamsfard M, Barforoush A (2004) Learning ontologies from natural language texts. Int J Human-Computer Stud 60(1): 17–63

Steinbach M, Karypis G, Kumar V (2000) A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota

Vitanyi P (2005) Universal similarity. In: Proceedings of the IEEE ITSOC information theory workshop on coding and complexity, New Zealand

Vizine A, deCastro L, Hruschka E, Gudwin R (2005) Towards improving clustering ants: an adaptive ant clustering algorithm. Informatica 29(2): 143–154

Wong W, Liu W, Bennamoun M (2006) Terms clustering using tree-traversing ants and featureless similarities. In: Proceedings of the international symposium on practical cognitive agents and robots, Perth, Australia

Yao Z, Choi B (2003) Bidirectional hierarchical clustering for web mining. In: Proceedings of the IEEE/WIC international conference on web intelligence