OntoPop: An Ontology Population System for the Semantic Web

Theerayut THONGKRAU†a, Student Member and Pattarachai LALITROJWONG†, Nonmember

SUMMARY The development of ontology at the instance level requires the extraction of the terms defining the instances from various data sources. These instances then are linked to the concepts of the ontology, and relationships are created between these instances for the next step. However, before establishing links among data, ontology engineers must classify terms or instances from a web document into an ontology concept. The tool for help ontology engineer in this task is called ontology population. The present research is not suitable for ontology development applications, such as long time processing or analyzing large or noisy data sets. OntoPop system introduces a methodology to solve these problems, which comprises two parts. First, we select meaningful features from syntactic relations, which can produce more significant features than any other method. Second, we differentiate feature meaning and reduce noise based on latent semantic analysis. Experimental evaluation demonstrates that the OntoPop works well, significantly out-performing the accuracy of 49.64%, a learning accuracy of 76.93%, and executes time of 5.46 second/instance. key words: ontology population, syntactic feature extraction, latent semantic analysis, semantic web

1. Introduction

The Semantic Web employs a knowledge base description known as ontology. The creation and development of ontology is a subtopic in knowledge engineering, called ontology engineering. Ontology engineering divides ontology into three levels [1]. The first level, the knowledge representation level, is to prescribe representations for ontological elements such as class, relation, property, and so on. To create ontology at this level, an accepted standard must be used so that all the processing is compatible. At present, the development of ontology at this level has produced some standard languages such as RDF or OWL. At the second level, the ontology concept level, generic or universal concepts are defined and classified in the form of a hierarchy, such as "island is a subclass of location", relations between concepts, such as "island located in country". The third level, the ontology instance level, specifies the instances of each concept; for instance, the concept "island" has the instances "Phuket" and "Maldives", and the concept "country" has the instance "Thailand". To link instances together, relations must be defined, such as "Phuket located in Thailand". The development of ontology at the instance level requires the extraction of the terms defining the instances from various data sources, including unstructured documents, semi-structured documents, and structured documents. These instances then are mapped to the concepts of the ontology, and relationships are created between these instances for the next step. However, the first step is very important to the development of ontology because it allows the use of derived knowledge for ontology development applications. If there are many data sources, the result is a large ontology, which can cause difficulties and require more time for developers [2]. By employing appropriate tools and architecture, the developer can reduce the complexity and time. Current research toward solving this problem is called ontology population. Ontology population is a new field in the application of knowledge acquisition [3], with the goal of creating a tool to help humans with the development of ontology. In general, the field of ontology population comprises two working parts: instance detection and instance classification. Instance detection consists of extracting likely instances from unstructured documents, while instance classification is gathering all instances to be classified into an ontology concept. Ontology population is similar to named entity recognition and classification (NERC). The difference between the two is that ontology population emphasizes concepts at a more specific level. NERC does not categorize many classes (such as person, location, organization, GPE, and miscellaneous). Current research on ontology population could solve problems with a larger number of concepts and further expand NERC research.

Current ontology population research can be divided into three groups. The first group is based on lexico-syntactic pattern [4]–[6] that counts the number of generated patterns in documents and classifies an instance into the most frequently occurring ontology concept. For instance, "the Niger river" is found more often in documents than "the Niger country", so "Niger" would be classified under the river concept. However, this method can be misleading if an instance is new or is not often mentioned. This approach leads to an incorrect classification concept. The second group is similarity-based classification [7], [8]. This method compares the context of an instance (such as "Everest") and the context of a concept (such as mountain). If they have similar context, the instance will be classified into the ontology concept. However, this technique cannot solve the problem that the context of a concept may be different from the context of the instance [3]. For example, the “mountain” concept may feature in the phrase “mountain sports” or “mountain shoes” but the instance is unlikely to occur in “Everest sports” or “Everest shoes”. The last group is the
supervised approach [3]. The weaknesses of this technique include lack of support for fine-grained classification and a classification model that is too large for ontology development applications.

We proposed methodology to improve the weaknesses of these three techniques with a new system called OntoPop. The proposed system consists of a bootstrapping process that automatically creates the classification model by extracting the features in syntactic relations to obtain context that is most relevant to the ontology concept. This approach places more importance on context meaning than the lexico-syntactic pattern technique. Furthermore, we also apply latent semantic analysis (LSA) to distinguish feature meanings, reduce noise, and truncate the classification model size while retaining classification effectiveness. The proposed methodology supports fine-grained classification, semantic annotation, and dimension reduction, so it is more easily applied in ontology development applications. The next section summarizes the related work in this area. Section 3 summarizes OntoPop system architecture, while Sects. 4 and 5 describe the details of the system components. Section 6 presents the experiment and evaluation of our proposed system. Section 7 describe the detail of the implementation. Finally, Sect. 8 concludes the paper and offers suggestions for future research.

2. Related Work

The previous works at the current period can be divided into three groups. The first group of research is based on lexico-syntactic pattern technique. The basis of this work is to find the frequency of an emerging pattern in the collection of documents. The research of this group is often based on the Hearst pattern [9] such as PANKOW [4] and C-PANKOW [5]. The Hearst pattern is formatted as “[concept] such as [instance]”, or “[the [instance] [concept]”. An example of applying PANKOW is to make a pattern for instance of “Niger” from the country concept such as “Niger such as country”, and “the Niger country”, and a river concept such as “Niger such as river”, or “the Niger river”. Next, the pattern is queried by a search engine, and the frequency of the pattern is counted from the number of hits returned from searched documents. The pattern of concept with highest number of hits is regarded as concept of that instance. C-PANKOW is an extended system from PANKOW. This system makes a pattern which has only an instance name “such as Hilton”, and is sent for query at search engine. The result snippet is then considered with the regular expression rule such as “[concept] such as ([instance].?)+ (and/or [instance])?” When the snippet corresponds with the rule, it will show which word in what position is the name of the concept, such as “hotels such as the Ritz, the Hilton and the Holiday Inn.” Ritz, Hilton, and Holiday Inn in the sentence that can match is regarded as the hotel concept. The weakness of this research group is the processing time. If there are a lot of concepts in the ontology, it would require a lot of patterns to be sent for query. It makes the processing time very slow. On the other hand to our methodology, bootstrapping processes are proposed to be used to accumulate the information of each concept in ontology and the model must be completely formed before instance classifications. This method effectively helps reduce the processing time.

The second research group is a similarity-based classification technique. This technique is to create context vector of every ontology concept. When need to classified an unknown instance, a context vector of that unknown instance is created to compare with the context vector of every concept. The comparison is to take the dot product of two vectors, whose derived similarity value to be used in classification. Each research in this group varies in methodology in extracting context to create context vector such as Alfonseca et al. [8] creating a context vector from four signatures: topic signature is word that occurs in the same sentence, subject signature is verb, object signature is verb and proposition, and modifier signature is adjective and determiner. It is based on the principle: “if two words are semantically related, their signatures will also be similar.” Collecting words from these signatures is done by sending concept or instance to a search engine, access the content, count word frequency, and transformed it into weight by $X^T$ [10]. An example of person concept with the highest weight in topic signature is “chromosome”, “department”, and “human”. For subject signatures are “work”, “kill”, and “write”. Cimiano et al. [7] is another work in this group that extracts different contexts using pseudo-syntactic dependencies. Context extraction is done by creating rule in regular expression. These are examples of context that matches the city concept rule: “a nice city” whose context is “nice”, “the city’s center” whose context is “has_center”, or “a city near the river” whose context is “near_river”. Context vector is created by counting context frequency and then transformed into weight before being compared with an unknown instance to be classified into an ontology concept.

The research in the third group employs a supervised approach which considers classification from training data. This technique employs ontology engineer to specify a training instance of each ontology concept before creating a classification model. An example of such technique is found in the work of Tanev et al. [3] which propose a method called Class-Example by manually specifying each instance of the ontology concept in order to find related context and create the context vector for classification. Creating a context vector is done by counting frequency of each context, weighed, and compared with context vector of every training instance using dot product as in similarity-based classification for assigning an unknown instance into the ontology concept.

It was found that the there were inaccurate classifications in methods of the second group and the third group in feature occurring of synonym or polysemy. That is, the context of instances of synonyms is classified in different concepts, while they should be in the same concept. In addition, the context of some instances of polysemy is classified under the same concept, while they should be in different concepts. As seen, we have applied LSA to solve this prob-
lem. It can effectively help solve the problem in information retrieval [11]. We will be further explained in the Sect. 4.2.

Some parts of our work are similar to those of Liu et al. [12]; however, they are considerably different. In regards to fields, Liu et al.’s methodology was in the field of text classification which focuses mainly on general words, but our approach is in the field of knowledge acquisition, which is more specific than the method of Liu et al. Our approach concentrates on the construction of facts or instances for ontology and the creation of the Semantic Web. Moreover, feature extraction is also different. Our method analyzes the context of instances by looking at grammatical relationships, while the methodology of Liu et al. analyzes the context of instances by using information entropy.

3. OntoPop System Architecture

OntoPop system architecture as shown in Fig. 1. The architecture has been divided into two main processes: bootstrapping and knowledge acquisition. The input of the bootstrapping process is an input ontology contained instances in each concepts that we call training instances, while the output is the LSA space. The components of this process are syntactic feature extraction, and LSA processing. The syntactic feature extraction starts by retrieving the sentences that contain training instances from Wikipedia database†, e.g., “Fiji is an island in the pacific ocean.” After that, the sentence will parse and extract the syntactic features that are the output of this component. For example, the syntactic features of “Fiji” are “island” and “ocean”. The next component is the LSA processing, which is the semantic processing using the syntactic features from all training instances. The output of this component is the LSA space, which is the model for classifying unknown instances in the next process.

The input of the knowledge acquisition process is a web document that is used to populate instances into the input ontology. The output are the populated ontology and the Semantic Web. The components of this process are instance detection, instance classification, and semantic annotation. The first step is instance detection, detecting possible instances from the web document. The output of this component is the unknown instance, e.g., “Phuket”, “Ionion”. The second component is the instance classification, using the LSA space from the bootstrapping process to assign an unknown instance to an ontology concept. The output of this component is a concept name for each unknown instance, e.g., “Phuket” is an island, or “Ionion” is a sea. Afterwards, the system will assign concepts to instances and update the input ontology. The input ontology that is updated in the instance level is called the populated ontology. This is the first output of the knowledge acquisition process. The last component is the semantic annotation that inserts metadata to web documents, identifying all instances of the ontology concepts. The result of this component is the Semantic Web that is the second output of the knowledge acquisition process.

The bootstrapping process has one initial stage, until the input ontology is updated, while the knowledge acquisition is an iterative process.

4. Bootstrapping Process

4.1 Syntactic Feature Extraction

The goal of this stage is the extraction of terms related to the training instance. Training instance is an instance that obtained from input ontology. The syntactic feature extraction component extracts terms by considering their syntactic relation in a sentence. We apply the Stanford Parser [13] to parse sentences into syntactic relation form. An example of such a relation is Fig. 2. It shows the dependency relationship in an asymmetric binary relationship, which is a relation between two words called the head (or governor) and the modifier (or dependent) [14], where the modifier of one word can be the head of another word. Extracting the instance context from other techniques produces unrelated terms. Therefore, we mitigate this problem by selecting only two features from the chain, i.e., first order features and second order features. A first order feature is a head that has as modifier a training instance. For example in Fig. 2, “island” is in a head relation with “Fiji”. Therefore, “island” is a first order feature. A second order feature is a modifier of a first order feature. For example, “an”, “is”, and “ocean” are modifier of “island”. However “an” and “is” are not noun. Therefore, “ocean” is the only one second order feature.

Table 1 is a syntactic feature extraction algorithm that will parse a sentence $s$ into a triple form, such as...
nsubj(island, Fiji), or prep in (island, ocean). Once the list of all triple forms is obtained, the first order features are extracted by working in a loop to find triples with modifiers as training instance $i$. Then the head is extracted and identified as a feature. Once the first order features are compiled, we extract the second order features by working in a loop to extract all modifiers that match the first order features. The result is a list of first and second order features from a sentence $s$ in training instance $i$, which will be further processed in the next stage to create the LSA space.

### 4.2 LSA Processing

The second component of the bootstrapping process is to create the LSA space for populating instances in the knowledge acquisition process. LSA is a procedure based on the vector space model, which manages large sets of documents in information retrieval [15]. LSA creates a term vector representing the terms in a document and a document vector representing the document. Given a query for related documents, a query vector is created, which is represented as a pseudo-document vector formed by a weighted combination of terms and documents, and mapped to the LSA space to search for the nearest document vector; the result represents a related document. LSA can also reduce the dimension of the vector space using singular value decomposition (SVD) [16]. This is a method from linear algebra to reduce the rank of a matrix without losing important content and to eliminate noise. SVD can differentiate noise from words with several meanings (polysemy) or words with similar meanings (synonym) [17]. When the SVD process is finished, the resulting vector space is called the LSA space. From the principle of LSA in information retrieval, we borrow the idea as shown in Fig. 3. The document (such as “$d_i$”) is the training instances (such as “Taipei”), and the query is the unknown instance (such as “Phuket”). The system use a training instance representations of the ontology concepts in the LSA space that are compared to unknown instances to find similar instances. If we know that an unknown instance is near a certain training instance, we can predict the ontology concept for the unknown instance. Using SVD to reduce the dimension of the vector space enables the training instance vectors in the LSA space with similar features to be grouped near each other. It helps in assigning the unknown instances to the ontology concepts at the instance classification stage. LSA processing is divided into three steps: creating, decomposing feature–by–training instance matrix, and truncating the factorized matrix.

#### 4.2.1 Creating the Feature–by–Training Instance Matrix

A small example of a training instance and its features can found in Table 2. The features are the underlined words. To create a vector space model for the LSA, a feature–by–training instance matrix $A$ must first be constructed. Figure 4 is an example of the matrix $A$. The rows of the matrix $A$ are comprised of features, which are the individual components that mark up a training instance. The columns of the matrix $A$ are comprised of training instances, which are a predetermined set of features. An instance collection composed of $n$ training instances and $m$ features can be represented as

---

**Table 1** Syntactic feature extraction algorithm.

| Input: Sentences $s$, An instance name $i$ | Output: Syntactic features of one sentence |
|------------------------------------------|-------------------------------------------|
| EXTRACT($s$, $i$)                         |                                           |
| 1. features ← φ, first_order ← φ         |                                           |
| 2. triples ← dependency_parse($s$)       |                                           |
| 3. FOREACH triples                        |                                           |
| 3.1 IF triples.modifier = $i$             |                                           |
| AND triples.head = NOUN THEN              |                                           |
| feature ← stemming_word(triples.head)   |                                           |
| features ← features $\cup$ feature       |                                           |
| first_order ← features $\cup$ feature    |                                           |
| 4. FOREACH first_order                    |                                           |
| 4.1 FOREACH triples                       |                                           |
| IF triples.head = first_order             |                                           |
| AND triples.modifier = NOUN THEN          |                                           |
| feature ← stemming_word(triples.modifier)|                                           |
| features ← features $\cup$ feature       |                                           |
| 5. RETURN features                       |                                           |

**Table 2** LSA in IR and ontology population.

| Ontology concept | Instance name | Sentence |
|------------------|---------------|----------|
| city             | London        | London is a capital and the largest city in the UK. |
| Taipei           | Taipei        | Taipei is a city of major industrial area. |
| island           | Fiji          | Fiji is an island in the pacific ocean. |
|                  | Solomon       | The Solomon is an archipelago area in the Pacific Ocean. |
| sea              | Caribbean     | The Caribbean is a sea of the Atlantic Ocean in the Western hemisphere. |
|                  | Mediterranean | The Mediterranean is a sea inflow from the Atlantic. |

**Table 3** Feature–by–training instance matrix $A$.

|                  | London | Taipei | Fiji | Solomon | Caribbean | Mediterranean |
|------------------|--------|--------|------|---------|-----------|--------------|
| archipelago area  | 0      | 0      | 0    | 1       | 0         | 0            |
| atlantic capital | 1      | 0      | 0    | 0       | 0         | 0            |
| city              | 1      | 1      | 0    | 0       | 0         | 0            |
| inflow            | 0      | 0      | 0    | 0       | 0         | 0            |
| island            | 0      | 0      | 0    | 0       | 0         | 0            |
| ocean             | 0      | 0      | 0    | 0       | 0         | 0            |
| sea               | 0      | 0      | 0    | 0       | 0         | 0            |

---

[1]http://nlp.stanford.edu/software/dependencies_manual.pdf
each column of the matrix $A$ contains zero and nonzero elements $a_{ij}$. Each nonzero element $a_{ij}$ of the matrix $A$ is the frequency of the $i$th feature in the $j$th training instance.

4.2.2 Decomposition of the Feature–by–Training Instance Matrix

The goal of the decomposition matrix $A$ is to represent the features together with the training instances in one space. That makes it possible to calculate similarities between training instances and unknown instances. To calculate the new basis for the space and the vectors, the SVD is needed.

SVD factors the $m \times n$ matrix $A$ into three matrices in Eq. (1).

$$A = U \Sigma V^T$$  

$U$ is a rectangular matrix of size $m \times r$ whose rows are the feature vectors scaled to the new basis. The other important rectangular matrix is $V^T$, which is of size $r \times n$ and contains the new vectors for the training instance. The square matrix $\Sigma$ is a diagonal matrix. It contains the singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$, where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0$ [18].

4.2.3 Truncating Factorized Matrix

The dimension of the LSA space has to be reduced by truncating the three matrices, as shown in Fig. 5. They are truncated to a two-dimensional LSA space by reducing the rank to $k = 2$ (Section 6.6 will show an experiment on reducing each dimension). The system uses only the training instance matrix to create the LSA space as the output of the bootstrapping process.

The values in the training instance matrix $V^T_k$ are used to encode the representations of the training instance in the two-dimensional LSA space. Figure 6 shows a plot of the training instances in the LSA space. The pair $(x, y)$ is defined by $x = \text{first row of matrix } V^T_k$ and $y = \text{second row of matrix } V^T_k$. Looking at the training instance vectors, the training instances most similar to each other are determined by the angles between vectors. If two vectors are similar, then they will have a small angle between them. In Fig. 6, the training instance “Solomon” is the closest training instance to “Fiji” because they share the “ocean” feature. In contrast, “Taipei” shares the “area” feature with “Solomon” but is not close to “Solomon” because “Taipei” also shares the “city” feature with “London”. It is clear that the LSA process clarifies the discrimination among training instances in line with their occurring features, which affects the efficiency of the classification model. This is explained further in the next section.

5. Knowledge Acquisition Process

5.1 Instance Detection

The first component of this process is the instance detection. The input of this component is a web document that is required to acquire knowledge. We apply the Stanford NER [19] to detect the possible instances from the web document. The output of this component is the set of unknown instances, which will be further processed in the next stage.

5.2 Instance Classification

The unknown instances detected from the web document will be classified in this component. This component employs the LSA space from the bootstrapping process. It comprises two steps: creating the unknown instance vector in the LSA space and assigning unknown instances to ontology concepts.

5.2.1 Creating the Unknown Instance Vector

To represent an unknown instance in the LSA space, we
need an external data source that has enough information about the context of the unknown instance. At this stage, data from the search engine web service, such as Yahoo! or Bing\(^1\), are used to obtain context. The system collects context terms by sending the query as the unknown instance to the search engine. However, searching with the unknown instance results in many irrelevant data, so we append “is” to form a query in the form of “[unknown instance] is”. For example, if the unknown instance is “Phuket”, the query is “Phuket is”. The search results from the search engine consist of a list of records. Each record comprises its title, abstract or snippet, and the document’s URL. We only use the snippets for our approach and merge all the snippets as the context of the unknown instance, called the total snippet. The resulting total snippet is used to create a vector in the LSA space, called the unknown instance vector or \( I \). Figure 7 is an example of the construction of \( I \) from two snippets. The \( i \)th row of \( I \) represents the frequency of the \( i \)th feature in the total snippets. There are two terms that match the feature in the LSA space: “city” and “island”. Therefore, \( I \) has values of 1 at the positions of “city” and “island,” while at other positions, its value is 0 as there is no word in the total snippets that matches the feature in the LSA space. The resulting \( I \) is a 9-dimensional vector, and to project it to the LSA space to find the nearest training vector, its dimension must be reduced to agree with the reduction in the LSA space, which can be done via the following equation [20]:

\[
J = I^T U_k \Sigma_k^{-1}
\]  

(2)

In the above example of constructing an LSA space (see bottom in Fig. 5), we reduce the LSA space to be two-dimensional \((k = 2)\). Therefore, we must reduce the dimension of \( I \) to two as well. Figure 7 shows how to obtain \( J \) from Eq. (2). The resulting \( J \) has two dimensions and can be projected onto LSA space. To make the view clearer, we plot the projection of “Phuket” onto LSA space in Fig. 6.

5.2.2 Assigning Unknown Instance to Ontology Concept

We apply the \( k \)-nearest neighbor algorithm \((k\text{-NN})\) classification method at this stage to assign the unknown instance to an ontology concept. The \( k \)-NN procedure comprises two parts: finding the nearest neighbor and majority voting. Finding the nearest neighbor means to find the training instance vector \((V_i)\) that is nearest (or most similar) to the unknown instance vector \((J)\) at top-\(K\), which is defined in terms of the angle between two vectors from the following equation:

\[
\cos(\alpha) = \frac{J \cdot V_i}{|J| |V_i|}
\]  

(3)

The values of \( \cos(\alpha) \) are in the range from –1, corresponding to 180° or 0% similarity, to 1, corresponding to 0° or 100% similarity [21]. Looking back at Fig. 6, it can be seen that “Phuket” and “Taipei” form the least angle. Calculating the value of \( \cos(\alpha) \) using Eq. (3), we find it to have value 0.9962 or similarity 99.62%. Furthermore, “Phuket” forms a close angle with “Solomon,” and computing the value of \( \cos(\alpha) \) yields 0.8840 or similarity 88.40%. We compute \( \cos(\alpha) \) values for the angles of “Phuket” with all the training instance vectors in the LSA space and select only the \( K \) training instance vectors with the highest similarity. When \( K = 1 \), we assign the unknown instance to the single concept with the highest similarity. For example, “Phuket” and “Taipei” have the highest similarity value. We therefore assign “Phuket” to the concept of “Taipei”, which is the “city” concept. When \( K \geq 2 \), we will apply majority voting that consists of counting the number of training instances of the concept in the top-\(K\). The assignment can occur in two cases: assigning as a single concept and as multiple concepts. For the single concept assignment, we assign the concept with the highest majority voting value to the unknown instance. We show examples of the training instance and the unknown instance in the LSA space in Fig. 6. In the oval defined at \( K = 4 \), the values of the majority vote of the “island” concept and the “city” concept are both two. Therefore, “Phuket” will be assigned to both the “island” concept and the “city” concept.

5.3 Semantic Annotation

Semantic annotation formally identifies concepts and relations between concepts in documents, and is intended primarily for use by machines [22]. At this stage, the system links all the instances on web documents with populated ontology. And displaying concept name of each instance that the system has assigned via OntoPop plug-in in the ontology editor. It help ontology engineer to easily create relation between the instances from other function in the ontology editor. The output of this component will create the Semantic Web document in the RDFa [23] format. This is a W3C recommendation that adds a set of attribute level extensions to XHTML for embedding rich metadata within Web documents. An example of RDFa is shown in Table 3. In line 7 specify “Phuket” is an instance of island concept by typeof

\(^1\)http://developer.yahoo.com/search/boss/

\(^2\)http://www.bing.com/toolbox/bingdeveloper/
attribute and in line 8 specify “Thailand” is an instance of country concept.

OntoPop is similar to PowerMagpie [24]. PowerMagpie is the existing tool for semantic annotation proposed by d’Aquin et al. This system supporting create only the Semantic Web, while OntoPop can be used to create both the Semantic Web and ontological knowledge base. Besides, PowerMagpie is embedded in web browsers, while OntoPop is embedded in an ontology development application. Therefore, it is convenient for ontology engineers who can use the extracted instances in constructing relations and restrictions after the process completion.

6. Experiment and Evaluation

To compare the effectiveness of our approach with the techniques of lexico-syntactic patterns [5] and similarity-based classification [7], input ontology and validation set of both were adopted in our experiment. The input ontology is a tourism ontology developed by the GETESS project [25], which contains 682 concepts but no instances. The experiment process is shown in Fig. 8. According to Fig. 8, there are four sections. The first section is a preparation for a training set based on learning methods from external ontology. The second one is a bootstrapping process, which is a preparation for a classification model. The third part is an instance classification. The result from the third part is a result set, which is used to evaluate the effectiveness of the system at the final stage of the experiment. In the experiment process, it can be seen that knowledge acquisition processes of the OntoPop are selected only the instance classification since instance detection and semantic annotation are beyond our work.

6.1 Training Set

To prepare a training set for GETESS ontology, information from external ontology which contains instances is used. In this work, YAGO ontology [26] which has the characteristics needed for our work. Similar to GETESS, YAGO covers a large amount of information in which tourism is also included. However, since GETESS contains a lot of concepts, to select training instances manually is difficult; as a result, an automatic learning system is instead developed by adopting working principles of ontology mapping [27]. Figure 9 shows the results from learning by ontology mapping. As seen in the figure, the system matched concepts of GETESS and YAGO, whose names are similar (shown by thick lines), such as “island” with “wordnet_island”. After that, the learning system collects instances from YAGO’s concepts to be training instances in GETESS’s concepts. For example, “Fiji” and “Solomon” are set as training instances of island concept in GETESS ontology. The accumulation of the training instances in this experiment is set at a maximum of 20 instances.

6.2 Model

When the learning system accumulates all concepts of training instances on the GETESS ontology, the classification model is prepared through feature extraction of each training instance and sent to LSA processing (as previously stated in Sect. 4). Based on this process, LSA space with 4,063 features is obtained. Since in this experiment, the aim is to know the effectiveness of the approach when reducing dimensions of the LSA space. So, the LSA space with 25 to 300 dimensions is prepared and tested on the OntoPop system, one by one.

6.3 Validation Set and Test Set

A validation set is a gold standard to performance test of the ontology population. The validation set is created by selecting 30 corpora from the Lonely Planet website (http://www.lonelyplanet.com) and utilizing a human expert to classify the instances manually into GETESS ontology concepts [5]. There are 277 instances in the validation set which are stored in pairs (instance, concept). The examples are listed in Table 4. The validation set contains instances that are not identical to the training set. The validation set is prepared and used as a test set. Only the names of the instances from the validation set are used. The number of the instances in test set is also 277. The examples of test sets are shown in Table 4.
Table 4 Example of dataset.

| Training Set | Validation Set | Test Set | Result Set |
|--------------|----------------|----------|------------|
| {London, city}, {Fiji, island}, {Caribbean, sea}, ... | {Phuket, city}, {Maldive, island}, {Ionion, sea}, ... | {Phuket, Maldives, Ionion, ...} | {Phuket, Maldive, nature}, {Ionion, beach}, ... |

6.4 Parameters

The parameters used in the experiment are divided into two parts. The first part of the parameters is based on learning, which refers to the dimensions of the LSA space. The second part is a parameters set by classification which is comprised of the size of snippets (between 5 and 100), which are used during the construction of unknown vectors, and $K$ values derived from the $k$-NN technique ranging from 1 to 10.

6.5 Evaluation Measures

We follow the evaluation of the ontology population from McDowell et al. [28] The evaluation comprises 2 measures: accuracy and Learning Accuracy (LA). The accuracy measures the correspondence between the exact concept name from the validation set and the system result set. For example, in Table 4, if “Phuket” in the validation set is a “city” and “Phuket” in the system result set is a “city”, then our system has the correct result. However, if “Ionion” in the validation set is a “sea” and “Ionion” in the system result set is a “beach”, then our system has an incorrect result. The accuracy of the system can be computed by Eq. (4). For the example dataset from Table 4, we can compute the accuracy value as 1/3, that is, 0.33 or 33%.

$$\text{accuracy} = \frac{\text{Number of correct classifications}}{\text{Total number of unknown instances}} \quad (4)$$

Some results of the ontology population are related in the ontology concept [29], e.g., if “Ionion” in the validation set is a “sea” and “Ionion” in the system result set is a “beach”. Figure 9 showed the concept level in the GETESS ontology, where “sea” and “beach” have the same parent node, “nature”. We can measure this relation using the taxonomic similarity $T_{\text{sim}}$. The computation of $T_{\text{sim}}$ is shown in Eq. (5).

$$T_{\text{sim}}(a, b) = \frac{\delta(\text{root}, lcs) + 1}{\delta(\text{root}, lcs) + \delta(a, lcs) + \delta(b, lcs) + 1} \quad (5)$$

where $lcs$ is the least common superconcept of two concepts $a$ and $b$, and $\delta(a, b)$ is the number of edges on the shortest path between $a$ and $b$. In Fig. 9, $a$ is “sea”, $b$ is “beach”, $lcs$ is “nature”, $\delta(\text{root}, lcs) = 1$, $\delta(a, lcs) = 1$, and $\delta(b, lcs) = 1$. Thus, $T_{\text{sim}}(\text{sea, beach}) = (1+1)/(1+1+1+1) = 0.5$. We find all $T_{\text{sim}}$ for each pair of concepts. The average of all $T_{\text{sim}}$, as shown in Eq. (6), is called the LA of the system. For the example dataset in Table 4, we can compute LA by finding $T_{\text{sim}}(\text{city, city}) = 1$, $T_{\text{sim}}(\text{island, nature}) = 0.8$, and $T_{\text{sim}}(\text{sea, beach}) = 0.5$. The LA is $(1 + 0.8 + 0.5)/3 = 0.76$ or 76%.

$$LA = \frac{\sum T_{\text{sim}}(c, c')}{\text{Total number of unknown instances}} \quad (6)$$

6.6 Evaluation Results

The experiments were all run on a 2.90 GHZ AMD Athlon II with 4 GB main memory. The first parameter is on reducing the dimension of the LSA space to between 25 and 300 dimensions. The experiment result is shown in Fig. 10. In addition, the contribution of the SVD features are analyzed using accumulative contribution rate (ACR) [30], as listed in Table 5. We find that the LSA space of dimension 75 has the highest LA value, at 76.93% (ACR is 62.86%), and dimension 100 has the highest accuracy values, at 49.64% (ACR is 69.67%). The averages execute time at this point is 5.4572 second/instance. The second parameter tests the use of different sizes of total snippets $N$, as shown in Fig. 11. We find that $N = 45$ has the highest LA and accuracy values, at 76.03% and 48.31%, respectively. The averages execute time at this point is 5.5346 second/instance. The last parameter test is shown in Fig. 12. This shows the average LA values and average accuracy values from the experiment with different $K$ values, where the $K$ value denotes the number of the nearest training instances. The experiment finds that $K = 7$ has the highest LA and accuracy value, at 76.56%, and 49.03%, respectively. The averages execute time at this point is 5.4633 second/instance.

Finally, we compare the efficiency of our approach with the lexico-syntactic patterns [5], and similarity-based classification [7]. Table 6 shows the efficiency comparison results.
We find that our approach yields higher accuracy values than the lexico-syntactic patterns (23.79%), and similarity-based classification (20.40%). For the LA value, our method yields higher accuracy values than the lexico-syntactic patterns (5.43%), and similarity-based classification (7.06%). Furthermore, we find that our approach yields lower execute time values than the lexico-syntactic patterns (74.97 second/instance), and similarity-based classification (5.73 second/instance).

The experiment aims to demonstrate the accuracy of our methodology. Looking at the first parameter, the experiment reduces the dimension of the LSA space to different dimensions. Figure 10 shows that at the starting dimension (25 dimensions), it has low accuracy and LA value. These values gradually rise and stabilize at 75 to 125 dimensions. Even though reducing dimension helps to speed up the instance classification, the experiment shows that too small a dimension adversely affects the accuracy. For this reason, we advocate reducing the dimension to the point that yields the highest accuracy and stability, 75 to 125 dimensions for our approach. The second parameter tests different sizes of the total snippets. Figure 11 shows that at the starting point; it has the lowest accuracy and LA value. Then the graph gradually rises until \( N = 45 \) before stabilizing. We conclude that the number of snippets affects the system’s efficiency. If the size of total snippets is too small, it is hard for the system to differentiate what training instance has the same meaning and direction to as a given unknown instance. For the third parameter, the different \( K \) values, note that the graph in Fig. 12 does not show much difference in the value of accuracy and LA. However, the real trend of the accuracy and LA values from \( K = 1 \) upward is that the accuracy value gradually rises and begins to stabilize when \( K = 4 \), and then it begins to decrease.

7. Implementation

We have implemented the OntoPop system as the plug-in for the Protégé. Protégé is currently the most well-known ontology editor, is freely available, open source, and based on Java [31]. OntoPop plug-in provides easy graphic user interface of ontology populations to end user. On its first tab (Fig. 13), the plug-in displays the configuration form and processing console for create LSA space, and shows the ontology mapping result (right side in Fig. 13.) The second tab (Fig. 14) browses the input web document for acquiring knowledge. When the instance detection starts, the unknown instances are highlighted and showed list of instances that allow user to remove unwanted instance. After starts classification, the unknown instance will be shown the assigning in the ontology concept (right side in Fig. 14) and show concept name when mouse over the instances on the input web document (left side in Fig. 14.) Ontology engineer can save web document in RDFa format, update classified instances into ontology, switch to object properties tab in Protégé for create relation between the instances, and switch to data properties tab in Protégé for adding attribute data after OntoPop system finish processed.

![Fig. 11 LA and accuracy for different snippet.](image1)

![Fig. 12 LA and accuracy for different \( K \).](image2)

| Approach                  | Accuracy (%) | Learning Accuracy (%) | Execute time (second/instance) |
|---------------------------|--------------|-----------------------|-------------------------------|
| Lexico-syntactic Patterns | 25.85        | 71.50                 | 80.25                         |
| Similarity-based Classification | 29.24    | 69.87                 | 11.19                         |
| OntoPop                   | 49.64        | 76.93                 | 5.46                          |

![Table 6 Comparison for different approaches.](image3)
8. Conclusion and Future Work

An approach to ontology population for the Semantic Web has been described. This work proposed two main processes. The first process is a syntactic feature extraction to select the features with contexts most related to the instance. The second process is to construct the classification model, or LSA space, which can effectively reduce the noise and the dimension of the classification model. The contribution of this research is a new system to be used to transfer technology from the current web into the Semantic Web. It supports ontology development applications that need to extend knowledge to ontology at the instance level. In our future work, we propose to automatically build relationships between the instances, which is the step after the populating step done.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and valuable suggestions.

References

[1] M.C. Daconta, L.J. Obst, and K.T. Smith, The Semantic Web, Understanding Ontologies, Wiley Publishing, Indianapolis, 2003.
[2] M. Okabe, A. Yoshioka, K. Kobayashi, and T. Yamaguchi, “Organizational knowledge transfer using ontologies and a rule-based system,” IEICE Trans. Inf. & Syst., vol.E93-D, no.4, pp.763–773, April 2010.
[3] H. Tanen and B. Magnini, “Weakly supervised approaches for ontology population,” Ontology Learning and Population: Bridging the Gap between Text and Knowledge, pp.129–143, 2008.
[4] P. Cimiano and S. Staab, “Learning by Googling,” ACM SIGKDD Explorations Newsletter, vol.6, pp.24–33, 2004.
[5] P. Cimiano, G. Ladwig, and S. Staab, ‘Gimme’ the context: Context-driven automatic semantic annotation with C-PANKOW,” Proc. 14th Int. Conf. on World Wide Web, pp.332–341, 2005.
[6] Z. Ibrahim, S. Noah, and M. Noor, “Rules for ontology population from text of Malaysia medicinal herbs domain,” in Rough Set and Knowledge Technology, Lect. Notes Comput. Sci., vol.6401, pp.386–394, Springer Berlin/Heidelberg, 2010.
[7] P. Cimiano and J. Volker, “Towards large-scale, open-domain and ontology-based named entity classification,” Proc. Int. Conf. on Recent Advances in NLP, pp.66–166, 2005.
[8] E. Alfonseca, M. Ruiz-Casado, M. Okumura, and P. Castells, “Towards large-scale non-taxonomic relation extraction: estimating the precision of rote extractors,” Proc. 2nd Workshop on Ontology Learning and Population, pp.49–56, 2006.
[9] M. Hearst, “Automatic acquisition of hyponyms from large text corpora,” Proc. 14th Conf. Computational Linguistics, pp.539–545, 1992.
[10] E. Agirre, O. Ansa, E. Hovy, and D. Martinez, “Enriching very large ontologies using the WWW,” Proc. ECAI 2000 Workshop on Ontology Learning, 2000.
[11] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” Journal of the American Society for Information Science, vol.41, no.6, pp.391–407, 1990.
[12] T. Liu, X.L. Wang, B.Q. Liu, Y.C. Liu, and M.H. Li, “Extracting domain-specific terms from unlabeled web documents by bootstrapping and term classifiers,” Proc. IEEE Int. Conf. Systems, Man and Cybernetics, pp.3875–3880, 2007.
[13] D. Klein and C.D. Manning, “The stanford parser: A statistical parser,” 2002.
[14] L.A. Melcuk, Dependency Syntax: Theory and Practice, State University of New York Press, New York, 1988.
[15] S. Dumais, “Improving the retrieval of information from external sources,” Behavior Research Methods, Instruments and Computers, vol.23, no.2, pp.229–236, 1991.
[16] G. Strang, Introduction to Linear Algebra, Wellesley Cambridge, Massachusetts, 2003.
[17] T.K. Landauer, D.S. McNamara, S. Dennis, and W. Kintsch, Handbook of LSA, Psychology Press, New Jersey, 2007.
[18] J. Geib, Latent Semantic Indexing and Information Retrieval, VDM Verlag, 2008.
[19] J. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by Gibbs sampling,” Proc. 43rd Annual Meeting of the Association for Computational Linguistics, pp.363–370, 2005.
[20] M. Berry, S. Dumais, and G. O’Brien, “Using linear algebra for intelligent information retrieval,” SIAM Review, vol.37, no.4, pp.573–595, 1995.
[21] P. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining, Pearson Addison Wesley Boston, 2006.
[22] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ceravega, “Semantic annotation for knowledge management: Requirements and a survey of the state of the art,” Web Semantics: Science, Services and Agents on the World Wide Web, vol.4, no.1, pp.14–28, 2006.
[23] B. Adida, M. Birbeck, S. McCarron, and S. Pemberton, “RDFa in XHTML: Syntax and processing.” http://www.w3.org/TR/rdfa-syntax/, accessed Oct. 11, 2011.
[24] M. d’Aquino, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, and D. Guidi, “Toward a new generation of semantic web applications,” IEEE Intell. Syst., vol.23, no.3, pp.20–28, 2008.
[25] S. Staab, C. Braun, I. Bruder, A. Dusterhoft, A. Heuer, M. Klettke, G. Neumann, B. Prager, J. Pretzel, and H. Schnurr, “GETESS: Searching the web exploiting german texts,” Proc. 3rd Workshop on Cooperative Information Agents, pp.113–124, 1999.
[26] F.M. Suchanek, G. Kasneci, and G. Weikum, “YAGO: A large ontology-based named entity classification,” Proc. Int. Conf. on Recent Advances in NLP, pp.66–166, 2005.
[27] R. Ichise, “An analysis of multiple similarity measures for ontology mapping problem,” Int. J. Semantic Computing, vol.4, no.1, pp.103–122, 2010.
[28] L.K. McDowell and M. Cafarella, “Ontology-driven, unsupervised instance population,” Web Semantics: Science, Services and Agents
on the World Wide Web, vol.6, no.3, pp.218–236, 2008.

[29] G. Petasis, V. Karkaletsis, G. Paliouras, A. Krithara, and E.
Zavitsanos, “Ontology population and enrichment: State of the art,”
in Knowledge-Driven Multimedia Information Extraction and Ont-
tology Evolution, Lect. Notes Comput. Sci., vol.6050, pp.134–166,
Springer Berlin/Heidelberg, 2011.

[30] I.T. Jolliffe, Principal Component Analysis, Springer-Verlag, New
York, 2002.

[31] P. Hitzler, M. Krtsch, and S. Rudolph, Foundations of Semantic
Web Technologies, Chapman and Hall/CRC, Florida, 2009.

Theerayut Thongkrau is a PhD candidate
at King Mongkut’s Institute of Technology Lad-
krabang, Thailand. He received the bachelor’s
degree from the Computer Science Department
at Ubon Ratchathani University and the MSc de-
gree from Khon Kaen University. His research
interests include ontology engineering, the Se-
mantic Web, and Web services.

Pattarachai Lalitrojwong is an assistant
professor in Faculty of Information Technology,
King Mongkut’s Institute of Technology Lad-
krabang, Thailand. He received PhD in Com-
puter Science from Illinois Institute of Technol-
yogy, USA in 1999. His research interests include
artificial intelligence, cognitive modeling, auto-
mated reasoning, neural network application, in-
telligent agents, and object-oriented methodol-
gy.