Journal of Theoretical and Applied Information Technology

www.jatit.org

ASWAACC
AUTOMATIC SEMANTIC WEB ANNOTATION BY APPLYING ASSOCIATIVE CONCEPT CLASSIFIER IN TEXT

[1]Behnam Hajian, [2]Kamran Zamanifar

[1]Department of Computer Engineering, University of Isfahan, Isfahan-81746

[2]Assis. Prof., [1]Department of Computer Engineering, University of Isfahan, Isfahan -81746

E-mail: be_hajian@yahoo.com, zamanifar@eng.ui.ac.ir

ABSTRACT

After appearance of semantic web, the framework which is machine-readable and machine-understandable, by Berners Lee, current web should be annotated by W3C standards in order to define semantic domain of each word by its ontology to alleviate the posed problems in the realm of search and information retrieval. However annotation is one major problem in the semantic web domain, which is presently performed by a human agent. But, considering low human precision for this time-consuming and expensive task and the advent of data mining in recent years, in this article, a system is proposed for automatic semantic web annotation that is based on machine learning techniques for mining association rules between words in already annotated texts.

In the present article, ASWAACC annotation system will be introduced. In this system, annotation rules are produced by analysis of words co-occurrence in a paragraph to eliminate sense ambiguity of words. This system has two phases: 1-learning by mining association rules in text data which has been already annotated by human. 2-utilizing mined rules to automatically annotate the new unseen text data in order to define domain ontology of each word.

Keywords: *Ontology, Semantic annotation, Associative classifier, Machine Learning, Text Mining.*

1. INTRODUCTION

Today's World Wide Web, as the most ever-growing source of information attracts many researches to integration of web and artificial intelligence [1]. According to Berners Lee, the inventor of web, there is a gap between intelligent web and current web that is demonstrated by search engines' irrelevant results [2]. Semantic web will bridge this gap and will make the web intelligent and machine-understandable to solve proposed problem in this domain. Therefore, by utilizing semantic web, computers are capable of reasoning and making other tools like search engines cable of answering human questions by induction from semantic web databases [3], [4].

Ontology is a semantic knowledge-base about the real world that demonstrates relations between concepts in the real world. Hence, if we consider computer knowledge as some class hierarchies and attributes, semantic web will bridge the mentioned gap by insertion of some metadata in web pages that connect each word to its proper concept in ontology [5], [6]. This task, which is currently performed by human agents, is called semantic web annotation. In other words, semantic annotation means ontology population that makes new instances of ontology classes through this process [7]-[9], [22], [23].

This labor task is so expensive and time-consuming that can not be done precisely by human. As a result, a tool is needed to automate this task. Such a tool needs some common sense and artificial intelligence in order to match each word to its appropriate concept in a correct ontology because there are some ambiguities in the meaning of each word that should be resolved by such intelligent tool.

If we attend to the meaning of each word and presence of other words in sentences, we can understand a close relationship between the sense of each word and co-occurrence of other words through a sentence [4], [24]. This principle gives us a clue to reach a method for determining the real sense of each word in its context. In other words, there are some patterns existing between words and their senses which should be extracted to reach the

meaning of each word in a sentence. Data mining is a way for exploring and mining these patterns [10], [11].

Association rules mining is a data mining process that finds frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories. This process is usually used for analysis of shopping baskets in shopping malls, cross-marketing, catalog designing, loss-leader analysis, clustering, classification, etc [13],[14].

We utilize an association rules mining algorithm to explore patterns between words and their proper concepts in order to implement a machine learning system that can mine annotation rules by mining association rules between words from text data in the first phase (i.e. learning phase) of ASWAACC. In the second phase, these rules are used to annotate new text data in order to determine the ontology and sense of each word in a sentence.

ASWAACC is a machine learning-based semantic web annotation tool that learns by mining association rules among words through the text. In this system we designed an algorithm that can extract annotation rules from already-annotated text in the learning phase. For the second phase, we designed an algorithm for matching mined rules with new texts to annotate them with proper ontology. In fact, this system is a kind of associative classifier which classifies words by their proper concepts through sentences.

The rest of this paper is organized as follows: section 2 discusses related work. In the next section, architecture of ASWAACC is presented, and then the algorithms which are used in ASWAACC will be illustrated in section 4. Then, some examples and evaluations are discussed in section 5 and finally, conclusion and future work are mentioned in section 6.

2. RELATED WORKS

Regarding posed reasons in the previous section, semantic annotation is an issue which attracts many researchers to design automatic semantic annotation systems. Therefore, there are some related researches about ontology population, ontology learning, and automatic semantic annotation related to this realm of research.

Semantic annotation platforms are classified based on the type of annotation method used. There are two primary categories: Pattern-based and Machine Learning-based, as shown in Figure 1.

Machine learning-based techniques utilize two methods: probability and induction. The Probabilistic method uses statistical models to predict the locations of entities within a text. There are some other methods, like natural language processing, which use regular expression or automata to extract annotation rules and patterns within texts [15]. After the learning phase, the system utilizes mined rules for annotation of new unseen texts; an example of this category is wrapper induction.

Some kinds of systems perform annotation task within ontology population by making instances for classes [7], while some other systems, such as Magpie, just match the existing instances in the ontology with words within the text [15]. On the other hand, co-relation between entities can be extracted by some of these systems. Amilcare and SCREAM are examples of such systems which use machine learning techniques for semantic annotation. Amilcare that uses LP2 algorithm to extract annotation rules from text by regular expression is classified in this category [17]. Amilcare is used by both the Armadillo and Ont-O-Mat platforms to perform wrapper induction.

Pattern-based methods can perform pattern discovery or have patterns manually defined. These kinds of systems can perform pattern discovery, such as C-PANKOW that starts by making a hypothesis on an entity, and then the hypothesis is changed into a fact by gathering click statistics around that entity from Google. An advantage of these systems is that there is no need for text mining and learning. But these systems are not capable of performing disambiguation between classes and concepts [7], [18].

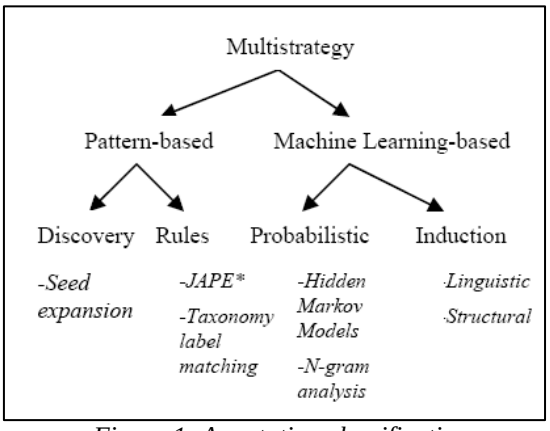

Figure.1: Annotation classification

There are a number of platforms which are implemented for semantic annotation. They utilize some kinds of systems that were mentioned above.

GATE, KIM, MnM, are such platforms as well as frameworks which can be used by researchers to perform implementation of their researches in this area [15]. Another group of annotation systems are named 'entity recognition' [25]. These systems are regularly used for annotating countries, companies or persons. Jape and Gazeteer are such tools designed to determine entities like persons, countries, cities and companies through the text. These systems are utilized by Gate to annotate entities within the texts. In Jape, rules can be identified by java language while Gazeteer utilizes lists of instances of entities for determining ontology classes of entities. Gate is a set of open source libraries and a text engineering framework, designed in the University of Sheffield for automatic semantic annotation [19], [20].

SemTag is the semantic annotation component of a comprehensive platform, called Seeker, for performing large-scale annotation of web pages. SemTag/Seeker is an extensible system, so new annotation implementations can replace the existing Taxonomy-based Disambiguation algorithm (TBD). [21].

In addition, platforms can use methods from both types of categories, called Multi-strategy, in order to take advantage of the strengths, and to compensate the weaknesses of the methods in each category. In ASWAACC we tried to utilize a data mining technique to perform word sense disambiguation by learning through extracting associations among words from already annotated texts.

3. ASWAACC ARCHITECTURE

In this section, the architecture of ASWAACC is described. If we pay our attention to the meaning of words in a sentence, we can understand that the sense of each word is tied to co-occurrence of other words in that sentence [10]. For example, if “java” occurred with “C++” or “object oriented” in a sentence, we can derive that java is a programming language. However, if this word comes with “beverage” or “caffeine”, its meaning changes into a "kind of coffee". We have used this idea in our system in order for it to learn by mining association rules among words. As mentioned before, we have designed ACC (Associative Concept Classifier) algorithm for mining annotation rules from manually-annotated text. The Architecture of ASWAACC is described in Figure 2. This architecture has two phases: learning phase and annotation phase. According to this architecture, each phase has preprocessing steps including sentence splitting, text tokenizing, eliminating stop words and prepositions, and finally indexing keywords. The first phase has an additional step, which is extracting annotations from manually-annotated documents.

The preprocessing steps are essential in this system because text data are not structured for manipulation. After preprocessing steps, in the first phase, annotation rules are extracted from annotated texts by ACC algorithm. This algorithm mines rules in the form of A→B where A illustrates a frequent set of words which frequently occurred in the annotated texts and B is an annotation object containing target word and the URI of the referenced class in an ontology. After the mining stage, these rules are stored in the database.

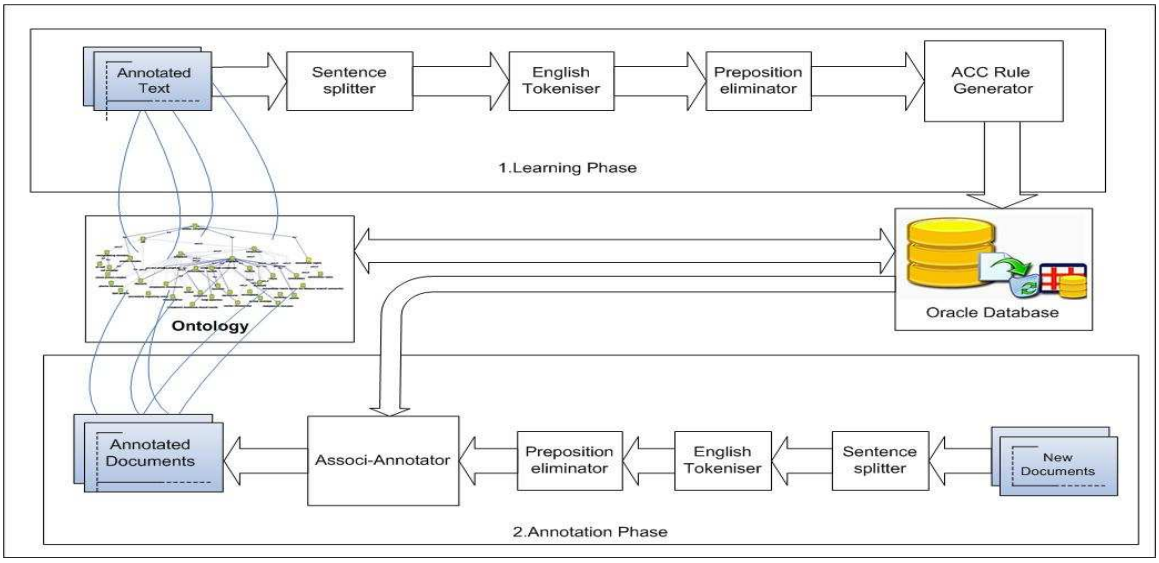

Figure.2: ASWAACC Architecture.

For the second phase we have designed Associ-Annotator algorithm that matches extracted rules with new unseen texts to annotate them with the proper ontology. In this phase, preprocessing steps are performed on new texts in order to produce word sets. In the next step, word sets are taken into the Associ-Annotator in order to match with mined rules which were stored in the database during the previous phase (As shown in Figure 2).

We used Gate framework and its libraries to implement out system. This tool contains facilities for text processing, data storing, parallel processing and information retrieval. In this system learning data and test data are stored in a Datastore which are accessible within a bounded corpus to Datastore. In addition all of processing resources are declared through a pipeline, in a way that, parallel processing is obtained. As an example, while a document is under sentence splitting, at the same time, tokenizer works on another document [19],[20].

4. OVERVIEW OF ASWAACC ALGORITHMS

According to previous section ACC algorithm and Associ-Annotator are established at the core of ASWAACC. In this section, after explanation of preprocessing steps, these algorithms will be described in details.

4.1. Preprocessing steps

As mentioned before, text is not a structured form of data for manipulation, so it is essential to convert text into the structured form of data that can be analyzed by a data mining algorithm. In ASWAACC, each text splits into sentences and then, they are tokenized into word sets. Hence, a sentence splitter and an English tokenizer, which are two components of Gate, are defined in a pipeline as processing resources.

In addition, if stop words and prepositions are not eliminated from token sets, many vain rules will be produced by ACC algorithm, since these tokens frequently occur in every sentence. Figure 3 illustrates some examples of rules which are prevented from being produced by filtering prepositions and stop words. Moreover, these rules decrease the performance of our system.

In the next step, there is a word indexer as another stage of preprocessing task, which is responsible for indexing tokens. In this stage, the position of each token will be determined in the sentence. This component also determines the index of each sentence in the text.

Document stemmer is another component that can be used in our system. But experimental results show that the stemmer decreases the performance of our system because it changes the form of tokens to their roots, while the meaning of words depends on the form they appear in a sentence. Hence, we have eliminated token stemmer from the architecture of this system.

{java , an, by } → <java, http://gate.ac.uk/owlim#programming_languages>
{java , the, oriented } → <java, http://gate.ac.uk/owlim #programming_languages>
{java , the } → <java, http://gate.ac.uk/owlim#cofee_class>
{java , and, by } → <java, http://gate.ac.uk/oawlim#cofee_class>

Figure.3. Rules that have been eliminated by filtering prepositions and stop words.

4.2. Overview of ACC algorithm

In this section, first, Apriori algorithm is discussed to show the basic concepts of association rules mining. Then, the changes that have been done on this algorithm are presented to describe our concept classifier algorithm.

Apriori is an association rules mining algorithm used to extract relations between sold items in shopping malls. In other words, this algorithm extracts rules to show which goods are sold together in what support and confidence degrees. These analyses are categorized in two major groups: mining associations among items, and sequence analysis. The first category just projects on mining association rules between items. However, sequence mining considers time and sequence order of events. Association rules are in the form of A→B, in which A and B are two sets of events or items that usually occur together in transactions. As an example, if a hammer existed in a sale transaction, this transaction would contain some pins with high probability. In this example, Apriori mines rules in the form of A→B where A = { hammer } and B = { pin }.

This algorithm was invented by Agrawal et al in 1993 [14]. Apriori is described as follow:

a) I is a collection of one or more items: I = {i1, i2, …, im} like {Milk, Bread, Diaper}
b) J = P(I) is a set of all subsets of I. in other words, the elements of J are called item-sets.

c) t is a set of items which appear in one event, like purchasing some goods from market. (transaction) t⊆I.
d) $T = \{t_1, t_2, \ldots, t_m\}$ is a transaction set.
e) An association rule is an implication of the form: X→Y, where X, Y are disjoint subsets of I (i.e. elements of J) [12-14]. In other words X → Y is an association rule, where X, Y ⊂ I, and X ∩ Y = ∅ [14].

Apriori: Apriori finds all rules having support and confidence greater than minimum support and minimum confidence thresholds which are specified by user. That means this algorithm extracts any rule in a way that, if X appears in a transaction, Y will occur in that transaction by the confidence probability ratio.

Support: Support is a fraction of transactions that contain both X and Y.

$$Support = \frac{\text{number of tuples containing both (A \& B)}}{\text{total number of tuples}} = P(AUB) \quad (1)$$

Confidence: Confidence measures how often items in Y appear in transactions which contain X.

$$Confidence = P(B \mid A) = \frac{P(AUB)}{P(A)} \quad (2)$$

$$Confidence = \frac{\text{number of tuples containing both (A \& B)}}{\text{number of tuples containing A}} \quad (3)$$

Frequent Item set: An item set whose support is greater than or equal to a *minsup* threshold.
Apriori has two-steps:

1. Frequent Itemset Generation (i.e. Fig 4):
   Generating all itemsets whose support ≥ minsup

2. Rule Generation (i.e. Fig 5):
   Generating high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset.

```
Algorithm Generate-frequent-set(T)
C1 ← init-pass(T);
      F1 ← {f | f ∈ C1, f.count/n ≥ minsup};
                // n: no. of transactions in T
      for (k = 2; Fk-1 ≠ ∅; k++) {
         Ck ← candidate-gen(Fk-1);
         for (each transaction t ∈ T) {
            for (each candidate c ∈ Ck) {
               if (c is contained in t)
                  c.count++;
            }
         }
         Fk ← {c ∈ Ck | c.count/n ≥ minsup};
      }
return F ← Yk Fk;
```

Fig.4: Frequent Itemset Generation in Apriori [12].

```
for (each frequent itemset X)
   for (each non-empty subset A of X)
      for (each B ∈annotation set , annotates Wk)
         if(Confidence(A→B) ≥ minconf )
            A → B will be extracted

Confidence(A→B)=
            probability(A&B)/probability(A)
```

Fig.5: Rule Generation in ACC algorithm.

It is obvious that this algorithm mines rules which have a set of items on both sides. This algorithm was changed for our system in a way that the left hand side of each rule is a set of words, but the right hand side is an ordered pair including target word and a referenced class from an ontology. This algorithm is described by the following definitions:

a) $W = \{w_1, w_2, \ldots, w_m\}$ is a word set.
b) c ∈ Ontology class.
c) $A = \{w_j, w_{j+1}, w_{j+2}, \ldots, w_k, \ldots, w_L\}$ A⊆W is a frequent word set.
d) B= <W_k,c> is an annotation object.
e) W_k is target word
f) Tk: a transaction set that contains all paragraphs which include W_k.
g) N_k : cardinality of T_k.
h) P(A) is the probability which A occurs in a text.
i) P(A&B) is frequency of all situations which A occurred within the paragraph while W_k has been annotated to class c of ontology.

j) In this system each paragraph is considered as a transaction.

k) Support and confidence are defined as follow:

$$Support(A) = \frac{\text{number of texts containing A}}{\text{number of texts containing } w_k} = \frac{P(A)}{N_k} \quad (4)$$

$$confidence(A \rightarrow B) = P(B \mid A) = \frac{P(A \& B)}{P(A)} \quad (5)$$

ACC algorithm mines rules in form of A→B from learning data, in a way that A is a frequent words set, and B is an annotation object which annotates the word w_k to the class c of an ontology. Hence, $A = \{w_1, w_{1+1}, w_{1+2}, .., w_k, .., w_m\} \rightarrow < w_k, c >$ is an annotation rule, which is stored in the database.

Fig 6 demonstrates example of some rules which were extracted by the above algorithm.

{java , programming, language} → <java, http://gate.ac.uk/owlim#programming_languages>
{java , c++, c#, object, oriented} → <java, http://gate.ac.uk/owlim#programming_languages>
{java , cofee } → <java, http://gate.ac.uk/owlim# Coffee_Class >
{java , tee } → <java, http://gate.ac.uk/owlim# Coffee_Class >
{java , caffeine } →<java, http://gate.ac.uk/owlim#Coffee_Class >

Fig. 6: Example of extracted rules

4.3. Overview of Associ-Annotator algorithm

This system learns by mining and storing the rules which ACC extracts from manually-annotated texts. The system utilizes these extracted rules for annotating new texts by Associ-Annotator algorithm. This algorithm tries to match the left hand side set of each rule with the words that have been used in the texts.

This phase contains preprocessing steps which have been illustrated before. After preprocessing steps, token sets are given to the Associ-Annotator algorithm in order to match with extracted rules. In this algorithm, a rule is fired whenever all words in its left hand side set, appear in the text. When a rule is fired, the target word in the text will be annotated to the concept class which exists at the right hand side of the fired rule. In other words, this algorithm classifies each word into the proper concept class of ontology. In this algorithm:

a) Y is a set of words that have appeared in the text.

b) X is the left hand side set of a rule.

In order to find rules that match with a text, it is sufficient to check whether X is a subset of Y. This algorithm is presented in fig 7 in more details.

```
Algorithm AssociAnnotator(Text T){
    Tokset ← Preprocess(T);
    Y← Tokset;
    For each (rule in ruleset) {
        If( W_k is on right hand side of rule)
            Add rule to targetruleset;
    }
    For each (Rule in targetruleset){
        X ← left hand side set of Rule
        If(X ⊆ Y){
            Rule is fired;
            Return;
        }
    }
}
```

Fig.7: Associ-Annotator algorithm.

In order to decrease the redundant matching and to increase the speed of annotation process, some excess rules should not be matched; hence, the rules which contain the target word on their right hand side are selected and are added to the target rule set. Then Associ-Annotator verifies if the left hand side of any of them is the subset of the paragraph that should be annotated in the text.

5. EXPERIMENTAL RESULT, DISCUSSION AND ASWAACC EVALUATION

This section discusses about evaluation of ASWAACC and experimental results. For this aim, ASWAACC learned by using 200 manual annotated documents which have already been annotated by human agents. In the next phase, ASWAACC was tested on 1000 news text documents; then the results were compared with Onto-Gazeteer system which is a Gate component that annotates documents with lists of instances bound to ontology classes. In addition, this component is not capable of performing word sense disambiguation because it just performs annotation by matching words and instances in the lists. So, this component usually produces more than one annotation for one word.

In order to make a precise evaluation for this system, 200 documents which were already annotated by hand, were annotated by ASWAACC and the results were compared with each other. We

consider two criteria for evaluation of this system: recall and precision, which are defined as follows [10].

$$recall = \frac{\text{accurate results produced by system}}{\text{all accurate answers within text}} \quad (6)$$

$$precision = \frac{\text{accurate results}}{\text{accurate results} + \text{inaccurate results}} \quad (7)$$

ASWAACC has two input parameters that affect its performance in the two mentioned criteria. These parameters are minimum support and minimum confidence thresholds for ACC algorithm in the learning phase.

With attention to experimental results achieved from this system, we realize that minimum support value should be determined according to the size of the input text because candidate sets are produced by considering support value of each set and minimum support threshold. However, in large texts, we have larger state space of words, so coherence among words would decrease. In this situation high minimum support value may prevent production of many important rules which causes reduction in recall value because in equation (4) N grows and support value of all sets would decrease, and as a result, many candidate sets would be eliminated. On the other hand, confidence value of each rule involved the accuracy of system. In addition, high minimum confidence threshold causes production of precise rules, but it decreases recall value. Similarly, very low minimum support value leads to production of many irrelevant rules and it results in decrease in precision but increase in recall. These relations are demonstrated in Fig 8. On the other hand, low minimum support threshold increases learning time due to increase in production of many candidate sets by ACC algorithm.

Therefore, it is significant to determine the minimum support value regarding text size, number of annotated words in the text, and the number of learning data. The experimental results show the best performance of this system is achieved by setting support value to 10-15% and confidence threshold to 65-70%. Regarding these parameter values and our test data repository, the system accuracy is measured as 91% and 88% for precision and recall value, respectively. This result can be changed by tuning mentioned parameters in the system.

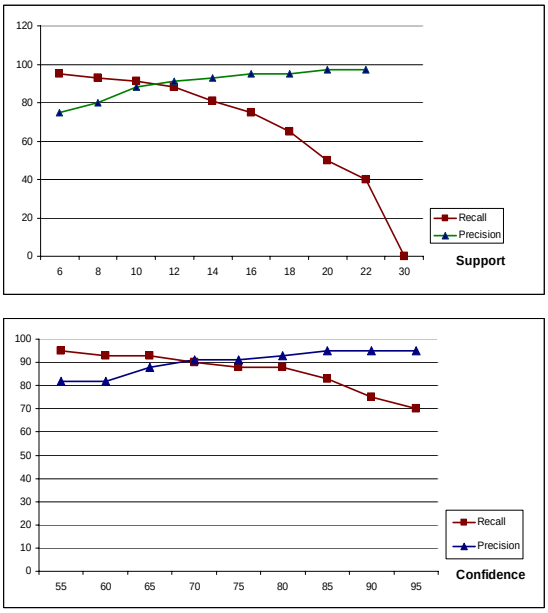

Fig. 8: Relation between minimum confidence and minimum support regarding recall and precision of ASWAACC.

Another matter to concentrate about is that in some cases, fired rules conflict with each other. In these cases, the system should decide and select the appropriate rule for annotation of the target word. This situation occurs when more than one rule are fired for annotating a word in the text and the right hand sides of fired rules contain the same word with different ontology references. In these situations, ASWAACC selects the rule with a higher confidence value. But in a case of equality of confidence values of conflicted rules, support value is the next priority for selecting the proper annotation rule.

Quality and performance of every machine learning system depend on the quality of learning data. Minimum support and minimum confidence are other important parameters for the performance of this system. Therefore, we can achieve better results by higher quality learning data and tuning values for minimum confidence and minimum support thresholds. The following picture demonstrates the schema of ASWAACC annotation system with Gate platform.

Fig. 9: Schema of ASWAACC and Gate.

6. CONCLUSION AND FUTURE WORK

This article discussed the system that performs word sense disambiguation by applying association rules mining technique in the text. The idea which has been used in this system is mining the relation of words' meaning and co-occurrences of other words in the sentences by a machine learning concept classifier. A key point for this system is that co-occurrence of words is less affected by the style of writer than other NLP systems.

This system eliminates ambiguities in words' meaning by utilizing extracted rules. Precision and speed are important properties of ASWAACC that is dependent on the values of minimum confidence and minimum support which are determined by the user.

This system can be installed as a plug-in on Gate platform in order to perform word sense disambiguation task in annotation. Consistency with non-English languages is another important property of ASWAACC because semantic relation among words is not only specific for one language, but also is an important issue in any language. In addition, this system can utilize crawler or search APIs facilities from Gate plug-ins in order to crawl the internet and annotate crawled web pages.

This system can be utilized in semantic search engine in order to annotate crawled web pages and index them semantically with appropriate ontology. Such a system can eliminate irrelevant pages from search engine results by classifying each word by its ontology semantically. Utilizing this system in information retrieval or a semantic search engine can form part of our future work.

REFRENCES

[1] Lotfi A. Zadeh, 2004. "A note on web intelligence, world knowledge and fuzzy logic", Data & Knowledge Engineering, Vol.50, pp. 291–304.

[2] Berners-Lee, T., Hendler, J. and Lassilo, 2001. "The Semantic Web", Scientific American, pp.34-43.

[3] Jacob Ko¨ hler, Stephan Philippi, Michael Specht, Alexander Ru¨egg, 2006. "Ontology based text indexing and querying for the semantic web", Knowledge-Based Systems vol.19, pp.744–754.

[4] Mingxia Gao, Chunnian Liu, Furong Chen, 2005. "An Ontology Search Engine Based on Semantic Analysis", at Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) IEEE.

[5] Quan Thanh Tho, Siu Cheung Hui, Fong, Tru Hoang Cao, 2006. "Automatic Fuzzy Ontology Generation for Semantic Web", IEEE transactions on knowledge and data engineering, vol. 18, No. 6.

[6] Chang-Shing Lee , Yuan-Fang Kao , Yau-Hwang Kuo , Mei-Hui Wang, 2007. "Automated ontology construction for unstructured text documents", Data & Knowledge Engineering, vol.60, pp.547–566.

[7] Philipp Cimiano, 2006. "Ontology Learning and Population from TextAlgorithms, Evaluation and Applications", Springer, pp.238-271.

[8] Hend S. Al-Khalifa, 2007. "Automatic Document-level Semantic Metadata Annotation using Folksonomies and Domain Ontologies", PHD thesis, University of Southampton.

[9] Jelena Jovanovich, Dragan Gaševic, Vladan Deved, 2006. "Ontology-Based Automatic Annotation of Learning Content", Int'l Journal on Semantic Web & Information Systems 2, pp.91-119.

[10] Michael W. Berry, 2004. "Survey of Text Mining Clustering, Classification, and Retrieval", Springer-Verlag New York, Inc, Scanned by Velocity, pp.173-181.

[11] Hany Mahgoub, Dietmar Rösner, Nabil Ismail and Fawzy Torkey, 2007. "A Text Mining Technique Using Association Rules Extraction", international journal of computational intelligence volume.4 No.1.

[12] Rakesh.Agrawal, Ramakrishnan.Srikant, 1997. "Fast Algorithm for Mining Association Rules", IBM Almaden Research.

[13] Rakesh Agrawal, Ramakrishnan Srikant, 1997. "Mining generalized association rules", IBM Almaden Research, Future Generation Computer Systems Volume 13, Issues 2-3, pp. 161-180

[14] Rakesh Agrawal, Tomasz Imielinski, Arun Swami, 1993. "Mining Association Rules between Sets of Items in Massive Databases", Int'l Conf. on Management of Data Center, USA.

[15] Lawrence Reeve, Hyoil Han, 2005. "Survey of Semantic Annotation Platforms", Proceedings of the 2005 ACM symposium on applied computing.

[16] J. Domingue, M. Dzbor, E. Motta, 2004. "Magpieie: Supporting Browsing and Navigationon Semantic Web" In Proceedings of ACM Conference on Intelligent UserInterfaces (IUI), pp: 191-197.

[17] Fabio Ciravegna, 2001. "(LP)2, an Adaptive Algorithm for Information Extraction from Web-related Texts", 17th International Conference on Artificial Intelligence (IJCAI-01), Seattle.

[18] Philipp Cimiano, Gunter Ladwig, Steffen Staab, 2005. "Gimme' the Context: Contextdriven Automatic Semantic Annotation with CPANKOW", International World Wide Web Conference Committee (IW3C2).

[19] Hamish Cunningham,Diana Maynard,Kalina Bontcheva, 2007. "Developing Language Processing Components with GATE Version.4 (a User Guide)", The University of Sheffield.

[20] Diana Maynard, Valentin Tablan, Hamish Cunningham, Cristian Ursu, Horacio Saggion, Kalina Bontcheva, Yorick Wilks, 2002. "Architectural elements of language engineering robustness", Journal of Natural Language Engineering Data.

[21] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R.Guha, Anant Jhingran, Tapas Kanungo, Kevin S.McCurley, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, Jason Y. Zien, 2003. "A case for automated large-scale semantic annotation", Web Semantics: Science, Services and Agents on the World Wide Web, vol.1, pp.115–132.

[22] Atanas Kiryakov, Borislav Popov, Ivan Terziev, 2004. "Semantic annotation, indexing, and retrieval", Web Semantics: Science, Services and Agents on the World Wide Web vol.2, pp.49–79.

[23] Alireza Mansouri, Lilly Suriani Affendey, Ali Mamat, 2008. "Named Entity Recognition Using a New Fuzzy Support Vector Machine", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.2.