A Framework for Semantic Interpretation of Noun Compounds Using Tratz Model and Binary Features

Ahmad Zaeri and Mohammad Ali Nematbakhsh

Semantic interpretation of the relationship between noun compound (NC) elements has been a challenging issue due to the lack of contextual information, the unbounded number of combinations, and the absence of a universally accepted system for the categorization. The current models require a huge corpus of data to extract contextual information, which limits their usage in many situations. In this paper, a new semantic relations interpreter for NCs based on novel lightweight binary features is proposed. Some of the binary features used are novel. In addition, the interpreter uses a new feature selection method. By developing these new features and techniques, the proposed method removes the need for any huge corpuses. Implementing this method using a modular and plugin-based framework, and by training it using the largest and the most current fine-grained data set, shows that the accuracy is better than that of previously reported upon methods that utilize large corpuses. This improvement in accuracy and the provision of superior efficiency is achieved not only by improving the old features with such techniques as semantic scattering and sense collocation, but also by using various novel features and classifier max entropy. That the accuracy of the max entropy classifier is higher compared to that of other classifiers, such as a support vector machine, a Naïve Bayes, and a decision tree, is also shown.

Keywords: Noun compounds, semantic relation, multiclass classifier, feature selection, semantic features.

Manuscript received Oct. 23, 2011; revised June 25, 2012; accepted July 5, 2012.

Ahmad Zaeri (phone: + 98 917 7711655, ahmadzaeri@gmail.com) and Mohammad Ali Nematbakhsh (nematbakhsh@eng.ui.ac.ir) are with the School of Engineering, University of Isfahan, Isfahan, Iran.

http://dx.doi.org/10.4218/etrij.12.0111.0673

I. Introduction

In recent years, interpreting the nature of semantic relations between the components of noun compounds (NCs) has received a great deal of attention, both in computational linguistics and in information retrieval, due to the key role of this process in applications such as question answering, building semantically rich ontologies, text summarization, information extraction, and machine translation. NCs are made up of more than one part (two parts in this paper), usually consisting of a head noun with one or more preceding noun modifier. The problem is how to interpret the semantic meaning of the relationship between a head noun and its modifier(s). For example, this relation could be a whole-part relation, as in "car engine," a consumption relation, as in "petrol engine," or a purpose relation, as in "search engine."

The interpretation of NCs is a difficult task because there is no limit to the possible number of NCs and the number of semantic relations (SRs) between their constituent parts [1]. Moreover, NCs should usually be interpreted without any contextual information. All the information should be extracted from the head and modifier, which makes the use of context-based methods impossible [2].

Most of the existing approaches semantically interpret the relationships of NCs by classifying them into a fixed taxonomy of SRs. Some studies suggest coarse-grained and generalizable taxonomies [3], whereas others propose fine-grained taxonomies [4]. There are two main challenges in establishing an effective taxonomy of semantic relations and an annotated NC data set. One challenge is to establish coverage of an NC data set [5], and the other is to examine issues related to the compatibility of SRs with regard to the application of interpretation tasks. Therefore, creating and maintaining a

standard and universal taxonomy of semantic relations is very likely to remain a demanding task [1], [6].

Tratz and Hovy [4] established a fine-grained taxonomy of semantic relations, consisting of 44 classes. This was done by integrating previous taxonomies and by producing the largest set of annotated NCs yet created, which includes 17,509 NCs. We will refer to this data set as "the Tratz data set."

However, due to the lack of contextual information regarding the semantic interpretation of NCs, the state-of-the-art approaches [4], [7] tend to use huge corpuses such as Google Web 1T 5-gram or the British National Corpus to extract contextual information about the usage of NCs. The size of such corpuses, including their indexes, exceeds hundreds of gigabytes, which makes them unusable in resource-limited applications (for example, mobile computing and robotics).

In this paper, we propose a set of novel features with low computational overhead to be used instead of features that require large corpuses. Additionally, a new selection algorithm is applied to the generated instances of the features to improve classification accuracy and efficiency. In the proposed method, a maximum entropy classifier outperforms other classic classifiers. For implementation of the proposed method, a plugin-based/modular framework is presented, which is easy to expand, configure, and test. The method is an improvement on the Tratz model [4] and is herein evaluated by a number of trials. The evaluation shows that the proposed method performs with a high degree of accuracy, surpassing the best established approaches, which use large external resources, such as text corpuses.

The remainder of the paper is divided into six sections. Section II discusses previous works. Section III introduces the new approach, its applied features, and the feature selection algorithm. Section IV presents our implementation of the approach, based on a new plugin-based multilayer architecture. Section V outlines our experiments, evaluations, and comparison with other approaches, using the largest and most current NC data set. Our conclusion is presented in section VI. Finally, section VII is devoted to some directions for future work.

II. Related Work

There are a large number of approaches to automatic semantic relations classification in the existing literature. Rosario and Hearst [8] used a neural network and a lexical hierarchy to generalize a training set. Moldavan and others [9] introduced a new learning approach (known as "semantic scattering"), along with sense collocation, as a means of achieving the semantic generalization of NCs. The authors used this approach to create a semantic space. Kim and Baldwin [10] used a WordNet [11] similarity measure to connect new NCs in terms of their corresponding semantic relations, by finding the most similar annotated NCs. Nastase and others [12] utilized three different types of machine learning algorithms (memory-based, decision tree, and support vector machine [SVM]) with features extracted from WordNet and large text corpuses (for word sense disambiguation [WSD]). They finally concluded that the accuracy of each learning algorithm depends on the class of semantic relations being utilized. Girju [13] investigated the use of cross linguistic resources in terms of NC semantic interpretation. Kim and Baldwin [14] used a mechanism called "co-training" to extend the space of each semantic relation by using WordNet synonym hypernym relations. Huang and others [15] introduced two new concepts of probabilistic and semantic relatedness for use in semantic relations classification.

Ó Séaghdha and Copestake [16] combined the use of lexical and relational similarity in a kernel-based framework, to establish a state-of-the-art approach. They used the British National Corpus [17] and the Web 1T 5-gram [18] to extract co-occurrence and contextual information. Tratz and Hovy [4] used a maximum entropy classifier with a large number of Boolean features, including WordNet features, Roget's Thesaurus [19] features, and usage features extracted from Google Web 1T 5-gram. They argued that their approach produces the best results among current methods.

In summary, the best previous approaches use a huge corpus like Google Web 1T 5-gram or the British National Corpus and do not consider the efficiency of the features and techniques used. To solve this problem, we propose a new approach that, by using efficient feature sets, produces results that are better than the best previous approaches in terms of reported accuracy. Our main contributions in this paper are as follows: a large number of new lightweight features, a new feature selection algorithm, and a plugin-based architecture for efficient use of features and ease of combination.

III. Approach

Figure 1 illustrates the proposed approach as a framework. It shows the overall structure, building blocks, and the flow of data transferred between building blocks. Generally, each building block can be implemented with different approaches. We first describe the role of each part of this framework in order of the data flows. Additionally, in the next sections, we fill each building block with a suggested solution to make the final architecture of our approach. The data flow of the framework is divided into two parts; that is, training time data flow and runtime flow, as is usual for most machine learning-based systems.

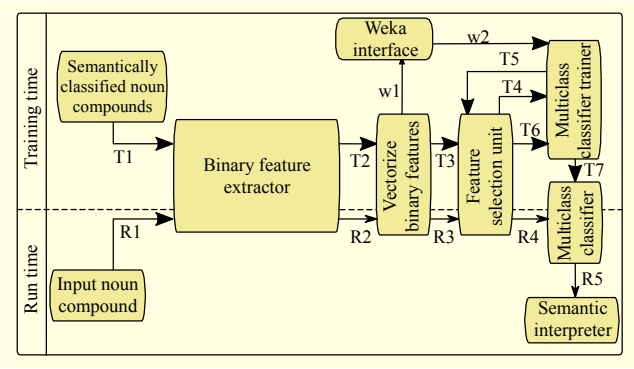

Fig. 1. Lightweight semantic relation interpreter.

To train the system, we need a taxonomy of semantic relations, many of which are in the literature, from very coarse grain to fine grain. The selection of taxonomy, however, depends on the application of the NC interpreter. The system then needs the training data set that includes a set of NCs annotated by its class in a selected semantic relations taxonomy.

This data set is then sent to the binary feature extractor. In this section, the NCs will be analyzed semantically and syntactically and will then be enriched by a large number of binary features (in the current implementation, every NC is enriched by an average of 120 features). All the features are binary, which means that every feature can be set to 1 or 0 (for example, "tree is the hypernym of N2," which will be set to 1 for the hyponyms of "tree" and 0 for others).

The feature-enriched data set is then passed to the vectorization unit. In this section, the system creates the features' alphabets and allocates a code for every distinct feature. After this coding is finished, the system feature space is created. Due to the use of binary features, the resulting space feature is very large (with dimensions of 12,000 to 20,000 in our experiments). The system now represents each instance by using a set of 1 for existing features and 0 for the absence of such features.

The vectorized data set is sent to the feature selection unit (T3). In this unit, the data set is sent to a primary classification (T4) and the classified results return to the feature selection unit. In this section, the classified instances will be sorted by different classification quality measures. When these results are collected, the unit decides to select a subset of space dimensions (features alphabets). This has an important role in improving both the accuracy and the efficiency of the system.

The vectorized and feature reduced data set is transferred to the classifier trainer (T6). The system uses maximum entropy as a multiclass classifier (the comparative results with other common classification algorithms like SVM and Naïve Bayes are shown in the evaluation section). After finishing the classifier training, the resulting classifier and its alphabets are used in the runtime.

The input NC has been interpreted according to R1 through R5 in the runtime. The system will extract all the binary features from this input NC before vectorizing the resulting binary features, but, at this stage, the system will not change the space dimension because the alphabets are frozen during runtime (R2). In the feature selection (R3), the system just applies the saved feature selection mask to this new instance of NC. The classifier will then classify this new instance to one of the predefined semantic relation classes (R4). At this stage the semantic interpreter knows the semantic relations between the head noun and its modifier and can react accordingly, for example, by setting out some descriptions or changing and mapping the classifications of other representations.

In the rest of this section, important aspects of this approach, such as used feature sets and feature selection, will be discussed in more detail.

1. Feature Sets

We have used some features from the literature and have developed many new features to improve accuracy and replace heavy and resource-consuming features (for example, the 4-gram feature). The following features are novel features.

A. Semantic Features Based on WordNet

WordNet is one of the most widely used semantic resources in linguistics. In WordNet, synsets are divided into four subgroups based on four parts of speech (POSs): nouns, verbs, adjectives, and adverbs. To access a word synset, it is first necessary to determine its POS. Word polysemy is another difficulty associated with synsets, and it can make the task of identifying the real synset of modifiers and the head noun in an NC very challenging [20]. By using a Stanford POS tagger [21], the POS of each NC constituent can be determined. For example, "shoe box" is distinguished as a noun-noun, while "black box" is an adjective-noun. There is also a difference between the granularity of the Stanford tagger output format and the four accepted WordNet POSs. The approach uses mapping between these two different POS formats.

A pair (synset1, synset2) should be calculated as the disambiguated form of the (w1, w2) NC. For example, consider that we have (n1, n2) as a (noun, noun) compound after POS tagging. Assume that there are six different synsets for n1 and four synsets for n2. Using all these synsets may overwhelm the feature's space. Therefore, it is required to choose one synset for each word. The framework considers an abstract WSD service to select the best sense for each word. WSD service based on available contextual information can be implemented by two approaches. First, if the system is used in a specific domain, such as ontology matching or user query

Table 1. Choosing first WordNet sense as disambiguated meaning of word can achieve better results compared to selecting other senses.

Number of selected WordNet sense	Accuracy
1	0.6008
2	0.5124
3	0.4705
4	0.4511

Table 2. Effects of applying different hypernym subset selection strategies for generating hypernym multiplication features on system accuracy and efficiency.

Combinations	Accuracy	Generated features
All hypernym	0.5814	192,728
Third most general hypernym	0.5973	16,005
Third most specific hypernym	0.6008	21,813
With index multiply of 3	0.5925	23,999

answering, it can utilize such information (domain ontology or user preference) in addition to a WSD algorithm. Describing such domain specific disambiguation techniques is out of the scope of this paper. Second, in the lack of such contextual information, the system can only rely on n1 as the contextual information for n2 and vice versa. For example, consider that the NC is "bank account." WordNet suggests eight different senses for the word "bank," the first and second of which are "sliding land" and "financial institute." The system can use the word "account" to decide that the second sense ("financial institute") is more relevant. One common way to find such relatedness is to use statistical analysis of a large corpus such as Web 1T 5-gram [18]. Implementing WSD service without using a large data corpus should be considered in future works.

Nonetheless, the experiment results represented in Table 1 show that using the first sense, that is, the most common sense, can achieve better results than using other senses. The result is based on the Tratz data set and 10-fold cross-validation. The first column shows the number of WordNet senses that have been used by a WSD service in each experiment, and the second column shows the accuracy achieved by using a corresponding sense.

Some WordNet features need to identify the WordNet similarity between two words. The semantic similarity of two words, sim (w1, w2) is represented as a real number between 0.0 and 1.0. This number is calculated by averaging different WordNet similarities from the WordNet similarity library [22].

Here, we summarize the proposed WordNet features:

- Synset IDs of synset1 and synset2: this will add the disambiguation feature to the NC.
- Full hypernym words of synset1: "full" signifies that the system uses a level order algorithm for traversing hypernym links from synset1 and collects all hypernym synsets of synset1. Then, it extracts all the synonyms in each hypernym synset.
- Partial multiplication of the third of the full hypernyms of synset1 and synset2 that are the most specific: this feature will generate many similar compounds to the NC; for instance, by combining a more general form of w1 (derived from the synset1 full hypernym list) and w2 (derived from the synset1 full hypernym list), so inspiring the idea of co-training and sense collocation [14]. The generated features can be very significant in relating the current instance to other seen and unseen NCs, particularly considering that the system only uses binary features. When using binary features, even very similar NC features can be coded totally differently because they are unrelated in a feature space (feature alphabet). However, using such features can create specific difficulties due to the large number of hypernyms needed for every sysnset. In WordNet, each synset could have between one and 17 hypernym synsets. If it is supposed that there are six on average, there will be (6×6)=36 synset combinations for each disambiguated NC. In addition, if each synset has just three synonyms on average (resulting from system experiments), the final combination could be (6×3)×(6×3) =324 different general forms of NCs. This would overwhelm the feature space, so it is critical for achieving accuracy and acceptable efficiency to select only a subset of this large number of combinations. To do this, four different scenarios have been considered: 1) selecting all the hypernyms of each synset, 2) selecting the third most general hypernym of each synset, 3) selecting the third most specific hypernym of each synset, and 4) selecting only hypernyms with an index divisible by three (to cover the full range). Table 2 shows the effects of applying different hypernym subset selection strategies for generating hypernym multiplication features regarding system accuracy and efficiency. The first column shows the strategy that has been used in the experiment and the second column shows the accuracy that is achieved by exploiting the strategy. The third column represents the number of generated features for measuring the efficiency of the strategy. This table shows that the third scenario achieves the best results in terms of accuracy and efficiency in the respect that the number of its generated features is relatively low. Therefore, the system will use the third scenario to only generate 36 out of 324 features on average. The results have been achieved by using the Tratz data set and 10-fold cross-validation.

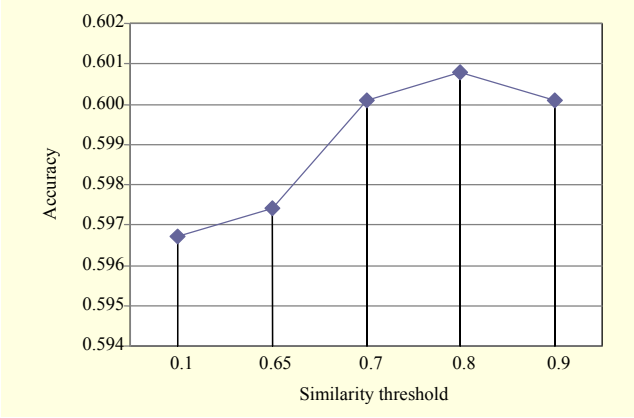

Fig. 2. Accuracy of classification by changing similarity threshold in glossary related features.

- Synset1 gloss terms that are similar to w2: this feature selects the terms in the synset1 gloss that are similar to w2, which helps extract more information about the relatedness of two constituents of an NC. As mentioned previously, the semantic similarity of two words is represented as a real number between 0.0 and 1.0. It is calculated by averaging four different WordNet similarities. This feature employs a predefined threshold value between 0.0 and 1.0 to find two similar words. Two words are considered similar if their calculated similarity is higher than a predefined threshold value. Figure 2 shows the results of using different threshold values on the accuracy of a classifier using this feature. Axis *x* shows the threshold value which is selected to be between 0.0 and 1.0. Axis *y* shows the classification accuracy that has been achieved using the threshold. The experiments show that features using the reverse relation (synset2 glossary terms that are similar to w1) have no positive effect on the results of the interpretation. This could be the consequence of the importance of head noun w2.

B. WordNet Relation Features

- X is the topic of synset2: X is a synset considered by WordNet to be the topic of synset2. This relation has been extracted by using WordNet's semantic TOPIC-type link.
- Synset2 has the attribute x: there are some synsets represented as x that could be used as adjectives for synset2 (for example, being "good" or "bad" for a product).
- Synset2 is a member of x: synset2 (head noun) is a member of synset x considering the WordNet HOLONYM-MEMBER link (for example, "aerospace" is a member of "army").
- Synset2 is a part of x: synset2 (head noun) is a part of synset x, when considering the WordNet HOLONYM-PART link (for example, "wheel" is a part of "car").
- Synset2 is a substance of x: synset2 is a substance of x, according to the WordNet HOLONYM-SUBSTANCE link (for example, "protein" is a substance of "egg" and "fish").
- Synset1 is a part of synset2: the MERONYM-PART links of WordNet are recursively searched from synset1 and if synset1 finds synset2, it will add this feature to show that synset1 is a part of synset2.
- Synset2 is a part of synset1: the reverse of the above feature.

C. Roget's Thesaurus Division Features

Roget's Thesaurus is a widely used English thesaurus, composed of six primary classes. Each class is divided into sections and subsections. Each division includes semantically related and similar words, so this system uses Roget's Thesaurus (class' number, section's number, and subsection's number) to categorize words. Each word may be in a different category at the same time. The system uses a digitized form of Roget's Thesaurus, the Electronic Lexical Knowledge Base (ELKB) library [23]:

- Roget's ClassNum:SectionNum:subSectionNum for w1.
- Roget's ClassNum:SectionNum:subSectionNum for w2.

Other than these new features, we also use some more WordNet and surface features from Tratz and Hovy [4] as follows.

D. More WordNet Features

- All synonyms of synset1 and synset2.
- Intersection of hypernyms of synset1 and synset2.
- Gloss terms of synset1 and synset2: the tokenized and reduced gloss terms of synset1 and synset2.
- Intersection of synset1 and synset2 gloss terms.
- Link types of synset1 and synset2.
- Lexicographer file for synset1 and synset2.
- Indicator as to whether an entry exists for a compound as a single, separated term (for example, "junk mail" or "farm worker").
- Indicator as to whether an entry exists for a compound as a single, solid term (for example, "birthrate" or "leatherjacket").
- Indicator that synset1 is hypernym of synset2 or vice versa.
- Word w1 existing in the glossary of synset2 or vice versa.
- All x POSs existing for w1 and w2.

E. Surface Features

- Prefix categories of w1 and w2.
- Indicators of whether a preposition occurs within w1 or w2.
- Meaning of w1 and w2 suffixes.
- Two and three last letters of w1 and w2.

For generating glossary-related features, the glossary of each

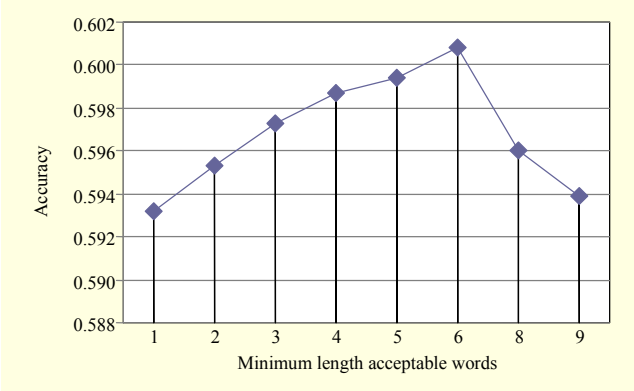

Fig. 3. Classification accuracy improvement due to removing words with less length than predefined value in extracting WordNet glossary features.

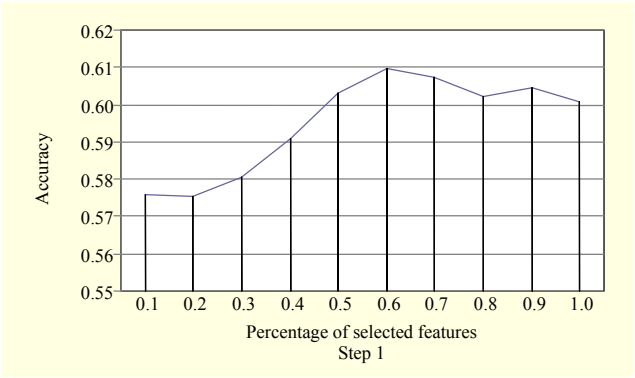

Fig. 4. Result of first step of search for selecting good percentage value for feature selection algorithm.

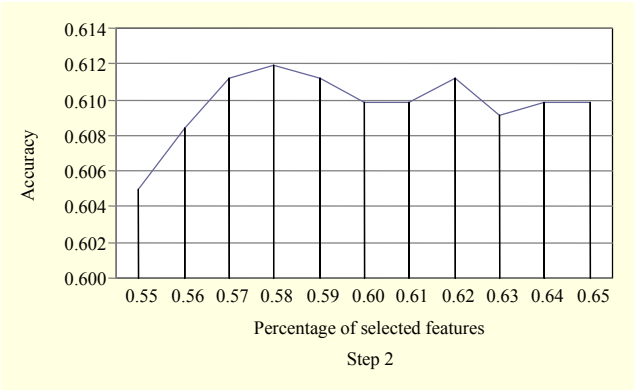

Fig. 5. Finer step 2 search for fraction of feature selection shows that 0.58 is good estimation for given data set.

synset can be tokenized and all English stop words of a lesser value removed. Even the reduced and tokenized glossary suffers from the problem of "multiplication full hypernyms" features, so the system just selects the gloss words that have a length of more than six. Figure 3 shows that the elimination of the words with lengths less than a predefined value will improve the impact of the WordNet glossary features plugin on classification accuracy. The axis *x* shows the minimum length limit that is used to eliminate the short words. The result is achieved by running the implemented approach on the Tratz data set without using the feature selection unit.

2. Feature Selection

The feature extractors generate a large number of different binary features (in our experiments, the number is between 38,000 and 208,000). Thus, discarding less important features will result in a significant improvement in both the accuracy and the efficiency of the system.

This unit uses two measures. Firstly, the frequency is joined with Bi-Normal Separation (BNS) [24] to sort the feature spaces by their importance. Then, a more significant fraction of features is selected by using a threshold. There are many feature selection measures, but the experiments show the advantages of using the frequency and BNS combination (see Table 6).

The threshold is selected by a heuristic search because checking a value for a threshold is a very time-consuming task. The heuristic search is carried out in four steps. Firstly, the unit calculates the accuracy of classification just by using selected percentages (10, 20, …, 80, 90) as the thresholds of more significant features and selects the best threshold (for example, 60%). Secondly, it will check for better thresholds closer to the previously selected threshold. Figures 4 and 5 illustrate these steps in a heuristic search of a given data set. As shown in Fig. 4, the result of the step 1 search for selecting a fraction of features shows that 0.6 is a good estimation for the first step. Here, the axis *x* represents the percentage of the features that will be selected by the feature selection algorithm as the more effective features. The axis *y* shows the classification accuracy, which is achieved only by using those selected features.

IV. Implementation

The approach is mainly implemented using a MALLET framework [25] but has an interface with a Weka data mining package [26]. This interface converts all vectorized instances in the data set to a Weka ARFF file format that can be processed by Weka, and, if it finds a better classifier, the system can use this as its main classifier.

The binary feature extractor is the most important part of the system because the precision and the efficiency of the system depend mainly on how this performs. Due to its importance, this part has been designed and implemented to provide high modularity and ease of configuration with a three-layered architecture. It can be restructured and extended for all three layers to reflect the needs of different applications (for example,

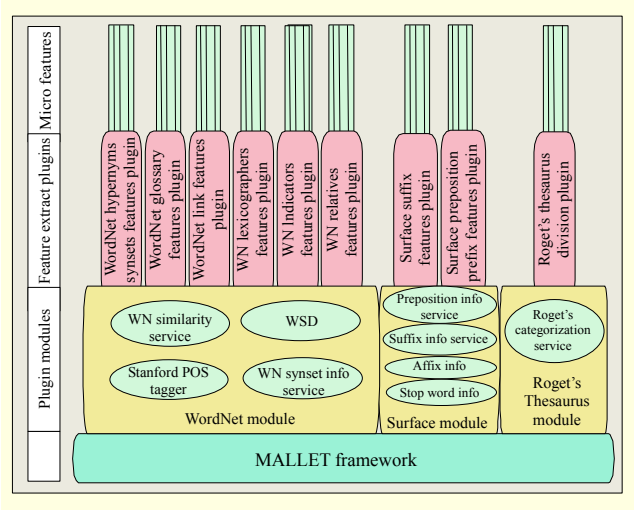

Fig. 6. Plugin-based binary feature extractor.

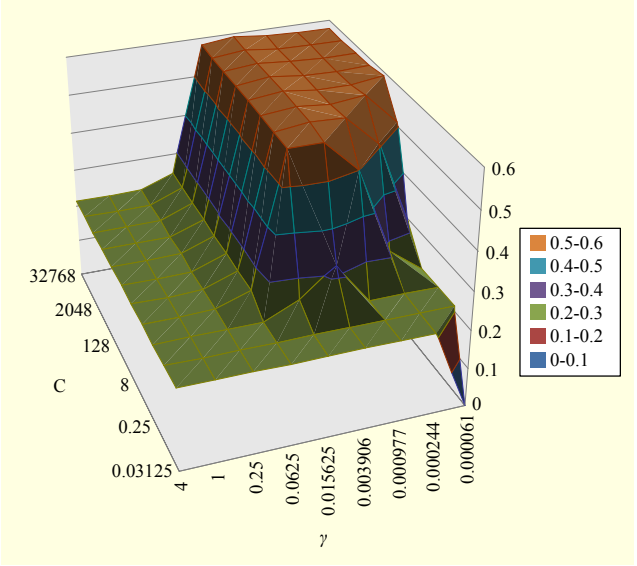

Fig. 7. Grid search for finding SVM parameters (C, γ).

different types of semantic or syntactic analysis or a different level of the required accuracy or efficiency).

The structure of this unit, which consists of three layers, is shown in Fig. 6. We use a plugin name for our feature extraction unit name to emphasize its importance for the configuration and efficiency of the overall system.

- The first layer is the plugin module layer, which encapsulates semantically- or syntactically-related plugins, while their common needs can be implemented as common services in their shared module. Sharing common services using modules is very important for the sake of efficiency because so many linguistic libraries are implemented as resource consuming services.
- The second layer, a feature extract plugin layer, gives the system a fine-grained mechanism for unit configuration (for example, the "WordNet hypernyms synonym feature plugin" which extracts all information about hypernyms and synonyms of NC constituents). This will give the system more cohesion and a lower rate of coupling.
- At the last layer, the micro feature layer, each binary feature is represented by a micro feature so that every feature plugin has a number of micro features. In addition, each micro feature encompasses a string as its pattern (for example, "x is the hypernym of n1") and an array list as its values. Then, the related plugin generates binary features by applying the relevant values to its correspondent string pattern.

V. Evaluation

For benchmarking and evaluating the efficiency of the system, it is essential to select good data sets of semantic relations. The system has been evaluated using the Tratz data set, which is the largest existing data set, with about 19,052 NC instances classified into 44 fine-grained semantic relations. Tratz and Hovy [4] also presented their classifiers and argued that they have the best reported results. We have used their results as a benchmark to assess the accuracy of our proposed approach. We have a data set of NC instances that have been classified by humans according to various fixed semantic relation classes. Each sample consists of an n1 and n2 as the constituents of an NC and two class numbers, one set by a human and one calculated by our approach.

All tests have been evaluated using 10-fold cross-validation with random folds. In this evaluation and also in previous sections, the accuracy is defined simply as

$$\text{Accuracy}=\frac{\text{Number of correctly classified instances}}{\text{Number of all instances}}. \quad (1)$$

The four different algorithms of classification, that is, Naïve Bayes [27], SVM [28], maximum entropy [29], and decision tree [30], are tested, and the best method is selected as the system classifier. Table 3 shows the comparison of classification accuracies achieved by exploiting different classification algorithms. The LIBSVM [31] has been used for the implementation of the SVM algorithm. The main challenge for using the SVM method is to find suitable SVM parameters. The SVM parameters of this system are selected by an approach presented in Hsu and others' work [32]. This approach employs RBF as the SVM kernel and utilizes multistep grid searching to find two parameters (C and γ). Figure 7 shows one step of such a grid search for finding the parameters for the SVM classifier for our data set. In this three-dimensional graph, we have different applicable values for the C parameter on one axis and different applicable values for the γ parameter on the other axis. The vertical axis shows

Table 3. Comparison of classification accuracies achieved by exploiting different classification algorithms.

Classification algorithm	Classification accuracy
Naïve Bayes	0.5651
SVM	0.7725
Maximum entropy	0.7910
Decision tree	0.4646

Table 4. Impacts of feature extraction modules/plugins on classification accuracy.

(%)

Module or Plugin	Accuracy of system without using module/plugin	Accuracy of system using only this module/plugin
Semantic module	70.19	74.85
Hypernym synonym feature plugin	76.62	73.32
WordNet glossary features plugin	77.17	70.35
WordNet link feature plugin	77.77	20.91
WordNet indicator features plugin	77.84	14.25
WordNet relation features plugin	77.84	21.57
Surface knowledge module	76.46	66.12
Thesaurus module	77.54	49.81

the accuracy values that are achieved by exploiting the SVM algorithm, which uses corresponding C and γ pairs from two other axes on the Tratz data set. The accuracy values are calculated using 10-fold cross-validation. In each step, the range of values for each axis has been narrowed to the range that has the best results on the previous step. The calculated parameters for our test data sets are C=119.43 and γ =0.000185.

For other classification methods, we use the implementation method used by the MALLET framework, and, for the maximum entropy method, the value of the Gaussian prior variance is set to 4.0. In this experiment, the implemented system, barring the feature selection unit, has been applied to the Tratz data set. The result is achieved using 10-fold cross-validation.

The achieved accuracy values presented in Table 3 show that using maximum entropy as the classifier has an obvious superiority over other methods. In addition, in comparison with its closest competitor, there is no need to determine a dependent data set parameter (such as, C and γ); therefore, a new manual reconfiguration of the system is not required in the case of changing the system input data set. Regarding the rest of the evaluation section, the results have been computed by using the maximum entropy method only.

Table 5. Percentage of features generated by each feature extraction module.

(%)

Feature module	Percentage of feature space used by feature module	All generated features
Semantic module	97.69	76.00
Surface knowledge module	2.22	8
Thesaurus module	0.09	26

Table 4 shows impacts of each module/plugin by two different experiments. The first column shows the module/plugin used in the experiment. For the first experiment, the second column shows accuracy values achieved by using all other modules except the module/plugin. For the second experiment, the third column shows the accuracy values achieved by using the module separately. The results of the test show that the "semantic module" has the highest impact and that removing it would result in an 8% drop in accuracy (that is, in relation to the full system accuracy of 79.8%). The table also shows good results even in standalone running, by a 4.5% margin, in comparison to the full feature results. A "hypernym synonym feature plugin" has the highest effect among all the plugins of this module.

While the results of running the system with different standalone plugins appear to be variable, the whole system is not sensitive to removing one of them. This could be due to the high redundancy in the information produced by different plugins.

Table 5 can be used as a means of comparing the amount of processing and memory resources consumed by each plugin. This table demonstrates the percentage of features generated by each feature extraction module. In addition, the percentage of coding space required for representing the module's features has been represented in the second column. The need for a higher feature space has a direct impact on the maximum entropy classification time, and a larger number of generated features has a direct influence on the amount of memory needed. A "semantic module" has the highest percentage of feature coding space, with a clear lead at 97.69% Removing this module would reduce accuracy from 79.7% to 70.2% (see Table 4), but the reduced system could still be used in an environment with high limitations on memory and processing power.

The overall disk volume needed by all feature extraction modules is about 110 MB, which is mainly used for storing

Table 6. Effect of feature selection algorithm on system accuracy and feature space size.

	Without feature selection	With feature selection
Accuracy	79.10%	79.70%
Feature space	208,860	128,971

data sources, such as WordNet, the POS tagger, stop words, and Roget's Thesaurus. The required disk volume is far lower than other comparable algorithms that use gigabytes or even hundreds of gigabytes of resources, such as the British National Corpus or the Web 1T 5-gram.

The effectiveness of the proposed feature selection algorithm is shown in Table 6. This table demonstrates the achieved accuracy and the size of feature space for a system running with a feature selection algorithm (second column) compared to a system running without a feature selection algorithm (third column). The results are achieved using the Tratz data set. These values show that a 0.6% improvement in accuracy can be achieved using only 61.75% of the feature space.

The system result of 79.7% accuracy passes the best previously reported result from Tratz and Hovy [4], which was 79.3%, and this is achieved while the proposed approach uses much smaller resources.

VI. Conclusion

Overall, the results presented in this study show that an NC semantic relation interpreter can achieve a level of accuracy higher than that of any state-of-the-art approach [4], without using a significant amount of linguistic resources. The system uses a large number of binary features from different resources, some of which are novel. Additionally, with the aid of co-training and collocation to generalize the feature knowledge and a novel feature selection algorithm to select more effective features, the desired accuracy and efficiency is achieved.

To test the effectiveness of the proposed approach, the implemented system is trained using the largest and most current available NC data set. Using different classification algorithms, maximum entropy classification shows the best results.

The other novelty of the system is the use of a WSD service to select the most appropriate sense for each constituent of an NC. However, the way the WSD is used is rather naïve and better disambiguation methods need to be utilized in the future.

The system is implemented using a modular- and plugin-based, layered framework; which is effective in providing ease of extension and reconfiguration for special cases.

VII. Future Works

In the future, we plan to implement the WSD service with some of the efficient disambiguation techniques and use various runtime dynamic configuration methods of feature extractors to cope with semantic relations interpretation in constrained and special cases. A promising research direction for future work is to extend the system to handle the prepositions in the NCs.

While a great deal of effort has been devoted to studying the semantic relations interpretation of NCs, little attention has been paid to the potential applications of sematic relations interpretation in specific areas. For example, one of the significant benefits of semantic interpretation is to find complex semantic alignments in different ontologies, and we are currently carrying out work in this area.

References

[1] S.N. Kim and T. Baldwin, "Benchmarking Noun Compound Interpretation," *Joint Conf. Natural Language Process. (IJCNLP)*, Hyderabad, India, 2008.

[2] D. Ó Séaghdha and A. Copestake, "Semantic Classification with Distributional Kernels," *COLING*, 2008.

[3] D. Ó Séaghdha, *Learning Compound Noun Semantics*, doctoral dissertation, Computer Laboratory, University of Cambridge, Cambridge, England, UK, 2008.

[4] S. Tratz and E. Hovy, "A Taxonomy, Dataset, and Classifier for Automatic Noun Compound Interpretation," *Proc. 48th Annual Meeting Assoc. Computational Linguistics (ACL-10)*, Uppsala, Sweden, 2010, pp. 678-687.

[5] D. Ó Séaghdha and A. Copestake, "Co-occurrence Contexts for Noun Compound Interpretation," *Proc. ACL-07 Workshop Broader Perspective Multiword Expressions*, Prague, Czech Republic, 2007.

[6] S.N. Kim and T. Baldwin, "An Unsupervised Approach to Interpreting Noun Compounds," *Proc. IEEE Int. Conf. Natural Language Process. Knowl. Eng. (NLP-KE-08)*, Beijing, China, 2008, pp. 1-7.

[7] D. Ó Séaghdha, "Semantic Classification with WordNet Kernels," *Proc. HLT-NAACL*, Boulder, Colorado, USA, 2009, pp. 237-240.

[8] B. Rosario and M. Hearst, "Classifying the Semantic Relations in Noun Compounds via a Domain-Specific Lexical Hierarchy," *Proc. Conf. Empirical Methods Natural Language Process.*, Somerset, NJ, USA, 2001, pp. 82-90.

[9] D. Moldovan et al., "Models for the Semantic Classification of Noun Phrases," *Proc. HLT-NAACL Workshop Computational Lexical Semantics*, Boston, MA, USA, 2004, pp. 60-67.

[10] S.N. Kim and T. Baldwin, "Automatic Interpretation of Noun

Compounds Using WordNet Similarity," *Proc. 2nd Int. Joint Conf. Natural Language Process.*., 2005, pp. 945-956.
[11] C. Fellbaum, *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press, 1998.
[12] V. Nastase et al., "Learning Noun-Modifier Semantic Relations with Corpus-Based and WordNet-Based Features," *Proc. AAAI*, 2006, pp. 781-786.
[13] R. Girju, "Improving the Interpretation of Noun Phrases with Cross-Linguistic Information," *Proc. ACL*, 2007, pp. 568-575.
[14] S.N. Kim and T. Baldwin, "Interpreting Noun Compound Using Bootstrapping and Sense Collocation," *Proc. Pacific Association Computational Linguistics (PACLING)*, 2007, pp. 129-136.
[15] J.-X. Huang et al., "Feature-Based Relation Classification Using Quantified Relatedness Information," *ETRI J.,* vol. 32, no. 3, June 2010, pp. 482-485.
[16] D. Ó Séaghdha and A.A. Copestake, "Using Lexical and Relational Similarity to Classify Semantic Relations," *Proc. EACL*, 2009, pp. 621-629.
[17] L. Burnard, Ed., *Users Reference Guide for the British National Corpus*, Oxford: Oxford University Press [for *BNC Consortium*], May 1995.
[18] T. Brants and A. Franz, "Web 1T 5-Gram Version 1," *Philadelphia Linguistic Data Consortium*, Philadelphia, PA, USA, 2006.
[19] K. Betty, *Roget's Thesaurus of English Words and Phrases*, UK: Penguin, 1998.
[20] S.N. Kim and T. Baldwin, "Disambiguating Noun Compounds," *Proc. 22nd Conf. Artificial Intell.*, 2007, pp. 901-906.
[21] The Stanford Natural Language Processing Group, "Stanford Log-Linear Part-of-Speech Tagger," accessed 20 Sept. 2010. Available: http://nlp.stanford.edu/software/tagger.shtml
[22] G. Pirró and N. Seco, "Design, Implementation and Evaluation of a New Semantic Similarity Metric Combining Features and Intrinsic Information Content," *On the Move to Meaningful Internet Systems: OTM 2008*, R. Meersman and Z. Tari, Eds., Berlin/Heidelberg: Springer, vol. 5332, 2008, pp. 1271-1288.
[23] M. Jarmasz, *Roget's Thesaurus as a Lexical Resource for Natural Language Processing*, master's thesis, University of Ottawa, Ottawa, Ontario, Canada, 2003.
[24] G. Forman et al., "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *J. Machine Learning Research,* vol. 3, 2003, pp. 1289-1305.
[25] A.K. McCallum, "MALLET: A Machine Learning for Language Toolkit," 2002.
[26] M. Hall et al., "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.,* vol. 11, 2009, pp. 10-18.
[27] I. Rish, "An Empirical Study of the Naive Bayes Classifier," *IJCAI-01 Workshop on "Empirical Method in AI,"* 2001, pp. 41-46.
[28] D. Fradkin and I. Muchnik, "Support Vector Machines for Classification," *Discrete Methods Epidemiology,* vol. 70, 2006, pp. 13-20.
[29] A.L. Berger et al., "A Maximum Entropy Approach to Natural Language Processing," *Computational Linguistics,* vol. 22, 1996, pp. 39-71.
[30] L. Liu and M. Ozsu, Eds., *Encyclopedia of Database Systems*, Berlin/Heidelberg: Springer-Verlag, 2009.
[31] C. Chang and C. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, May 2011.
[32] C. Hsu et al., "A Practical Guide to Support Vector Classification," Department of Computer Science, National Taiwan University, 2003. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Ahmad Zaeri received his BSc and MSc in computer software engineering from Shahid Beheshty University, Tehran, Iran, in 1998 and the University of Isfahan, Isfahan, Iran, in 2001, respectively. Currently, he is a PhD candidate at the University of Isfahan. He was a visiting researcher at the Knowledge Representation and Knowledge Management Research Group of Mannheim University, Mannheim, Germany, in 2011. His main research interests include NLP, semantic web, description logic, and multi-agent systems.

Mohammad Ali Nematbakhsh is an associate professor of computer engineering in the School of Engineering at the University of Isfahan, Isfahan, Iran. He received his BSc in electrical engineering from Louisiana Tech University, Ruston, LA, USA, in 1981 and his MSc and PhD in electrical and computer engineering from the University of Arizona, Tucson, AZ, USA, in 1983 and 1987, respectively. He worked for the Micro Advanced Co. for three years in the USA and for the Toshiba Corporation for six years, split between the USA and Japan, before joining the University of Isfahan. He has written more than 80 published research papers and a database book that is widely used in universities, and he has registered three US patents. His main research interests include semantic web, database systems, and computer network software.