Interantional Jounal of Soft Computing 2 (1): 96-102, 2007

A Generic Discriminative Model for Information Extraction

[1]M. Sathya, [1]V. Prasanna Venkatesan and [2]G. Sureshkumar
[1]Department of Computer Science, Ramanujam School of Mathematics and
Computer Sciences, Pondicherry University, Pondicherry, India
[2]Department of Information Technology,
Sri Manakula Vinayagar Engineering College, Pondicherry, India

Abstract: Information extraction is the automatically extracting of facts from text, which includes detection of named entities, entity relations and events. Conventional approaches to information extraction try to find syntactic patterns based on deep processing of text, such as partial or full parsing. The problem these solutions have to face is that as deeper analysis is used, the accuracy of the result decreases and one cannot recover from the induced errors. On the other hand, lower level processing is more accurate and it can also provide useful information. However, within the framework of conventional approaches, this kind of information cannot be efficiently incorporated. This study describes a novel supervised approach based on kernel methods to address these issues. In this approach customized kernels are used to match syntactic structures produced from different preprocessing phases. Using properties of a kernel, individual kernels are combined into a composite kernel to integrate and extend all the information. The composite kernels can be used with various classifiers, such as Nearest Neighbor or Support Vector Machines (SVM). Each level of syntactic information can contribute to Information Extraction (IE) tasks and low-level information can help to recover from errors in deep processing.

Key words: Discriminative model, extraction, syntactic patterns

INTRODUCTION

The development of information technology and digital libraries is changing the means of scholarly communication and research. With the increased abundance of digital material, making use of the digital collections becomes one vital issue in digital libraries. Future digital libraries should not only support information organization and access, but should also assist knowledge discovery from digital collections[1,2].

Information Extraction is useful in situations where a set of text documents exist containing information, which could be more easily used by a human or computer if the information were available in a uniform database format. Thus, an information extraction system is given the set of documents and a template of slots to be filled with information from the document. Information extraction systems locate and in some way identify the specific pieces of data needed from each document. Two different types of data may be extracted from a document: more commonly, the system is to identify a string taken directly from the document, but in some cases the system selects one from a set of values which are possible fillers for a slot. The latter type of slot-filer may be items like dates, which are most useful in a consistent format, or they may simply be a set of terms to provide consistent values for information, which is present in the document, but not necessarily in a consistently useful way. To address the problems existing in prior IE approaches, a new discriminative model based on the kernel method is proposed here. It incorporates different levels of syntactic information to find useful clues for an IE task. The idea is that an IE model should not commit itself to only deep analysis. Shallow information, such as word collocations, may also give important clues.

THE KERNEL BASED CLASSIFICATION MODEL

Kernels can be seen as representations of objects using large number of features. In this model, kernels are designed to represent each level of processing result and combine them. Many classifiers can be used with kernels. The most popular ones are SVM (Support Vector Machines), KNN (k-Nearest-Neighbors) and Voted

Corresponding Author: M. Sathya, Department of Computer Science, Ramanujam School of Mathematics and Computer Sciences, Pondicherry University, Pondicherry, India

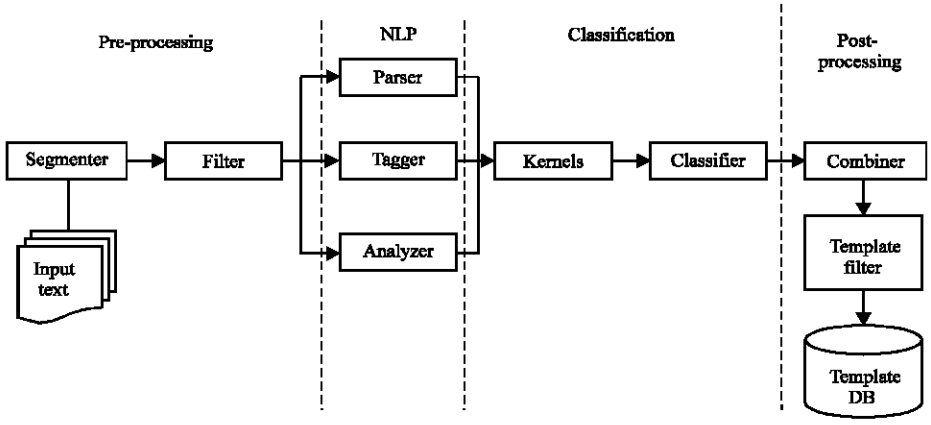

Fig. 1: Structure of generic discriminative IE model

Perceptrons. This generic framework makes no assumption about the text structure of events. Instead, kernels are used to represent syntactic information from various syntactic sources and active learning is used for classifying different features. The structure of this model is shown in Fig. 1.

The generic IE model consists four major task modules namely:

- Pre-processing
- Natural Language Processing
- Classification
- Post-processing

In this model, text is first extracted from input documents. Then it is preprocessed at different levels. The preprocessing modules mainly consist of segmenting and filtering process. The NLP modules include a part-of-speech tagger, name tagger, sentence parser and dependency analyzer, but are not limited to these[3]. Other general or custom tools can also be included. Different kernels are used to encode the corresponding syntactic processing result. Individual kernels can be combined into composite kernels as input to a classifier. In this model the classifier used is linear classifier. In the training phase, targets are labeled in the text as examples to train the classifier. In the test phase, unlabeled text is processed in the same way as training data. Candidate examples are generated from the text based on the preprocessing results. Then the classifier can make predictions on candidates to identify the targets. The kernel based on the dependency analysis has ability to capture more regularization in text. Most IE tasks are multi-class classification. After each classifier makes predictions, post-processing can be applied to resolve conflicts or to improve the predictions using other heuristics. The final result is outputted in template DataBase. With kernel functions things can be much easier. A kernel function is used to match two structured objects and produce a similarity value. As long as the kernel is mathematically valid, it can be plugged into any learning algorithm, which has dual form representations.

Preprocessing: The text layout analysis works on each complex text segment by segmenting it into a section-paragraph-sentence hierarchical structure depending on the input text document layout heuristics. This hierarchical structure is useful to the Information Extraction system to enable it to more intelligently determine the relevant segment of the entire text as the desired extracted information in the Template Filling module, i.e., whether the entire section, or specific paragraphs or sentences are to be extracted. After the input text is segmented into logical text segments by the Text Pre-processing module, relevant text segments are sent to the Surface Text Analysis module for text analysis. There are two phases to Surface Text Analysis-Morphological Analysis and Word Pattern Analysis[4]. Morphological Analysis decomposes the input text segment into its basic word constituents and normalize these constituents to their base-forms. In addition, it assigns potential part of speech as well as other linguistic information to the normalized words.

Segmenter: The text layout analysis module segments a text into the following categories:

- Basic text segments, such as address, are extracted immediately without further analysis.
- Simple text segments, such as organization names, will be subjected to simple text analysis.

- Complex text segments, such as instructions and descriptions, will be subjected to Text Layout Analysis module.
- Redundant or irrelevant text segments that will not contribute to subsequent processing are identified and filtered out of the system. E.g. tabular data in appendices. This is handled by the text segment filtering module.

The text layout analysis works on each complex text segment by segmenting it into a section-paragraph-sentence hierarchical structure depending on the input text document layout heuristics. This hierarchical structure is useful to the GIE Information Extraction system to enable it to more intelligently determine the relevant segment of the entire text as the desired extracted information in the template filling module, i.e., whether the entire section, or specific paragraphs or sentences are to be extracted.

Filter: The filtering module is used to eliminate the unnecessary words and symbols, which are not used in further processing. The words used as connectives, stop-word and symbols like comma are filtered out in 2 stages. The needless pages should filtered out and process only relevant pages.

Natural language processing: The inputs to Natural Language Processing (NLP) applications are structures like word sequences, trees or graphs. Most NLP applications involve finding syntactic patterns from these structures. A traditional way to tackle this problem often involves human knowledge to identify smaller pieces as features and then applies a learning algorithm on the features[5,6]. Obviously, this could be insufficient due to the limitation of human observation over large amounts of data.

Parsing: As a textual representation of a information is available, a token parsing is performed in which single words, delimiters, full stops, special characters etc. are identified in a text. Such identification is naturally easier from (more formal) written text, as it is to identify sentence structures from spoken text. Parsing forms an important basis for the processes taking place at the higher levels of abstraction, such as (proper) noun identification (and possibly sub-types thereof), compound analysis and sentence fragment determination. During parsing, typically a database is generated containing the token types, offset/position in the text and additional information about the tokens for later use. The role of the lexical level is to determine the word categories that words belong to, as well as identification of specific word types that do not necessarily follow from the application of a grammar (such as proper nouns). Generally two word groups can be identified on the lexical level, grammatical words and lexical words (or function words and content words, respectively[7]:

- Grammatical words:
- Articles (the; a; and),
- Adjectives (green; white; cold),
- Predeterminers (half; both),
- Prepositions (under-; over-; un-) and
- Pronouns (it; its; he; she)
- Lexical words:
- Adverbs (shortly),
- Qualitative adjectives (larger;smaller),
- Nouns (project; people),
- Verbs (co-operate; communicate)

The word groups that bear the real semantics of a text are identified as being the adverbs, adjectives, nouns and verbs[7]. Many approaches that deal with the analysis of documents and corpora of texts will concentrate on one or more of these four groups. Depending on how elaborate an approach is with respect to the understanding, it will typically include the analysis of nouns, adjectives, verbs and adverbs respectively. The grammatical words represent a group of words that is also referred to as closed class words and they are easily captured in dictionaries for lookup. The second category of open class words poses a problem for table lookup or rule based identification. On the one hand the number of lexical words is not stable, i.e., new words belonging to this class are added to the language regularly. On the other hand, besides of the regular introduction of new words, there is the problem of the several tenses of verbs and the modularity of words (plural, singular), which often makes them hard to identify[8].

Often approaches solve these problems by using direct table lookups for the grammatical words and using a more or less knowledge intensive approach for the lexical words. For the latter, rule bases can be used (with as main disadvantage their maintainability), or statistical heuristics can be applied for calculation of the most probable word types given certain sentence fragment and corpora of texts. Often used in this respect POS -Taggers (Part Of Speech Taggers). Results of such taggers are frequently published at MUC conferences. Statistical POST calculates conditional probabilities (using Bayes theorem) and uses actual appearances of word sequences to calculate the most probable word type for a particular word. Whereas in theory such a probability may depends

on all preceding words in a text, in practice POST (and other systems based on probabilistic theory) take a pragmatic approach and calculate the conditional probabilities according to the n preceding words. This is then referred to as n-Gram Models.

On top of this, often a separate heuristics is implemented to identify proper nouns. Once identified proper nouns can be sub-divided into categories like person, organization, location (named entity recognition). Such rules build upon the results of the token parsing process. Basic rules for identification of proper nouns are often defined in the following way:

- If all_capitalised then proper noun,
- If not the first word in a sentence and first letter is a capital then proper noun (check lists with names and locations for possible proper noun types.),
- If string contains and Co. or ltd. then tag previous word as proper noun (type: company).

Words that are thus annotated according to the processes described above are used in typical information extraction tasks. Often a special focus is on proper nouns in combination with the categories of words mentioned above. The knowledge extracted here can now be forwarded to the next level of analysis.

Tagging: Tagging is a process of selecting the most likely sequence of syntactic categories for the words in a sentence. For example, in the sentence Mary picked up the rose. Preposition to up, article to the and noun to rose. Especially, for the word with more than one syntactic categories, such as rose which could also be a verb as in the sentence Mary rose from her chair, part of speech tagging needs to assign a unique syntactic category. In this case, it assigns a noun rose in the first sentence. The advantages of incorporating a part of speech tagging in an information extraction system include the following two aspects: 1) ambiguity at subsequent stages of processing can e reduced; 2) could avoid many errors due to incorrect categorization of rare senses. Because of these advantages, some systems in applied part of speech tagging.

Part-of-speech tagger: A Part Of Speech (POS) tagger provides basic syntactic information by taking sentences as input and labeling each word or symbol in the sentence with a part-of-speech tag (e.g., noun, verb, adjective, preposition). This doesn't provide as much information as a parser, since it doesn't identify phrases or the relationships between parts of the sentence. However, taggers are typically faster and more robust than full parsers, particularly in the face of ungrammatical text such as would be commonly found in newsgroup postings and email messages and to a lesser extent in newswire articles. If a large amount of hand-tagged text happens to exist for a given domain, the tagger can be trained from scratch. In the absence of such text, several methods exist for tuning the tagger for a specific domain. The lexicon is used by the tagger to determine what parts of speech a word can be and what its most common part of speech is, can be modified to include new words from the new domain, or to reach the actual distribution of parts of speech in the new domain. Adding the most frequent novel words to the lexicon seems to be the most effective way of improving tagging quality. Finally, the rules used by the tagger are quite comprehensible and new rules can be added by hand to improve the tagging quality. This is a much more difficult and time-consuming process, since it requires a good understanding of what the tagger is doing wrong and how to do it.

Analyzer: named entity recognition: The simplest and most reliable IE technology is Named Entity recognition (NE) or Named Entity Extraction. NE systems identify all the names of people, places, organizations, dates, amounts of money, etc. This process is weakly domain dependent, i.e., changing the subject matter of the texts being processed from financial news to other types of news would involve some changes to the system and changing from news to scientific papers would involve quite large changes. Real text is rich in proper names, expressions for dates, values, etc. Those phrasal units do not pose any problem to human readers, but they do cause some ambiguities when processed by the computer. For example, in Mitsubishi announced today…., is Mitsubishi referring to a person name or a company name? For the information extraction task which aims for who did what, where, when, successfully identifying name entities is particularly important. The approaches to the Name Entity identification adopted in many systems range from the more or less purely statistical such as the use of HMM. In some cases, the target-information is not as straight forward as identifying well-defined word sequences. Hence, the analyzed node-list of words and entities from the surface text analysis has to be subjected to further deep level of analysis in the Deep Text Analysis module to discover more complex concepts and the relationships between the concepts.

There are two phases to deep text analysis-structural analysis followed by syntax and semantic normalization. The structural analysis matches the word and pattern node-list from the surface text analysis with

appropriate grammar rules, assigns meaning to them and verifies sentence processing to produce the text meaning in the form Message Intermediate Representation (MIR). The Syntax and Semantic Normalization normalizes variant forms of the MIR with the same text meaning for ease of extraction during the Template Filling process.

For example, the following sentences:

- All documents are to be DHLed to the personnel department.
- Pls send us the original transcripts through surface mail.
- Dispatch to us by express mail the certificates to the above effect.
- Fax us the letters as soon as possible.
- All disclaimers are to be signed and forwarded to us by courier.

State the same request of wanting someone to send some kind of documents to a person or an entity through some means. The purpose of syntax and semantic normalization is to transform all the variant but semantically equivalent surface structures into the same MIR structure. After the text has been pre-processed and analyzed, it is now ready for extraction by the Template Filling module. This module uses the Frame for Extracting Information from Messages (FEIM). Knowledge on what constitutes relevant target information from the input text is stored as a set of FEIM specifications, which can be viewed as a template of slots; each slot corresponds to a piece of relevant information and contains the descriptions of the information to be captured. It also contains information on how to search for the required information.

Classifier: The linear classifiers are separating data with largest margin. This property gives it good generalization ability in high-dimensional spaces, making it a good classifier for this approach where using all the levels of linguistic clues could result in a huge number of features. Given all the levels of features incorporated in kernels and training data with target examples labeled, the classifier can pick up the features that best separate the targets from other examples, no matter which level these features are from. In cases where an error occurs in one processing result (especially deep processing) and the features related to it become noisy, the classifier might pick up clues from other sources, which are not so noisy. This forms the basic idea of this approach. Therefore under this scheme errors introduced by one processing level can be overcome; more particularly, accurate low level information is expected to help with less accurate deep level information.

The classification technique relies on a document classifier to create the values for filling the template; the first step is to train such a classifier. A supervised learning is used to construct the classifier from a set of pre-classified documents. The procedure follows a sequence of steps, described below[6,9]. Eliminate the set all words that appear very frequently in the training documents, as well as very infrequently appearing words. This initial feature selection step provides a functional form for the distribution of word frequencies in document collections. Very frequent words are usually auxiliary words that bear no information content (e.g., am and so in English). Infrequently occurring words are not very helpful for classification either, because they appear in so few documents that there are no significant accuracy gains from including such terms in a classifier.

However, frequency information alone is not, after some point, a good indicator to drive the feature selection process further. Thus the information theoretic feature selection algorithm is used to eliminate the terms that have the least impact on the class distribution of documents. This step eliminates the features that either do not have enough discriminating power (i.e., words that are not strongly associated with one specific category) or features that are redundant given the presence of another feature. Using this algorithm the number of features can be decreased in a principled way and a much smaller subset of words can be used to create the classifier, with minimal loss in accuracy. In addition, the remaining features are generally more useful for classification purposes, so classifiers constructed from these features will tend to include more meaningful terms.

After selecting the features (i.e., words), an existing machine learning algorithm is used to create a document classifier. Many different algorithms for creating document classifiers have been developed over the last few decades. Well-known techniques include the Naive Bayes classifier, RIPPER and Support Vector Machines, to name just a few. These document classifiers work with a flat set of categories. Once the document classifier is trained, that can be used to classify all the documents in a database of interest to determine the number of documents about each category in the database. A binary classifier decides whether a document, represented using m features (i.e., words), belongs to one class or not. A binary linear classifier makes this decision by calculating, during the training phase, m weights w1, …,wm and a threshold b determining a hyperplane such that all points t = <t1, : : :, tm> in the hyperplane satisfy the equation:

$$\sum_{i=1}^{m} w_i t_i = b \qquad (1)$$

This hyperplane divides the m-dimensional document space into two regions: the region with the documents that belong to the class in question and the region with all other documents. Then, given the m-dimensional representation <s1,..., sm> of a document, the classifier calculates the document's score as,

$$\sum_{i=1}^{m} w_i s_i \quad (2)$$

The value of this score relative to that of threshold b determines the classification decision for the document.

A large number of classifiers fall into the category of linear classifiers. Examples include Naive Bayes and Support Vector Machines (SVM) with linear kernel functions. A classifier for n classes can be created using n binary classifiers, one for each class. Note that such a composite classifier may result in a document being categorized into multiple classes or into no classes at all. The Eq. 1 is used to approximate a linear classifier[10].

Types of kernels: To make use of syntactic information from different levels, kernel functions or syntactic kernels can be developed to represent a certain level of syntactic structure.

The possible syntactic kernels include

Sequence kernels: Representing sequence level information, such as bag-of words, n-grams or a string kernel.

Phrase kernels: Representing information at an intermediate level, such as kernels based on multiword expressions, chunks or shallow parse trees.

Parsing kernels: Representing detailed syntactic structure of a sentence, such as kernels based on parse trees or dependency graphs.

These kernels can be used alone or combined with each other using the properties of kernels. They can also be combined with general kernels like polynomial or RBF (Radial Basis Function) kernels to generate a high-order, non-linear decision surface. This can be done either on individual kernels or on the composite kernel. In practice each kernel can be tested for the task as the sole input to a classifier, to determine if this level of information is helpful or not. After figuring out all the useful kernels, try to combine them to make a composite kernel as final input to the classifier. The way to combine them and the parameters in combination can be determined using validation data. Since the preprocessing modules are standard analyzers, once a kernel is developed for a certain level of information, it could be reused when the underlying domain changes. Many information extraction tasks can be implemented by this model, such as named entity recognition, entity relation detection and slot filler detection for events.

Event occurrence detection kernels: In information extraction, one interesting issue is event occurrence detection, which is determining whether a sentence contains an event occurrence or not. If this information is given, it would be much easier to find the relevant entities for an event from the current sentence or surrounding sentences. Traditional approaches do matching (for slot filling) on all sentences, even though most of them do not contain any event at all. Event occurrence detection is similar to sentence level information retrieval, so simple models like bag-of-words or n-grams could work well.

Post processing: Post processing consists of combiner and template filler. Keywords annotated by the classifier are arranged based on the segments sequence and processed by the template filler for choosing appropriate keyword to fill template slots.

Combiner: The keywords extracted for individual segments are combined together and collectively used by the template filler for generating meta-data database.

Template filler: There can be more than one template but at any one time, there is only one master template, which drives the extraction process. The master template will essentially contain information on the sequence of filling up the subordinate templates. The subordinate templates will in turn contain the target-information knowledge. If there is only one template, then the template itself will contain the target-information knowledge. For multiple templates, the template filling process will iterate through each template, in the order accorded by the master template. The template filling process will go through each FEIM slot within a template and attempt fill up the value for the content attribute of the slot. In the process, it will trigger the filling mechanism to execute the functions or macros embedded in the Scenario Templates (ST's) are the prototypical outputs of IE systems, being the original task for which the term was coined.

RESULTS

We developed a Generic Information Extraction System (GIES) for testing the proposed generic IE model. The computer-related job-posting corpus used to test the

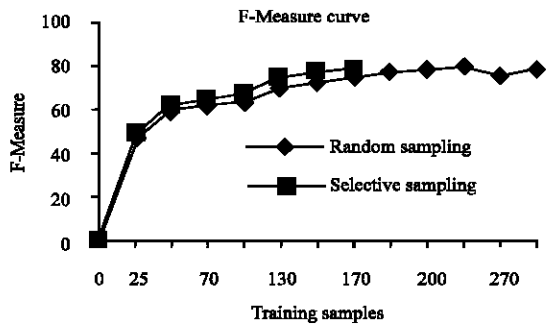

Fig. 2: The F-Measure curve of active learning

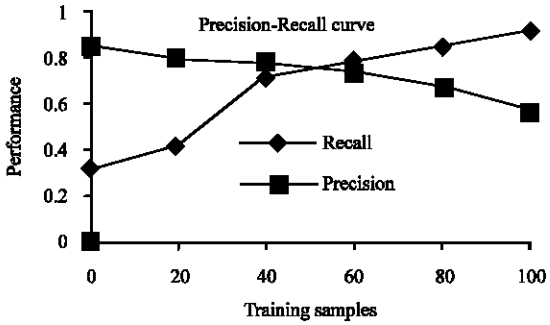

Fig. 3: The precision-recall curve of active learning

generic model. The training set consists of 300 postings to the local newsgroup. Training and test sets were generated using 10-fold cross-validation and learning curves generated by training on randomly or actively selected subsets of the training data for each trial. For active learning, there were n = 10 bootstrap examples and subsequent examples were selected one at a time from the remaining 260 examples.

In information extraction, the standard measurements of performance are precision (the percentage of items that the system extracted which should have been extracted) and recall (the percentage of items that the system should have extracted which it did ex- tract). In order to combine these measurements to simplify comparisons, it is common to use F-measure: F = (2 *precision* recall)/(precision+recall). It is possible to weight the F-measure to prefer recall or precision, but we weight them equally. For the active learning results, we measured performance at 10-example intervals. The results for random sampling are measured less frequently.

Increasing the size of training corpus did not dramatically improve the performance in terms of F measure. Figure 2 shows the results, where GIES uses random sampling and selective sampling. The difference between the curves is not large, but does represent a large reduction in the number of examples required to achieve a given level of performance. At 150 examples, the average F-measure is 74.56, exactly the same as the average F-measure with 270 random examples. This represents a savings of 120 examples, or 44%. The differences in performance at 120 and 150 examples are negligible.

The Fig. 3 shows the Precision-recall performance of active leaning process. In job-posting, the most documents contain the header section with all the target fields easily identifiable right after the corresponding key word. We have created a derivative dataset in which documents are stripped of headers and two extra fields are sought: date and topic. Indeed this corpus turned out to be more difficult, with our current set of features we obtain only 64% performance on post and 68% performance on specialization.

REFERENCES

1. Chen, H., 2003. Towards building digital library as an institution of knowledge. In NSF Post Digital Library Futures Workshop.
2. Witten, H., K. J. Don, M. Dewsnip and V. Tablan, 2004. Text mining in a digital library. Intl. J. Digital Libraries, 4: 56-59.
3. Declerck, T., J. Klein and G. Neumann, 1998. Evaluation of the NLP components of an information extraction system for German. In Proceedings of the first Intl. Conference on Language Resources and Evaluation (LREC), Granada, pp: 293-297.
4. Fernandez, E.B. and X. Yuan, 1999. An analysis pattern for reservation and use of reusable entities. Pattern Languages of Programs Conference, (PLoP99). http://stwww.cs.uiuc.edu/~plop/plop99.
5. Califf, M. E. (Ed.), 1999. Papers from the AAAI-1999 Workshop on Machine Learning for Information Extraction, Orlando, FL. AAAI Press.
6. Califf, M.E. and R.J. Mooney, 1999. Relational learning of pattern-match rules for information extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, FL., pp: 328-334.
7. Winograd, T., 1972. Understanding Natural Language. Cognitive Psychology No.3, Academic Press, New York.
8. Fayad, M.E. and M. Nabavi, 2002. Stable Model-Based Software Reuse. ECOOP 2002, Workshop on Model-based Software Reuse, Malaga, Spain.
9. Cardie, C., 1997. Empirical methods in information extraction. AI Magazine, 18: 65-79.
10. Chang, C.C. and C.J. Lin, 2001. Training nu-Support Vector Classifiers: Theory and Algorithms. Neural Computation.