Text Categorization Based on Domain Ontology

Qinming He[1], Ling Qiu[2*], Guotao Zhao[1], and Shenkang Wang[1]

[1] College of Computer Science
Zhejiang University, Hangzhou, China 310027
hqm@cs.zju.edu.cn
[2] School of Information Systems
Singapore Management University, Singapore 259756

Abstract. Methods based on machine learning have been proposed with certain advantages for TC (*text categorization*). However, it is still difficult to further increase the precision and understandability of categorization due to certain aspects of text itself. In this paper, we propose an architecture for TC by addressing domain ontology. Not only more effect and understandability of categorization are achieved, simulation results show a great reducing of keyword numbers and saving of system costs.

1 Introduction

Currently *text categorization* (or TC for short, also known as *text classification*) is being widely applied in many contexts covering document indexing, document filtering, word sense disambiguation, etc. However, the study of TC can be dated back to 1960s. Before the early 1990s, *knowledge engineering* (or KE for short) was the main tool deployed by the most popular approaches to TC. But from the early 1990s, with the drastically increased number of electronic documents, there was an urgent demand for high quality of TC with various classification criteria. Approaches originated from KE have increasingly lost popularity due to some technical limitations, whereas methods based on *machine learning* (or ML for short) come onto the stage at this time. ML approaches are advantageous over KE methods in terms of high degree of automation, stability of performance, flexibility, accuracy comparable to that achieved by human experts, and considerable savings of expert labor power [5].

Normally an ML approach obtains the classifiers by analyzing training documents, but there still exists several difficulties for TC. Firstly, there lacks semantical support. For example, if the classification is done word by word as the basic object, it is impossible to match the concepts the words represent, especially in the cases where a concept is represented by a phrase in the context. Secondly, there exists multi-presentation of information. Usually the same concept in a document appears in different presentations, but unluckily they are

* The author acknowledges the School of Mathematics & Computer Science at Fujian Normal University, Fuzhou, China, for the support and convenience provided for teaching, research and accommodation during his visiting, and is also thankful to the group members of Zhejiang University for the productive and cheerful collaboration.

treated as different words. Thirdly, there is a paucity of related information. Incomplete information about a concept in a document will cause incomplete results of machine learning.

Due to the above reasons, in this paper, we introduce domain ontology to the TC system, aiming to overcome the above difficulties and to improve the precision of classification. We construct an architecture of the TC system, i.e., *Text Categorization Agency* (or TCA for short). We deploy COSA algorithm [3] in our architecture for the extraction of valid concepts, greatly reducing the number of key words to be searched and thus saving a lot of system costs. This is an attempt by introducing ontology to TC, however, our simulation results show that our method is competitive to others (e.g., Naive Bayes, RIPPER [1]) in terms of precision and recall of categorization.

The remaining part of the paper is organized as follows. In Section 2, we present the architecture of TCA and describe in detail how it performs its functionality. Following that in Section 3, the testing results are given, which aim to show the feasibility and effectiveness of TCA. Finally in Section 4, we conclude the paper and point out research directions for future study.

2 Architecture of TCA

Recently, ontology is intensively applied by research community of computer science and engineering for the presentation of domain knowledge [6]. By introducing ontology to TC, we propose an architecture for our TC system, namely *Text Categorization Agency* (or TCA for short henceforth). Based on the structural information extraction from structure ontology, TCA firstly extracts structural information from texts and afterwards processes them uniformly. This is helpful to eliminate the heterogeneity of texts. The domain ontology addressed in the system is used for categorization on a basis of semantics. The TCA also adopts the COSA algorithm [3] to sieve up those concepts with extremely low or extremely high supports, and thus saves a lot of system costs by greatly reducing the number of key words to be searched. Figure 1 gives the skeleton of the architecture, which contains three layers, namely ontology definition layer, structurization layer, and categorization layer. In what follows, we will describe the core components of each layer and how they perform their functions.

2.1 Ontology Layer

This layer is comprised of structure ontology, domain ontology, and domain vocabulary. The structure ontology defines the structures of various heterogeneous texts, and provides the methods for the uniformly processing of heterogeneous texts (see [7] for details). We adhere to the presentation of ontology as a *six*-tuple [3]. The structure ontology is organized as a tree [7]. By applying the *Left Filtering Maximization* algorithm [2], the matching of expressions extraction is unique. With the support of domain vocabulary, the domain ontology provides related domain knowledge, including concepts, associations, entities, attributes, etc. It is also the basis for semantic extraction and sieving of key words.

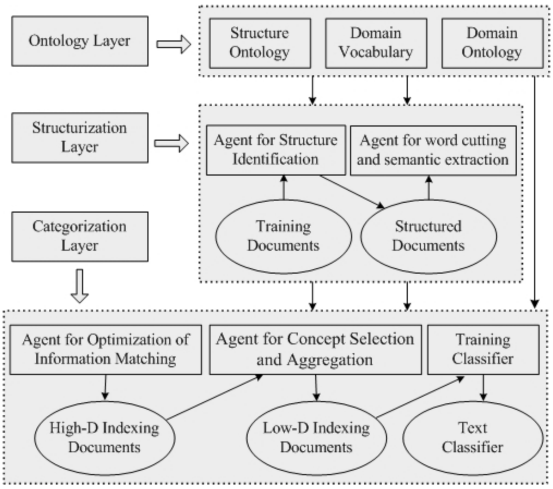

Fig. 1. Architecture of TCA

2.2 Structurization Layer

This layer is the portal of input texts waiting for categorization. Input texts are pre-processed (structurized and then vectorized) in this layer. With two agents as its main components, this layer functions as follows. Firstly, assisted by the ontology layer, training documents (possibly heterogeneous from each other) are processed by the agent for structure identification. Thus we get well formatted (structured) documents as outputs. Then these documents are passed to the agent for word cutting and semantic extraction. Secondly, the agent for word cutting and semantic extraction cuts the structured documents into meaningful words (or phrases) defined by the domain vocabulary, and extracts the semantics based on the domain ontology. After this process, all documents can be represented by a vector space with a clear definition of semantics. Here the vector space is a set of vectors each of which represents the aspects (and the corresponding weight of each) of certain type of text. The value of each dimension of the vector can be an integer, a string, or an array of attributes [1].

The first stage work is finished here, and the results are forwarded to categorization layer for further optimization and aggregation, and finally the training classifiers are obtained. With the classifiers, an unknown document (after preprocessing) can be sent to the categorization layer for classification.

2.3 Categorization Layer

This layer performs two main functions. Firstly, the generation of classifiers. With the output from structurization layer, we first optimize the information matching, then convert the high dimensional indexing documents to low dimensional indexing documents with the assistance of certain mechanism, and lastly we obtain the training classifiers. Secondly, categorization for unknown documents. The training classifiers are now used as classifiers for the categorization of an input document. In what follows, we describe these procedures.

- *Agent for Optimization of Information Matching*

  Due to the complexity of semantic extraction, ambiguity of semantics is commonly encountered during this process. However, with the support of ontology, some of the ambiguity can be eliminated by some heuristics reasoning rules.

  TCA adopts two rules to eliminate ambiguity [7]: (1) *Association Rule.* If a text (or a small part of a text, e.g., a sentence in the text) matches several attributes of certain domain, then the key word closely associated with the attribute is used for decision. For example, let us consider the sentence “He is at the age of 40”. Here number 40 might match either attribute *age* or attribute *weight* in the ontology because both attributes appear in the form of numbers. However, an associated word *age* appears nearby, which tells us that number 40 should match attribute *age* in this case. (2) *Ontology Constraint Rule.* All the matching values should satisfy the constraints of the domain ontology, such as the range of value defined by the ontology.

- *Agent for Concept Selection and Aggregation*

  With domain vocabulary, a structured document set is converted to an indexing document set, whose elements are represented as vectors with many dimensions. We name this set as *high-dimensional indexing document set* (or HDIDS). The main goal of this agent is to lessen the dimension number of HDIDS, and the result is referred to *low-dimensional indexing document set* (LDIDS).

  We adopt COSA algorithm [3] for sieving of concepts. Those concepts with extremely low frequency of occurrence are abandoned, whereas those with extremely high frequency of occurrence are further divided into sub-concepts. After such processing by COSA, the set of concepts is more identifiable. This concept set, together with the attributes of the corresponding concepts and their parent concepts in domain ontology, and the associations among these concepts, form the vector space of LDIDS. A transform matrix, namely dimension-lessening matrix, fulfils the mapping from an HDIDS to an LDIDS. Readers are referred to [7] for the details of mapping function.

- *Agent for Training Classifiers and Text Categorization*

  After the above processes, the set of heterogeneous training documents are presented as the LDIDS of low-dimensional vector space. In TCA, RIPPER [1] is deployed here for the rule set learning with LDIDS as the input[1]. RIPPER is a method for rule set learning. It can filter out noisy data. It is competitive to C4:5 algorithm [4] in terms of precision, but is faster than C4:5 for large amount of data with noises [5]. The training classifiers are obtained after this process, and are further used as classifiers for text categorization in next step.

[1] Other methods (e.g., Naive Bayes) can also be deployed here for the rule set learning. We have actually applied RIPPER and Naive Bayes in our simulations for the comparison of categorization performance. See Section 3.

Table 1. Performance under RIPPER and Naive Bayes with/without ontology

Methods	Dimensions of vectors	$Precision$(%)	$Recall$(%)	$F1$(%)
Naive Bayes	859	95.3	89.9	92.5
RIPPER	859	97.2	92.5	94.8
TCA+Naive Bayes	397	96.1	97.7	96.9
TCA+RIPPER	397	97.7	97.7	97.7

2.4 Categorization for an Unknown Document

The TC process for an unknown document is almost the same but relatively simpler as compared with the training process. The first several steps are the same. In structurization layer, the previous training document (refer to Figure 1) is now replaced by the unknown document as the input. Then the next several steps are done as what we have done in training process. However, the last step is different. Once we get the LDIDS for an unknown input text which is waiting for classification, we just apply the classifiers we have already obtained from training to this text and do the categorization on it.

3 Testing Results

We obtained personal information from some university websites, which includes staff's basic information, e.g., name, title, office, email, etc. All staff are classified into academic staff and non-academic staff. In total, we have collected 57 documents in our test set, among which 39 are academic and 18 are non-academic. In what follows, we are going to compare the performance of categorization under the methods of RIPPER and Naive Bayes *with* and *without* ontology.

We get the ontology of staff in a CS department from an open website (`http://www.daml.org`), but with minor revision for our experiments. At the same time, we define a domain vocabulary, and the mapping from texts to corresponding ontology concepts. The testing results are given in Table 1.

In Table 1, *precision rate* denotes the accuracy of classification, which is defined as the percentage of the number of documents correctly classified into a category among the total number of documents classified into the category (some documents not belonging to the category are wrongly included); and *recall rate* denotes the percentage of the number of documents classified into a category among the total number of documents that should be classified into the category (some documents of the category are missed out); whereas $F1$ denotes an overall consideration of precision and recall rates, which is defined as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\%.$$

By definition, $F1$ reveals the deviation of $precision$ and $recall$. It can be seen from Table 1, the performance of categorization increases about 2% on average with the applying of ontology in TCA.

Here we have an interesting finding. When directly applying RIPPER, we get a classification rule as follows:

$$\texttt{if:}\ \text{vpi} \leqslant 0, \qquad \texttt{then:}\ \text{non-academic staff.}$$

This is actually a false rule, because as a word in the text, word *vpi* is meaningless when it is taken out from the context of the document. The occurrence of such false rules lies on that we do not provide sufficient number of counterexamples in our training, which results in that the false rules cannot be eliminated from the classifiers. However, when we combine RIPPER together with TCA, the classification rule we obtained is

$$\texttt{if:}\ \text{CONCEPT\_professor} \leqslant 0, \qquad \texttt{then:}\ \text{non-academic staff.}$$

The latter rule shows that those non-professor related documents would be put into non-academic staff. This finding tells that domain ontology is more effective in terms of better understanding of documents and classification performance.

4 Conclusions

Based on ML mechanism, we have proposed an architecture, namely TCA, for text categorization by addressing ontology to it. Supported by the testing results, our approach has been proven to be a successful attempt in terms of addressing ontology to TC with ML approaches. As compared with several other methods of TC, such as Naive Bayes and RIPPER, our attempt has shown its advantages in terms of uniformly processing of heterogeneous documents, less number of key words, saving of system costs, and lastly yet still an increase of categorization performance. For future study, more tests should be conducted to verify the robustness. Another promising direction should be in autonomous learning.

References

1. Cohen, W. W. Learning trees and rules with set-valued features. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 709–717, Portland, Oregon, USA, 1996.
2. Davulcu, H., G. Yang, M. Kifer, and I. V. Ramakrishnan. Computational aspects of resilient data extraction from semistructured sources. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'2000)*, pages 136–144, Dallas, Texas, USA, May 15–17, 2000.
3. Hotho, A, A. Mädche, and S. Staab. Ontology-based text document clustering. *Künstliche Intelligenz*, 16(4):48–54, 2002.
4. Quinlan, J. R. *C4:5: Programs for Machine Learning.* Morgan Kaufmann Publishers, Inc., USA, 1993.
5. Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.
6. William, S. and T. Austin. Guest editor's introduction: ontology. *IEEE Intelligent Systems*, 14(1):18–19, 1999.
7. Zhao, G. The study of ontology applied in text categorization. Master's thesis, College of Computer Science, Zhejiang University, Hangzhou, China, 2004.