Classification of E-mail Queries by Topic: Approach Based on Hierarchically Structured Subject Domain

Anna V. Zhdanova[1,2] and Denis V. Shishkin[1,2]

[1] Novosibirsk State University, Novosibirsk 630090, Russia
[2] A.P. Ershov Institute of Informatics Systems, Novosibirsk 630090, Russia
{anna, denis}@sib3.ru

Abstract. We describe a Classifier of e-mail queries, which executes text categorization by topic. The specifics of our Classifier is that it allows accurate categorization of short messages containing only a few words. This advantage is achieved by executing morphological and semantic analyses of an incoming text. Specifically, the Classifier provides an efficient information extraction and takes the meaning of words into consideration. By using the hierarchically structured subject domain and classification rules, the Classifier's engine assigns an e-mail query to the most relevant category or categories.

1 Introduction

The need for implementation of automatic classification or categorization of electronic documents is acute nowadays, due to the increasing flow of data in the World Wide Web and enlarging electronic databases. In particular, classification of electronic queries and information extraction are the key steps towards creating an automatic e-mail answering system. To handle the problem of text categorization, many methods have been proposed. Among these methods are the Naive Bayes technique [1], maximum entropy modeling [2], k-nearest neighbor classification [5], document clustering [3], rule-based method [4], and topic identification via using a hierarchical structure of the subject domain [5]. However, the precision of classification performed by using these approaches is often insufficient for constructing automatic answering systems, because the semantics of words in classified documents is not taken into consideration. Incorporating the word semantics into the classification process by using the external linguistics knowledge-base (WordNet) results in improving the classification precision [5]. The latter approach does however not take into account the word order and special words such as negation words (e.g., the texts consisting of expressions "to be" and "not to be" are assigned to the same category).

In this paper, we describe a Classifier, which makes it possible to categorize e-mail queries of different sizes including those containing only a few words, where identifying the semantics is especially important (questions, addressed to call-centers, are examples of such queries). The software presented contains an information extraction tool with included morphological analysis of words, rule-based

engine for assigning an e-mail query to the appropriate category/categories, and a database of hierarchically structured domains with the corresponding dictionaries. A hierarchical domain structure is used by the rule-based engine in order to take into account the semantics of an incoming e-mail query. The semantic analysis implemented in our Classifier is more complete compared to those employed earlier. Combination of these features results in improvement of the precision of classification.

2 Hierarchical Domain Structure

In our approach, the subject domain is represented as a hierarchy (i.e., a tree) of categories, because such domain representation makes it possible to take into consideration the semantics of the words from a query. A hierarchy is built according to the properties of objects belonging to categories. Each category is a generalization of its subcategories, while a subcategory has an "is a" or "part of" relationship with the category it belongs to. Specifically, each category is represented by its mnemonic name, relationships with other categories, and a set of regular expressions. The set of regular expressions attributed to a category should guarantee covering the semantics of all possible queries, which are to be assigned to this category. If a regular expression characterizes all the subcategories, it is replaced to a more general category. Obviously, a set of regular expressions can not be empty.

In the framework of our Classifier, a tree of categories is constructed manually by the experts who are familiar with the corresponding domain. To implement classification, we have constructed category trees for two relatively narrow domains: insurance and banking. We have used English and Russian languages. A part of the hierarchically structured insurance domain is shown in Fig. 1 (the left sub-window).

3 Regular Expressions and Classifier's Dictionary

Information extraction is often based on a dictionary of keywords. Information extraction using a dictionary of regular expressions is however more efficient, because it takes into consideration the morphology and semantics of the words from an e-mail query. Regular expressions include keywords and collocations. In addition, the Classifier's dictionary constructor allows representing a regular expression as a stem of a keyword or collocation and attributed numbers, which correspond to the lengths of the possible endings of this word or collocation. In our Classifier, using regular expressions guarantees extraction of all the relevant words independently of their morphological form and extraction of all the text components, which can be represented as a regular expression (e.g., any e-mail address). The "stem + ending length" approach is especially helpful for inflexional languages (e.g., Russian), because in this way a more complex and unnecessary here morphological analysis is avoided. For English, this approach is also useful, because it allows to differentiate relevant and irrelevant words.

For example, creating a regular expression with the meaning of an accident (i.e., crash and/or breakage) and stating that a stem "accident" has a zero or one letter length of ending, we achieve that the words "accident" or "accidents" are matched to this regular expression, but the word "accidental" is not. In addition, the "stem + ending length" approach is more competent in handling the problem of misprints and incorrectly written words in an e-mail query.

The Classifier's dictionary is a list of regular expressions, which corresponds to the chosen domain. Each regular expression has one or several pointers at the related category or categories from the tree. The functional expressions, such as the expressions for negation (e.g., "no", "isn't", "besides"), are marked by special labels. This dictionary structure is independent of a language. To make the Classifier multilingual, it is necessary to add the appropriate regular expressions in different languages to the dictionary.

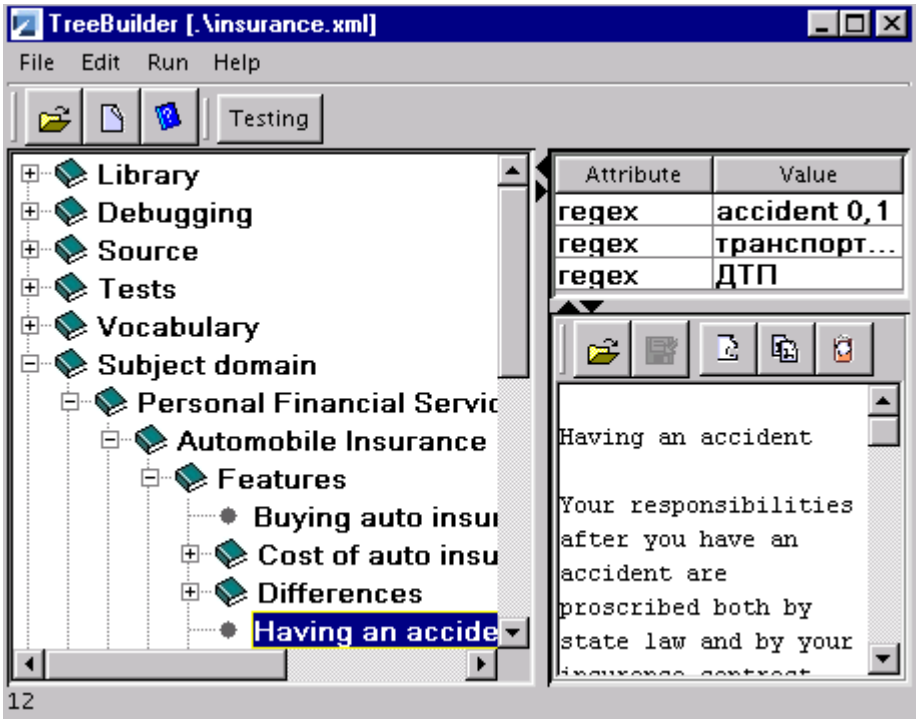

Fig. 1. Snapshot of the Classifier's user interface

4 Classification Rules

After the process of information extraction is completed, we generate a list of pointers at categories (i.e., a set of tree node numbers) and negations in the

order of location of the corresponding words (regular expressions or negation expressions). The classifying rules, used to assign a query to the most relevant category/categories associated with the list, are as follows.

Intersection: If a category and its subcategory at any level of the tree are found in the list together, the category, as being more general, is removed from the further processing in order to attain a precise classification.

Negation: The category or categories assigned to the regular expression, which follows a negation expression, are substituted by their nearest-neighbor categories at the same level of a tree.

Union: The categories, which have not been removed, are united in a new list. An e-mail query is assigned to the category or categories of this list, if none of the rules can be executed anymore.

Formally, the rules above can be executed by applying the following operations to the sets of tree nodes (i.e., categories): $A \cap B = \{x : [(x \in A)\&(\exists b \in B : x \leq b)] \vee [(x \in B)\&(\exists a \in A : x \leq a)]\}$, $\neg A = \{x : (x \notin A)\&(\exists a \in A : isBrother(x, a))\}$, $A \cup B = \{x : [(x \in A) \vee (x \in B)]\}$.

Here, A and B are the sets of tree nodes; a, b, and x are the tree nodes; $x \leq y$ is the predicate, which equals "true" when x is a descendant of node y, and equals "false" otherwise; $isBrother(x, y)$ is the predicate, which equals "true" when x and y are the sub-nodes of the same node, and equals "false" otherwise.

5 Implementation of the Classifying Rules

As a reader might have seen in the previous section, we operate with the partially ordered set $< T, \leq >$ and the algebra $\Omega =< A, \cup, \cap, \neg >$, where $A = \{B : (B \subseteq T)\&(\forall a \in T([\exists b \in B : a \leq b] \Rightarrow a \in B))\}$. The operations $\cup, \cap$ are union and intersection of sets. The operation of negation $\neg j = \{x : (x \notin j)\&(\exists a \in j, \exists y \in T : isBrother(y, a)\&x \leq y)\}$, where $j \in A$ and $isBrother(a, b) = \exists c \in T : (a \leq c)\&(b \leq c)\&(\neg\exists d \in T : (a < d < c) \vee (b < d < c))$, where $a, b \in T$.

The definition of algebra Ω uses the existence quantifiers '$\exists$'. For this reason, a straightforward implementation of the operations defined above implies searching through a set of nodes every time a rule is executed, which is time consuming. To avoid this problem, we introduce a new algebra $\Psi =< A^4 = \{< a, b, c, d >: a, b, c, d \in A\}, \cup, \cap, \neg >$ and define its operations (intersection, negation and union) in the following way:

$< a1, b1, c1, d1 > \cap < a2, b2, c2, d2 >=< (a1 \cap a2) \cup (a1 \cap b2) \cup (b1 \cap a2), (b1 \cap b2), (c1 \cap c2) \cup (c1 \cap (b1 \cup b2)) \cup (c2 \cap (b1 \cup b2)), (d1 \cap d2) \cup (d1 \cap (b1 \cup b2)) \cup (d2 \cap (b1 \cup b2)) >$, $\neg < a, b, c, d >=< c, d, a, b >$,

$< a1, b1, c1, d1 > \cup < a2, b2, c2, d2 >=< a1 \cup a2, b1 \cup b2, c1 \cup c2, d1 \cup d2 >$.

Then, we define a mapping $\phi : A \to A^4$. Let $J \in A$, then $\phi(J) =< a1, a2, a3, a4 >$, where $a1 = \{x : \exists a \in J, a \leq x\}$, $a2 = \{x : \exists a \in J, x \leq a\} = J$,

$a3 = \{x : \exists a \in J, \exists b \in T : (b \notin J)\&isBrother(a, b)\&(b \leq x)\}$,

$a4 = \{x : \exists a \in J, \exists b \in T : (b \notin J)\&isBrother(a, b)\&(x \leq b)\}$.

According to the construction above, ϕ is an isomorphism. This enables us to deal with the algebra Ψ instead of the algebra Ω. One can see that the negation

operation in the vector space Ψ is global, while the operation of the space Ω is local. In the algebra Ψ, we use the four-dimensional space in order to store the "brothers" (for the operation of negation), "descendants", and "ancestors" (to find the tree nodes corresponding to an element of the algebra Ψ quickly). In the case of the negation operation, the algebra Ψ construction provides a constant execution time of '$\neg$' operation (by using the bit set model). Hence, the model Ψ is more effective then the model Ω.

6 Used Tools

The Classifier's implementation has required construction of a few tools including TreeBuilder, LinguaEngine, and RegExParser. In principle, these tools can be used in solving natural language problems different from classification. Java is employed as a programming language.

The main goal of the program **TreeBuilder** is helping a user to create a tree (a hierarchy of categories in Classifier). The tree nodes and their sub-nodes can be easily created and associated with runnable Java objects or hard disk data (e.g., files). In addition, the TreeBuilder's graphics user interface allows one to edit the source code of rules and dictionaries, create tests, store the hierarchy and its data (the tree is stored in the XML format), and view the debug information. Copy&paste technology can be applied to any hierarchy part that can be edited. A node can be extended on another tree by linking the XML file of a tree to the node. A tree, containing such nodes, may function as a "library" for a user. Thus, the TreeBuilder's features facilitate the process of tree construction.

LinguaEngine is the rule-based engine, used by the Classifier. LinguaEngine takes a list of Java objects (a list of pointers at categories in our case) and rules as an input. The output is the objects left in the input list, when all of the rules have finished their execution. Technically, a rule is presented as a Java method. In the process of execution, a rule replaces a sub-list of objects with another list of objects.

In this paper, we have described only three classification rules. However, the rules might be more numerous after introducing less general rules for more particular cases. The engine makes it possible to unite the rules into groups and to introduce a "group-order". "Group-order" defines the group, containing the rules to be executed earlier than the rules from the other groups. For example, sometimes it is necessary to execute more important rules before the rules of less importance. A group of rules is implemented by the means of a Java class, where every boolean method represents a rule. If a rule is executed, the corresponding method returns "true" (i.e., the engine tries to execute the first rule of this group), else the method returns "false" (i.e., the engine tries to execute the next rule of the same group). When none of the rules from the current group can be executed, the engine switches to the rules of the next group according to the "group-order".

RegExParser is the tool for morphological analysis and information extraction, as described in Sec. 3.

7 Future Work

Performing its main task of text categorization, our Classifier extracts information, which can be used for creating an automatic e-mail answering system. Such a system, for example, may rely on information extraction performed by the Classifier and a template-based natural language generation [6]. Presently, if a query is classified into any category, the Classifier displays a natural language reply proposed by the creators of domain's hierarchical structure. Generation of replies in natural language may result in a higher interactivity with a user and provide an adequate amount of information for different kinds of users [7].

We believe that the performance of an automatic e-mail answering system may also be improved by identifying the genre of an incoming query. After deciding whether the e-mail document is a question, demand, or compliant, the system should act correspondingly. This strategy can be implemented by introducing a multi-dimensional hierarchy of the subject domain.

Finally, it is appropriate to notice that the presently used manual construction of category trees is a time-consuming process that requires special knowledge. For this reason, automation of this step is also desirable.

Acknowledgements The authors thank F. Dinenberg and I. Kononenko for constructing the hierarchies of categories and useful discussions, and D. Levin and D. Petunin for useful discussions.

References

1. Žižka, J., Bourek, A., Frey, L.: TEA: A Text Analysis Tool for the Intelligent Text Document Filtering. In: Proceedings of the 3-rd International Workshop on Text, Speech and Dialogue (2000).
2. Manning, C.D., Schutze, H.: Foundations of Statistical Natural Language Processing. The MIT Press (2001).
3. Makagonov, P., Sboychakov, K.: Software for Creating Domain-Oriented Dictionaries and Document Clustering in Full-Text Databases. In: Proceedings of the Second International Conference on Intelligent Text Processing and Computational Linguistics (2001).
4. Cohen, W.W.: Fast Effective Rule Induction. In: Proceedings of the Second International Conference on Machine Learning (1995).
5. Tiun, S., Abdullah, R., Kong, T.E.: Automatic Topic Identification Using Ontology Hierarchy. In: Proceedings of the Second International Conference on Intelligent Text Processing and Computational Linguistics (2001).
6. Kosseim, L., Beauregard, S., Lapalme, G.: Using Information Extraction and Natural Language Generation to Answer E-mail. In: Proceedings of the 5-th International Conference on Applications of Natural Language to Information Systems (2000).
7. Bergholtz, M., Johannesson, P.: Validating Conceptual Models - Utilizing Analysis Patterns as an Instrument for Explanation Generation. In: Proceedings of the 5-th International Conference on Applications of Natural Language to Information Systems (2000).