Feature Selection for Natural Language Call Routing Based on Self-Adaptive Genetic Algorithm

To cite this article: A Koromyslova et al 2017 IOP Conf. Ser.: Mater. Sci. Eng. 173 012008

View the article online for updates and enhancements.
Feature Selection for Natural Language Call Routing Based on Self-Adaptive Genetic Algorithm

A Koromyslova¹, M Semenkina¹, R Sergienko²

1Reshetnev Siberian State Aerospace University, Krasnoyarsky rabochy avenue 31, Krasnoyarsk, Russia, 660037
2Ulm University, Lise-Meitner-Str. 9, Ulm, Germany, 89081

E-mail: akoromyslova@mail.ru, semenkina88@mail.ru, roman.sergienko@uni-ulm.de

Abstract: The text classification problem for natural language call routing was considered in the paper. Seven different term weighting methods were applied. As dimensionality reduction methods, the feature selection based on self-adaptive GA is considered. k-NN, linear SVM and ANN were used as classification algorithms. The tasks of the research are the following: perform research of text classification for natural language call routing with different term weighting methods and classification algorithms and investigate the feature selection method based on self-adaptive GA. The numerical results showed that the most effective term weighting is TRR. The most effective classification algorithm is ANN. Feature selection with self-adaptive GA provides improvement of classification effectiveness and significant dimensionality reduction with all term weighting methods and with all classification algorithms.

1. Introduction
Natural language call routing is an important problem in the design of modern automatic call services and the solving of this problem could lead to improvement of the call service [21]. Generally natural language call routing can be considered as two different problems. The first one is speech recognition of calls and the second one is topic categorization of users utterances for further routing. Topic categorization of users utterances can be also useful for multidomain spoken dialogue system design [12]. In this work we treat call routing as an example of a text classification application.

In the vector space model [16] text classification is considered as a machine learning problem. The complexity of text categorization with a vector space model is compounded by the need to extract the numerical data from text information before applying machine learning algorithms. Therefore, text classification consists of two parts: text preprocessing and classification algorithm application using the obtained numerical data. Text preprocessing comprises three stages:

- Textual feature extraction.
- Term weighting
- Dimensionality reduction.

The first one is the textual feature extraction based on raw preprocessing of the documents. This process includes deleting punctuation, transforming capital letters to lowercase, and additional procedures such as stop-words filtering [4] and stemming [14]. Stop-words list contains pronouns, prepositions, articles and other words that usually have no importance for the classification. Using stemming it is possible to join different forms of the same word into one textual feature.

The second stage is the numerical feature extraction based on term weighting. For term weighting we use “bag-of-words” model, in which the word order is ignored. There exist different unsupervised and supervised term weighting methods. The most well-known unsupervised term weighting method is TFIDF [15]. The following supervised term weighting methods are also considered in the paper:
Gain Ratio (GR) [3], Confident Weights (CW) [10], Term Second Moment (TM2) [22], Relevance Frequency (RF) [11], Term Relevance Ratio (TRR) [9], and Novel Term Weighting (NTW) [18]; these methods involve information about the classes of the documents.

As a rule, the dimensionality for text classification problems is high even after stop-words filtering and stemming. Due to the high dimensionality, the classification may be inappropriate time-consuming, especially for real-time systems such as natural language call routing. Therefore, the next stage of preprocessing is the dimensionality reduction based on numerical features; it is possible with feature selection or feature transformation. In our research we use a feature selection method based on genetic algorithm or GA-based wrapper.

One of the most complicated problems with GA applications is setting algorithm parameters. A conventional genetic algorithm has at least three methods of selection (proportional, tournament, and rank), three methods of recombination (one-point, two-point, and uniform). Mutation probability requires tuning as well. The amount of various combinations can be estimated at tens. Exhaustive search of combinations requires a lot of time and computational power, especially for such time-consuming problems as GA applications for machine learning. Parameters combination selection at random can be also insufficient as algorithm efficiency on same problem can differ very much for various parameters setting. This problem can be solved with self-configuring GA [17] or co-evolutionary GA [19]. Therefore, we propose a use of self-adaptive GA for the feature selection in the field of natural language call routing.

As classification algorithms we use the k-NN algorithm, the linear SVM, and artificial neural networks (ANN). Some comparative studies of machine learning algorithms in the field of text classification showed high classification effectiveness of k-NN, SVM-based algorithms, and ANN [2, 7, 8, 10, 13].

The tasks of our research are the following:
- Perform research of text classification for natural language call routing with different term weighting methods and classification algorithms.
- Investigate the feature selection method based on self-adaptive GA.

The paper is organized as follows: In Section 2, we describe the considered corpus for natural language call routing. Section 3 describes the considered term weighting methods. The GA-based feature selection is described in Section 4. The self-adaptive GA are described in Section 5. Section 6 contains short description of classification algorithms. The results of numerical experiments are presented in Section 7. Finally, we provide concluding remarks in Section 8.

2. Corpus description
The data for testing and evaluation consists of 292,156 user utterances recorded in English language from caller interactions with commercial automated agents. Utterances are short and contain only one phrase for further routing. The database contains calls in textual format after speech recognition. The database is provided by the company Speech Cycle (New York, USA). Utterances from this database are manually labelled by experts and divided into 20 classes (such as appointments, operator, bill, internet, phone and technical support). One of them is a special class TE-NOMATCH which includes utterances that cannot be put into another class or can be put into more than one class.

The database contains 45 unclassified calls and they were removed. The database contains also 23,561 empty calls without any words. These calls were placed in the class TE-NOMATCH automatically and they were also removed from the database. As a rule, the calls are short in the database; many of them contain only one or two words. The average length of an utterance is 4.66 words, the maximal length is 19 words. There are a lot of identical utterances in the database; the corpus contains only 24,458 unique non-empty classified calls. The corpus is unbalanced. The largest class contains 27.05% and the smallest one contains 0.04% of the unique calls.

Due to the very high frequency of a small number of utterances in the corpus, we formulate two different problem definitions.
Problem definition 1. The whole database with 268,550 classified non-empty calls is used for training and test sets forming. Numbers of repetitions of the utterances in training and test sets are used as weights for classification. This problem definition is the closest to the real situation but frequently repeated utterances decrease difference between preprocessing and classification methods. Additionally, there are some identical utterances in training and test sets simultaneously. In this case the over-fitting problem of classification may be hidden. Therefore, this problem definition is not very appropriate for the comparative study.

Problem definition 2. Before training and test samples forming, all utterance duplicates were removed from the database. It means that there is no intersection between training and test sets and frequency of utterances is ignored.

Therefore, the problem definition 1 is suitable for the quality estimation of the real natural language call routing system; the problem definition 2 is the most appropriate for the comparative study of different preprocessing and classification methods.

For statistical analysis we performed 20 different divisions of the database into training and test samples randomly. This procedure was performed for two problem definitions separately. The train samples contain 90% of the calls and the test samples contain 10% of the calls. For each training sample we have designed a dictionary of unique words which appear in the training sample after deleting punctuation and transforming capital letters to lowercase. The size of the dictionary varies from 3,275 to 3,329 words for problem definition 1 and from 3,277 to 3,311 for problem definition 2.

3. Term weighting methods
As a rule, term weighting is a multiplication of two parts: the part based on the term frequency in a document (TF) and the part based on the term frequency in the whole training database. The TF-part is fixed for all considered term weighting methods and is calculated as following:

\[ TF_{ij} = \log(f_{ij} + 1) \; ; \; tf_{ij} = \frac{n_{ij}}{N_j}, \]

where \( n_{ij} \) is the number of times the \( i^{th} \) word occurs in the \( j^{th} \) document, \( N_j \) is the document size (number of words in the document). The second part of the term weighting is calculated once for each word from the dictionary and does not depend on an utterance for classification. We consider seven different methods for the calculation of the second part of term weighting.

3.1. Inverse Document Frequency (IDF)
IDF is a well-known unsupervised term weighting method which was proposed in [15]. There are some modifications of IDF and we use the most popular one:

\[ idf_{ij} = \log \left( \frac{|D|}{n_i} \right), \]

where \(|D|\) is the number of documents in the training set and \( n_i \) is the number of documents that have the \( i^{th} \) word.

3.2. Gain Ratio (GR)
Gain Ratio (GR) is mainly used in term selection [24], but in [3] it was shown that it could also be used for weighting terms. The definition of GR is as follows:
\[ \text{GR}(t_i, c_j) = \max_{c_j \in C} \text{GR}(t_j, c_j) \]

where \( C \) is a set of all classes.

### 3.3. Confident Weights (CW)

This supervised term weighting approach has been proposed in [20]. Firstly, the proportion of documents containing term \( t \) is defined as the Wilson proportion estimate \( p(x, n) \) by the following equation:

\[
p(x, n) = \frac{x + 0.5Z_{\alpha/2}^2}{n + Z_{\alpha/2}^2},
\]

where \( x \) is the number of documents containing the term \( t \) in the given corpus, \( n \) is the number of documents in the corpus and \( \Phi\left(Z_{\alpha/2}\right) = \alpha/2 \), where \( \Phi \) is the \( t \)-distribution (Students law) when \( n < 30 \) and the normal distribution when \( n \geq 30 \).

In this work \( \alpha=0.95 \) and \( 0.5Z_{\alpha/2}^2 = 1.96 \) (as recommended by the authors of the method). For each term \( t \) and each class \( c \) two functions \( p_{\text{pos}}(x, n) \) and \( p_{\text{neg}}(x, n) \) are calculated. For \( p_{\text{pos}}(x, n) \) \( x \) is the number of documents which belong to the class \( c \) and have term \( t \); \( n \) is the number of documents which belong to the class \( c \). For \( p_{\text{neg}}(x, n) \) \( x \) is the number of documents which have the term \( t \) but do not belong to the class \( c \); \( n \) is the number of documents which do not belong to the class \( c \). The confidence interval \((p^-, p^+)\) at 0.95 is calculated using the following equation:

\[
M = 0.5Z_{\alpha/2}^2 \sqrt{\frac{p(1-p)}{n + Z_{\alpha/2}^2}},
\]

\[ p^- = p - M; \quad p^+ = p + M. \]

The strength of the term \( t \) in the category \( c \) is defined as the follows:

\[
\text{str}(t, c) = \begin{cases} 
\log_2 \frac{2p^-_{\text{pos}}}{p_{\text{pos}} + p_{\text{neg}}}, & \text{if } p_{\text{pos}} > p_{\text{neg}}^+; \\
0, & \text{otherwise.}
\end{cases}
\]
The maximum strength \((Maxstr)\) of the term \(t_i\) is calculated as follows:

\[
Maxstr(t_i) = \max_{c_j \in C} str(t_i, c_j)^2.
\]

3.4. Term Second Moment (TM2)

This supervised term weighting method was proposed in [22]. Let \(P(c_j|t)\) be the empirical estimation of the probability that a document belongs to the category \(c_j\) with the condition that the document contains the term \(t\); \(P(c_j)\) is the empirical estimation of the probability that a document belongs to the category \(c_j\) without any conditions. The idea is the following: the more \(P(c_j|t)\) is different from \(P(c_j)\), the more important the term \(t_i\) is. Therefore, we can calculate the term weight as the following:

\[
TM2(t_i) = \sum_{j=1}^{|C|} (P(c_j|t) - P(c_j))^2,
\]

where \(C\) is a set of all classes.

3.5. Relevance Frequency (RF)

The RF term weighting method was proposed in [11] and is calculated as the following:

\[
rf(t_i) = \max_{c_j \in C} rf(t_i, c_j),
\]

\[
rf(t_i, c_j) = \log_2 \left( 2 + \frac{a_j}{\max_{c_k \in C \setminus \{c_j\}} a_k} \right),
\]

where \(a_j\) is the number of documents of the category \(c_j\) which contain the term \(t_i\) and \(\overline{a_j}\) is the number of documents of all the other categories which also contain this term.

3.6. Term Relevance Ratio (TRR)

The TRR method [9] uses \(tf\) weights and it is calculated as the following:

\[
TRR(t_i, c_j) = \log_2 \left( 2 + \frac{P(t_i|c_j)}{\overline{P(t_i|\overline{c_j})}} \right)
\]

\[
P(t_i|c_j) = \frac{\sum_{k=1}^{V} f_{ik}}{\sum_{l=1}^{T_c} \sum_{k=1}^{V} f_{lk}},
\]

\[
TRR(t_i) = \max_{c_j \in C} TRR(t_i, c_j),
\]

where \(c_j\) is a class of the document, \(\overline{c_j}\) is all of the other classes of \(c_j\), \(V\) is the vocabulary of the training data and \(T_c\) is the document set of the class \(c\).
3.7. Novel Term Weighting (NTW)
This method was proposed in [1, 18]. The details of the procedure are the following. Let \( L \) be the number of classes; \( n_i \) is the number of documents which belong to the \( i \)th class; \( N_{ij} \) is the number of occurrences of the \( j \)th word in all documents from the \( i \)th class. \( T_{ij} = N_{ij}/n_i \) is the relative frequency of occurrences of the \( j \)th word in the \( i \)th class; \( R_j = \max_i T_{ij} \); \( S_j = \arg \max_i T_{ij} \) is the class which we assign to the \( j \)th word. The term relevance \( C_j \) is calculated by the following:

\[
C_j = \frac{1}{\sum_{i=1}^{L} T_{ij}} \cdot \left( R_j - \frac{1}{L-1} \cdot \sum_{i=1, i \neq S_j}^{L} T_{ij} \right)
\]

4. GA-based feature selection
The popular kind of wrappers is feature selection based on genetic algorithms (GA). GA is used as an optimization algorithm for finding the optimal or the sub-optimal subset of the original feature set with the predefined classification algorithm [23]. The details of the GA-based wrapper are the following:

1. Randomly initialize the population of the binary strings with the length equals to dimensionality of the considered classification problem. Set generation counter \( g = 0 \).
2. Apply the classification algorithm on the validation set for all individuals form the population (with including only features with 1 value in the chromosomes)
3. Set the classification effectiveness measure (i.e. \( F \)-score) on the validation set as a fitness function value for all individuals.
4. Check if the generation counter is greater than some predefined value \( G \) (maximal number of generations): if \( g > G \) than go to the step 7; otherwise go to step 5.
5. Using GA operators: selection, crossover and mutation, form the next population.
6. Increment the generation counter \( g = g + 1 \). Go to step 2.
7. Put the best individual as a solution of the feature selection procedure; apply the chosen feature subset for the test set.

5. Self-adaptive GA
For solving the problem of GA setting parameters we use the self-configuring algorithm that was proposed in [17]. The scheme of the self-configuring GA is presented in Figure 1.

In the self-configuring GA (SCGA), different types of selection, recombination, and different levels of mutation are performed simultaneously. In the beginning of the algorithm, all types of GA operators have the same probability to be use for a new off-spring generation. After that, the dynamic adaptation of probabilities is performed according “usefulness” of GA operator types in terms of fitness function.
The details of the self-adjusting GA are the following:

1. Put the probabilities of all used types of GA operators: \( p_{ij} = \frac{1}{N_i} \), where \( j = 1, \ldots, N_i \), \( N_i \) is the number of types of the \( i \)th operator, \( i = 1, \ldots, N \), \( N \) is the number of GA operators. In our case \( N=3 \): selection, recombination, and mutation.

2. Set threshold probabilities for all used types of GA operators: \( p_{ij} = 3/(10 \cdot N_i) \).

3. Generate new population. For each off-spring we randomly choose types of selection, recombination, and mutation according probabilities \( p_{ij} \).

4. Recalculate the probabilities with the following:
   4.1. For each \( i \)th operator do:
   4.1.1. Set \( S_i = 0 \).
   4.1.2. For each \( j \)th type of the \( i \)th operator do:
   4.1.2.1. If \( p_{ij} < \overline{p}_{ij} + \frac{1}{(T \cdot N_i)} \) AND \( p_{ij} > \overline{p}_{ij} \), where \( T \) is the number of generations, then:
   \[ S_i = S_i + (p_{ij} - \overline{p}_{ij}); \]
   \[ p_{ij} = \overline{p}_{ij}. \]
   4.1.2.2. If \( p_{ij} > \overline{p}_{ij} + \frac{1}{(T \cdot N_i)} \) then: \( S_i = S_i + 1/(T \cdot N_i); \)
   \[ p_{ij} = p_{ij} - 1/(T \cdot N_i). \]
   4.1.2.3. Calculate average fitness function \( F_j \) of all off-springs of the current generation that were generated with the \( j \)th type of the \( i \)th operator.
   4.1.3. Find the best type \( d \) of the \( i \)th operator with the maximal fitness function on the current generation and recalculate its probability: \( p_{id} = p_{id} + S_i \)
   5. Check stop criterion. If TRUE then: END; else: go to the step 3.

Moreover two co-evolutionary algorithms (CEA) [19] have been realized. Both use island model with four islands and cooperative-competitive strategy for resources allocating. First of them (CEA) used four best genetic algorithms (for different types of problems), second one use four independent (on the start and on the probabilities recalculation stage) SCGA (CESCGA).

6. Classification Algorithms
As classification algorithms we use the k-NN algorithm, the linear SVM algorithm, and ANN. The classification criterion is the macro F-score [6] which is appropriate for classification problems with
unbalanced classes. For k-NN we performed validation of $k$ from 1 to 15 on the validation sample. We used 80% of the train sample for the first level of learning and 20% for the validation. The same scheme of validation is used for feature selection based on self-adjusting GA.

6.1. Genetic Algorithms for Automated Artificial Neural Network Design

Artificial neural network (ANN) is a set of interconnected neurons. Typically, the transfer functions of all the neurons in the neural network are fixed, and the weights are the parameters of the neural network.

In case of GA using for ANN weights tuning: weights are recorded sequentially in the chromosome as a binary code. The number of bits that are used to encode a single weighting coefficient depends on the accuracy and the spread of possible values of weights.

In case of GA using for ANN structure design (GA-ANN):

1. Hidden layers are coded sequentially;
2. Each neuron is encoded in four bits;
3. Each neuron will exist with probability equal to 0.3.
4. If in the network a neuron is not presented, its place in the chromosome is marked with zeros.

Otherwise, it is randomly selected as one of the fifteen activation functions, whose number is written in binary code.

Since the efficiency of the genetic algorithm for ANN design depends on the dimension of the problem in hand, it is reasonable to avoid the use of uninformative features. The modification of the genetic algorithm for the choice of the most informative features during the automated design of ANN assumes the use of additional bits in the GA chromosomes. These bits determine whether an input is included in the input layer or not.

7. Results of numerical experiments

Tables 1-10 show the results of the numerical experiments for problem definitions 1 and 2 without dimensionality reduction and after feature selection with the self-adaptive GA-based wrapper. The best results in terms of $F$-score are bold (for each case independently). The column “Features” contains number of features after GA-based wrapper performing.

The numerical results showed that the most effective term weighting is TRR. The most effective classification algorithm is ANN. The most effective optimization technique is SCGA. Feature selection with self-configuring GA provides improvement of classification effectiveness and significant dimensionality reduction with all term weighting methods and with all classification algorithms.

### Table 1. Results for problem definition 1 with k-NN

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>All terms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.778</td>
<td>0.800</td>
</tr>
<tr>
<td>GR</td>
<td>0.759</td>
<td>0.795</td>
</tr>
<tr>
<td>CW</td>
<td>0.763</td>
<td>0.803</td>
</tr>
<tr>
<td>RF</td>
<td>0.822</td>
<td>0.863</td>
</tr>
<tr>
<td>TM2</td>
<td>0.742</td>
<td>0.778</td>
</tr>
<tr>
<td>TRR</td>
<td><strong>0.851</strong></td>
<td><strong>0.887</strong></td>
</tr>
<tr>
<td>NTW</td>
<td>0.847</td>
<td>0.881</td>
</tr>
</tbody>
</table>

### Table 2. Results for problem definition 2 with k-NN

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>All terms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.643</td>
<td>0.667</td>
</tr>
<tr>
<td>GR</td>
<td>0.629</td>
<td>0.671</td>
</tr>
<tr>
<td>Term weighting method</td>
<td>F-score</td>
<td>Features</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>----------</td>
</tr>
<tr>
<td>All terms</td>
<td>Feature selection</td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.749</td>
<td>0.762</td>
</tr>
<tr>
<td>GR</td>
<td>0.724</td>
<td>0.784</td>
</tr>
<tr>
<td>CW</td>
<td>0.776</td>
<td>0.803</td>
</tr>
<tr>
<td>RF</td>
<td>0.798</td>
<td>0.851</td>
</tr>
<tr>
<td>TM2</td>
<td>0.712</td>
<td>0.765</td>
</tr>
<tr>
<td>TRR</td>
<td>0.812</td>
<td>0.874</td>
</tr>
<tr>
<td>NTW</td>
<td>0.803</td>
<td>0.869</td>
</tr>
</tbody>
</table>

**Table 3.** Results for problem definition 1 with SVM

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>All terms</td>
<td>Feature selection</td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.627</td>
<td>0.668</td>
</tr>
<tr>
<td>GR</td>
<td>0.619</td>
<td>0.659</td>
</tr>
<tr>
<td>CW</td>
<td>0.632</td>
<td>0.672</td>
</tr>
<tr>
<td>RF</td>
<td>0.639</td>
<td>0.681</td>
</tr>
<tr>
<td>TM2</td>
<td>0.611</td>
<td>0.647</td>
</tr>
<tr>
<td>TRR</td>
<td>0.654</td>
<td>0.693</td>
</tr>
<tr>
<td>NTW</td>
<td>0.648</td>
<td>0.685</td>
</tr>
</tbody>
</table>

**Table 4.** Results for problem definition 2 with SVM

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>All terms</td>
<td>Feature selection</td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.699</td>
<td>0.796</td>
</tr>
<tr>
<td>GR</td>
<td>0.731</td>
<td>0.784</td>
</tr>
<tr>
<td>CW</td>
<td>0.766</td>
<td>0.812</td>
</tr>
<tr>
<td>RF</td>
<td>0.837</td>
<td>0.850</td>
</tr>
<tr>
<td>TM2</td>
<td>0.715</td>
<td>0.771</td>
</tr>
<tr>
<td>TRR</td>
<td>0.844</td>
<td>0.891</td>
</tr>
<tr>
<td>NTW</td>
<td>0.832</td>
<td>0.876</td>
</tr>
</tbody>
</table>

**Table 5.** Results for problem definition 1 with SCGA-ANN

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>All terms</td>
<td>Feature selection</td>
<td></td>
</tr>
<tr>
<td>IDF</td>
<td>0.679</td>
<td>0.684</td>
</tr>
<tr>
<td>GR</td>
<td>0.662</td>
<td>0.675</td>
</tr>
<tr>
<td>CW</td>
<td>0.654</td>
<td>0.691</td>
</tr>
<tr>
<td>RF</td>
<td>0.682</td>
<td>0.713</td>
</tr>
<tr>
<td>TM2</td>
<td>0.621</td>
<td>0.663</td>
</tr>
<tr>
<td>TRR</td>
<td>0.703</td>
<td>0.727</td>
</tr>
<tr>
<td>NTW</td>
<td>0.699</td>
<td>0.717</td>
</tr>
</tbody>
</table>

**Table 6.** Results for problem definition 2 with SCGA-ANN
8. Conclusions

The text classification problem for natural language call routing was considered in the paper. Seven different term weighting methods were applied. As dimensionality reduction methods, the feature

<table>
<thead>
<tr>
<th>Term weighting method</th>
<th>F-score</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDF</td>
<td>0.673</td>
<td>0.78</td>
</tr>
<tr>
<td>GR</td>
<td>0.711</td>
<td>0.758</td>
</tr>
<tr>
<td>CW</td>
<td>0.739</td>
<td>0.795</td>
</tr>
<tr>
<td>RF</td>
<td>0.82</td>
<td>0.828</td>
</tr>
<tr>
<td>TM2</td>
<td>0.69</td>
<td>0.754</td>
</tr>
<tr>
<td>TRR</td>
<td>0.825</td>
<td>0.866</td>
</tr>
<tr>
<td>NTW</td>
<td>0.813</td>
<td>0.852</td>
</tr>
</tbody>
</table>

8. Conclusions

The text classification problem for natural language call routing was considered in the paper. Seven different term weighting methods were applied. As dimensionality reduction methods, the feature
selection based on self-configuring GA is considered. k-NN, linear SVM and ANN were used as classification algorithms.

The numerical results showed that the most effective term weighting is TRR. The most effective classification algorithm is ANN. Feature selection with self-configuring GA provides improvement of classification effectiveness and significant dimensionality reduction with all term weighting methods and with all classification algorithms.

Acknowledgements
The reported study was funded by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Region Science and Technology Support Fund to the research project №6-41-240822.

References


