A Novel Approach to Neural Network Design for Natural Language Call Routing

Roman Sergienko[1], Oleg Akhtiamov[2], Eugene Semenkin[2] and Alexander Schmitt[1]
[1]*Institute of Communications Engineering, Ulm University, Albert-Einstein-Allee 43, Ulm, Germany*
[2]*Institute of Computer Science and Telecommunications, Siberian State Aerospace University, Krasnoyarsk, Russia*

Keywords: Natural Language Processing, Classification, Artificial Neural Network, Ward Net, Error Backpropagation Algorithm, Genetic Algorithm, Supervised Learning.

Abstract: A novel approach to artificial neural network design using a combination of determined and stochastic optimization methods (the error backpropagation algorithm for weight optimization and the classical genetic algorithm for structure optimization) is described in this paper. The novel approach to GA-based structure optimization has a simplified solution representation that provides effective balance between the ANN structure representation flexibility and the problem dimensionality. The novel approach provides improvement of classification effectiveness in comparison with baseline approaches and requires less computational resource. Moreover, it has fewer parameters for tuning in comparison with the baseline ANN structure optimization approach. The novel approach is verified on the real problem of natural language call routing and shows effective results confirmed with statistical analysis.

1 INTRODUCTION

Natural language call routing is an important problem for modern automatic call service design. A solution of this problem could provide improvement to the call service (Suhm et al., 2002). Generally, natural language call routing can be considered as two different problems. The first one is the speech recognition of calls and the second one is the call categorization for further routing. This paper focuses on text categorization methods applied to call routing.

In the vector space model (Sebastiani, 2002) text categorization is considered as a machine learning problem. The complexity of the text categorization with the vector space model is compounded by the need to extract the numerical data from text information before applying machine learning methods. Therefore, text categorization consists of two parts: a text preprocessing and a classification using the numerical data obtained.

Text preprocessing for text classification can be performed with term weighting. There exist different unsupervised and supervised term weighting methods. The most well-known unsupervised term weighting method is TF-IDF (Salton and Buckley, 1988). The following supervised term weighting methods are also considered in the paper: Gain Ratio (Debole and Sebastiani, 2004), Confident Weights (Soucy and Mineau, 2005), TM2 (Xu and Li, 2007), Relevance Frequency (Lan et al., 2009), Term Relevance Ratio (Ko, 2012), and Novel Term Weighting (Gasanova et al., 2014).

As a classification algorithm we propose artificial neural networks. An artificial neural network (ANN) is a powerful tool for information processing and classification. The most popular method of ANN learning is the error backpropagation algorithm (Hecht-Nielsen, 1989), which allows training an ANN within a reasonable period of time with an appropriate result. However, this approach does not take into consideration the structure of an ANN (the number, location and activation functions of neurons). All these features have a significant influence on the training quality and training speed; it would be effective to control them in order to improve the result of the training process. There are some approaches to ANN structure optimization based on evolutionary algorithms, such as a genetic algorithm (GA) (Bukhtoyarov and Semenkina, 2010; Bukhtoyarov et al., 2011). It is also possible to optimize neuron weights with GA (Whitley et al., 1990). However, GA-based approaches require a lot of computational time and computational resources due to the high

Sergienko R., Akhtiamov O., Semenkin E. and Schmitt A..
A Novel Approach to Neural Network Design for Natural Language Call Routing.
DOI: 10.5220/0005515401020109
In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2015), pages 102-109
ISBN: 978-989-758-122-9

dimensionality of the optimization problem and the complexity of the fitness function calculation.

We propose a novel ANN structure optimization approach which is based on a combination of determinate and stochastic optimization methods: the error backpropagation algorithm and the classical genetic algorithm (GA); the GA-based part of the algorithm has a simplified solution representation. The novel approach provides an effective ANN structure optimization within a reasonable computational period.

This paper is organized as follows. In Section 2 we describe the problem in hand and the database. Section 3 describes considered term weighting methods and the feature transformation method. Section 4 reports on the baseline and novel approaches of ANN application to the problem. Finally, we provide concluding remarks and directions for future investigations in Section 5.

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. The database contains calls in textual format after speech recognition. The database is provided by company *Speech Cycle* (New York, USA). Utterances from this database are manually labelled by experts and divided into 20 classes (appointments, operator, bill, internet, phone, technical support etc.). 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 contained also 23,561 empty calls without any words. These calls were removed. As a rule, the calls are short; many of them contain only one or two words. So, there are many duplicated utterances in the database and these utterance duplicates were also removed. After that the database contains 24,458 unique non-empty classified calls.

We performed 20 different random splits of the database. Then we divided each one into training and test samples in two ways using different proportions: 9/1 and 7/3. Training and test sets should not be the same; the ANN structure optimization algorithm uses the first proportion, and the second one is used for testing the best solution obtained with the algorithm. For each training sample we have designed a dictionary of unique words which appear in the sample. The size of the dictionary varies from 3,275 to 3,329 words for different variants.

3 TEXT PRE-PROCESSING

After the generation of training and test samples, we performed term weighting. As a rule, term weighting is a multiplication of two parts: the part based on term frequency in a document (TF) and the part based on term frequency in the whole database. The TF-part is fixed for all considered term weighting methods and calculated in the following way:

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

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 term weighting is calculated once for each word from the dictionary and does not depend on an utterance for classification. We consider 7 different methods for the calculation of the second part of term weighting.

3.1 IDF

IDF is a well-known unsupervised term weighting method, which was proposed in (Salton and Buckley, 1988). There are some modifications of IDF and we use the most popular one:

$$idf_i = \log\frac{|D|}{n_i},$$

where $|D|$ is the number of documents in a 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 (Yang and Pedersen, 1997). However, in (Debole and Sebastiani, 2004) it was shown that it could also be used for weighting terms, since its value reflects the importance of a term. The definition of GR is as follows:

$$GR(t_i, c_j) = \frac{\sum_{c\in\{c_j,\overline{c_j}\}}\sum_{t\in\{t_j,\overline{t_j}\}} P(t,c)\cdot\log\frac{P(t,c)}{P(t)\cdot P(c)}}{-\sum_{c\in\{c_j,\overline{c_j}\}} P(c)\cdot\log P(c)},$$

where *P(t, c)* is the relative frequency that a document contains the term *t* and belongs to the category *c*; *P(t)* is the relative frequency that a document contains the term *t* and *P(c)* is the relative frequency that a document belongs to category *c*.

Then, the weight of the term t_i is the max value between all categories as follows:

$$GR(t_i) = \max_{c_j \in C} GR(t_i, c_j),$$

where *C* is a set of all classes.

3.3 Confident Weights (CW)

The method uses the special value *Maxstr* as an analogy of IDF.

The method firstly estimates the probability *P*, that a document contains a term *t* with a confidence interval for every category c_j, to get $P(t \mid c_j)$ and $P(t \mid \overline{c_j})$ with a confidence interval. Let *M* denote the lower bound of $P(t \mid c_j)$ and *N* denote the upper bound of $P(t \mid \overline{c_j})$. The strength of a term t_i considering c_j is defined as:

$$str(t,c) = \begin{cases} \log_2\left(\frac{2M}{M+N}\right), if\,(M > N) \\ 0, otherwise \end{cases}.$$

The maximum strength (*Maxstr*) of the term *t* is calculated as follows:

$$Maxstr(t) = \max_{c \in C}\left(str(t,c)\right)^2.$$

3.4 TM2 (Second Moment of a Term)

Let $P(c_j \mid t)$ be the probability that a document belongs to the category c_j with the condition that the document contains the term *t* and belongs to the category *c*; $P(c_j)$ is the probability that a document belongs to category *c* without any conditions. The idea is the following: the more $P(c_j \mid t)$ is different from $P(c_j)$, the more important the term t_i is. Therefore, we can calculate the term weight as the following:

$$TM(t_i) = \sum_{j=1}^{|C|}\left(P(c_j \mid t) - P(c_j)\right)^2.$$

3.5 Relevance Frequency (RF)

The RF value is calculated as follows:

$$rf(t_i, c_j) = \log_2\left(2 + \frac{a_j}{\max\{1, \overline{a_j}\}}\right),$$

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

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 remaining categories which also contain this term.

3.6 Term Relevance Ratio (TRR)

The TRR method (Ko, 2012) uses *tf* weights and is calculated as follows:

$$TRR(t_i, c_j) = \log_2\left(2 + \frac{P(t_i \mid c_j)}{P(t_i \mid \overline{c_j})}\right),$$

$$P(t_i \mid c) = \frac{\sum_{k=1}^{|T_c|} tf_{ik}}{\sum_{l=1}^{|V|}\sum_{k=1}^{|T_c|} tf_{lk}},$$

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

where c_j is the class of the document, $\overline{c_j}$ is all 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 (Gasanova et al., 2014).

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 articles 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:

$$C_j = \frac{1}{\sum_{i=1}^{L} T_{ij}}\left(R_j - \frac{1}{L-1}\sum_{\substack{i=1 \\ i \neq S_j}}^{L} T_{ij}\right).$$

3.8 Feature Transformation Method

We propose a feature transformation method based

on terms belonging to classes. The idea is to assign each term from the dictionary to the most appropriate class. Such an assignment is performed during the calculation of GR, CW, RF, TRR and NTW. With TF-IDF and TM2 we can also assign one class for each term using the relative frequency of the word in classes:

$$S_j = \arg\max_{c \in C} \frac{n_{jc}}{N_c},$$

where S_j is the most appropriate class for the j^{th} term, c is an index of a class, C is a set of all classes, n_{jc} is the number of documents of the c^{th} class which contain the j^{th} term, N_c is the number of all documents of the c^{th} class.

After assigning each word to one class and term weighting we can calculate the sums of term weights in a document for each class. We can put these sums as new features of the text classification problem. Therefore, such a method reduces the dimensionality radically; the dimensionality of the classification problem equals the number of classes.

4 ANN STRUCTURE AND WEIGHT OPTIMIZATION METHODS

4.1 Error Backpropagation Algorithm for Weight Optimization

The first and basic method of ANN learning is the error backpropagation algorithm. At some tasks, it works effectively itself. There is an effective ANN implementation with the error backpropagation algorithm built in the *Rapid Miner* software package (Shafait et al., 2010). In this study, results obtained with *Rapid Miner* are used as a reference point for other algorithms. The described corpus was processed with all the pre-processing methods, and then the classification problem was solved with this tool. The number of output neurons is equal to the number of classes; a value close to 1 on an output means that the ANN has chosen the corresponding class. Otherwise, there is a value close to 0 on the output. Also, there is a rule according to which *Rapid Miner* chooses the ANN structure. As default settings, there is one hidden layer. The number of neurons in it is calculated as follows:

$$n_h = \frac{n_i + n_o}{2} + 1,$$

where n_i is the number of input neurons, n_o is the number of output neurons.

We use macro *F*-score as the main criterion of the classification effectiveness. It is calculated as follows:

$$P_i = \frac{|Dr_i \cap Df_i|}{|Df_i|},\ R_i = \frac{|Dr_i \cap Df_i|}{|Dr_i|},$$

$$F(a) = \frac{1}{a\frac{1}{P} + (1-a)\frac{1}{R}},\ a \in [0,1],$$

where i is the number of a class, Dr_i is the set of objects in a test set which belong to this class, Df_i is the set of objects in the test set classified by the system to this class. We use macro F_1 score (a=0.5).

The following settings of the error backpropagation algorithm in *RapidMiner* are used: learning rate = 0.5, momentum = 0.2, training cycles = 500, using decay. Learning rate and momentum were discretised on the interval [0.1, 0.5] with the discretization step 0.1. Then exhaustive search among all possible pairs was performed. The algorithm with the specified pair of values had the best *F*-score.

Then we performed a *t*-test for all different pairs of the preprocessing methods. The *t*-test demonstrated a statistically significant advantage of the TRR pre-processing method with the confidential probability 0.95. Therefore, it was decided to test other approaches using the TRR method only.

Table 1: Results of the pre-processing methods tested on the error backpropagation algorithm built in *Rapid Miner*.

Pre-processing method	Mean F-score
TRR	0,640
CW	0,630
RF	0,576
NTW	0,573
GR	0,558
TR2	0,476
IDF	0,460

4.2 GA for Weight Optimization

GA can be used for the ANN weight optimization as well. At small dimensionalities, it performs better than the error backpropagation algorithm. The dimensionality of the ANN weight optimization task is calculated as the following:

$$D = \sum_{i=2}^{m} n_{i-1} n_i,$$

where m is the number of layers (including the input layer) and n_i is the number of neurons in the i^{th} layer.

In this way, dimensionality of the default ANN structure in *RapidMiner* for the natural language call routing problem is equal to 840. Therefore, GA-based weight optimization requires much more computational resources than the error backpropagation.

We used 300 generations and a population size equal to 300 for the GA-based weight optimization application. The mean *F*-score with TRR was equal to 0.585; it is significantly worse than the result with the error backpropagation algorithm. Furthermore, the computational time for the GA application equals approximately 4 hours; the error backpropagation algorithm requires approximately 5 minutes with the same computer.

Therefore, GA-based weight optimization is not appropriate in comparison with the error backpropagation algorithm.

4.3 Baseline ANN Structure Optimization with GA

In the definition of GA, each individual contains all the information about the ANN structure (the number of neurons in each hidden layer, the kinds of their activation functions). It is critical for the classical GA to represent solutions as binary strings of a fixed length. We consider the Ward ANN, where neurons in each layer are divided into blocks having the same activation function. This approach uses the following solution encoding: there is some superstructure, which can represent any ANN of a smaller size. A potential solution is located in the space of multilayer feedforward ANNs where any two neurons from neighbouring layers surely have a connection. In order to define this structure, it is necessary to set the sizes of blocks s_1 and s_2 for hidden and output layers accordingly, the max number of neurons in a hidden layer a_1, the number of neurons in the output layer a_2, the max number of hidden layers n_1 and the number of bits k required for coding an activation function kind. Each neuron of a hidden layer occupies $1+k$ bits (the first bit is equal to 1 if the neuron and the corresponding connections with other neurons exist, otherwise it is equal to 0). In that way, the general length of the binary string is calculated as follows:

$$l = n_1 \frac{a_1}{s_1}(k+1) + \frac{a_2}{s_2}k,$$

s_1 and s_2 are to be aliquot parts of a_1 and a_2 accordingly. The approach proposed in (Bukhtoyarov, 2010) implements a special case of this encoding when $s_1=s_2=1$.

The following activation functions are used:

- Linear function:

$$f_0(x) = \frac{a}{10}x + 0.5,$$

- Sigmoidal function:

$$f_1(x) = \frac{1}{1+e^{-ax}},$$

- Hyperbolic tangent:

$$f_2(x) = \frac{\tanh(ax)}{2} + 0.5,$$

- Rational sigmoidal function:

$$f_3(x) = \frac{x}{2(|x| + 1/a)} + 0.5,$$

where a is a parameter. In this encoding, it is constant and equal to 1.

As a weight optimization algorithm for the fitness function calculation we use error backpropagation. Due to the features of the algorithm, each activation function is required to have a continuous derivative. The parameters of the algorithm (such as learning rate, momentum and so on) are the same as in the *RapidMiner* implementation. A single fitness function calculation does not provide an appropriate estimation of the individual fitness; error backpropagation is a local search algorithm, starting weights are generated randomly, and the result varies a lot. In order to get a more reasonable individual fitness estimation it was proposed to implement several training attempts (we used 5 attempts) and then the mean *F*-score was set as the fitness function value.

However, this encoding has some essential disadvantages. The first one is transposition sensitivity. It means that there can be two different binary strings (genotypes) representing the same ANN structure (phenotype). This effect increases the search space and slows the algorithm down. Another disadvantage is a risk of so-called destructive recombination. There are different entities in the encoding (neurons belonging to different layers), and recombination between them does not make any sense. It is possible to avoid this risk using the uniform recombination operator only.

The following genetic operators are used: rank selection, uniform recombination and average mutation.

The numerical experiment in *RapidMiner* showed that it was enough having the ANN with one hidden layer of at least 21 neurons to solve the

classification problem. The classification efficiency was not improved when the ANN contained more than one hidden layer: the ANN remembered the training examples more successfully but lost the ability to generalize them. Therefore, it was decided to set the max number of hidden layers equal to 1 and the max number of neurons in a hidden layer equal to 32 with a reserve on the safe case.

We obtained some optimization tasks of different dimensionality with this encoding. The dimensionality depends on the sizes of blocks in the hidden and output layers. These sizes were chosen as the following: all aliquot parts of 32 and 20 were sorted in ascending order and then were grouped in pairs. There were 6 different pairs in total.

We used resource proportional to the length of a binary string in each case. The max length of the binary string with the blocks (1,1) is equal to 136. In this case we used 110 generations and a population size equal to 110. The min length of the binary string with the blocks (32,20) is equal to 5. In this case we used exhaustive search instead of GA due to the small volume of the search space.

The solutions obtained include one hidden layer with 30 neurons (for the blocks pairs containing the first block size which is an aliquot part of 30) or 32 neurons (for other blocks pairs) in it. A *t*-test with the confidential probability 0.95 was performed for all different pairs of the solutions. Generally, the classification effectiveness depends on the sizes of blocks: the ANN with the blocks (2,2) works significantly worse than the others (*F*-score = 0.670), the ANN with the blocks (8,5) works significantly better than the others (*F*-score = 0.684), and there is no significant difference between the ANNs with the blocks (1,1), (4,4), (16,10) and (32,20). All these implementations of the ANN structural optimization work significantly better than the error backpropagation algorithm for the fixed ANN structure. However, they are more time-consuming; the simplest version requires approximately 10 minutes, and the most complicated one - 11 hours.

4.4 The Novel Approach to ANN Structure Optimization with GA

We found that the baseline encoding was excessive and proposed a novel approach to the ANN structure representation. In fact, the novel encoding is a simplified version of the original one. It deals with whole ANN layers, not with separate blocks of neurons; all the neurons in one layer have the same activation function. However, a new tuning tool was added; all the activation functions were parametrized with the parameter *a*. It is possible to get different forms of each activation function changing this parameter.

In that way, it is necessary to set the max number of hidden layers, the max number of neurons in each hidden layer, the number of activation functions and to discretize the parameter *a*. The required number of bits for each entity can be found after all. The general length of the binary string is calculated as follows:

$$l = n_1(m + k + k_a) + k + k_a,$$

where n_1 is the max number of hidden layers, m is the number of bits for coding the number of neurons in each hidden layer, k is the number of bits for coding the activation function kind and k_a is the number of bits for coding the parameter a. We obtained a less flexible model than the original one, however, it has reasonable dimensionality, fewer parameters for tuning, and there is no transposition sensitivity. It requires much less resource of the GA and works much faster. In our case, the length of the binary string equals 19; it is 7 times smaller than the max binary string length for the baseline approach with the blocks (1,1). Moreover, the difference between the dimensionalities increases exponentially when we increase the max structure of an ANN.

Other options of GA such as a fitness function calculation method and genetic operators remain the same as in the baseline approach.

We obtained a GA implementation with effective convergence and reasonable resource consumption (a population size was 50 individuals, the generation number was 50; the resource is 5 times less than for the baseline approach with the blocks (1,1)).

The results obtained show that the novel GA of the ANN structural optimization performs well at the call routing task; the mean *F*-score with TRR is equal to 0.684; according to the *t*-test with the confidential probability 0.95, it is significantly better than the *RapidMiner* implementation of a simple ANN trained with the error backpropagation algorithm. Moreover, it is also significantly better than the baseline GA-based ANN structural optimization. There is the only case when the novel and the baseline encodings have no significant difference: when the baseline approach uses the blocks (8,5). Generally, the structure of the solution obtained (1 hidden layer with 30 neurons) remains the same as with the baseline approach. At the same time, the novel approach requires much less resource and computational time than the baseline one in

order to get a better result.

The comparison of different ANN structure optimization algorithms that use the error backpropagation weight optimization is illustrated in Table 2.

Table 2: Results of different ANN structure optimization algorithms.

Algorithm	Binary string length	Resource	Mean F-score
Fixed structure ANN	-	-	0,640
Ward ANN (1,1)	136	12100	0,673
Ward ANN (2,2)	68	8100	0,670
Ward ANN (4,4)	34	4900	0,678
Ward ANN (8,5)	20	2500	0,684
Ward ANN (16,10)	10	exhaustive search	0,675
Ward ANN (32,20)	5	exhaustive search	0,675
Novel encoding ANN	19	2500	0,684

5 CONCLUSIONS

The text classification problem of natural language call routing was considered. The most effective term weighting method (TRR) was determined.

Artificial neural networks (ANN) were applied for classification. The numerical experiments have shown that the error backpropagation algorithm is more effective than the GA for the ANN weight optimization: the mean *F*-score values are 0.640 and 0.585 accordingly. Furthermore, GA-based ANN weight optimization is much more time-consuming.

The structural optimization is important and can significantly improve the ANN learning effectiveness. The baseline GA-based structural optimization provides the mean *F*-score from 0.670 to 0.684 depending on the sizes of blocks in the Ward ANN.

A novel approach of GA-based ANN structure optimization was proposed. The solution encoding of the novel approach has an effective balance between the ANN structure representation flexibility and the problem dimensionality, also it has fewer parameters for tuning. Due to this encoding, the GA provides a significantly better improvement of the classification result (the mean *F*-score is 0.684) than the baseline method, excluding the only implementation when the Ward ANN in the baseline approach contains blocks (8,5). Only in this case there is no significant difference between the encodings. However, if sizes of blocks in the Ward ANN have been chosen wrong way, the result is significantly worse. Moreover, the novel approach does not complicate the solution structure and requires less computational resource than the baseline approach. The problem dimensionality is 7 times lower than for the most complicated implementation of the baseline approach. The difference between the dimensionalities of the baseline approach and the novel one increases exponentially with increasing the max structure of an ANN, that gives the possibility for a more effective way of problem solving.

REFERENCES

Bukhtoyarov, V. and Semenkina, O. 2010. Comprehensive evolutionary approach for neural network ensemble automatic design. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE.

Bukhtoyarov, V., Semenkin E. and Sergienko, R. 2011. Evolutionary Approach for Automatic Design of Neural Networks Ensembles for Modeling and Time Series Forecasting. *Proceedings of the IADIS International Conference Intelligent Systems and Agents*: 93-96.

Cohen, W. 1995. Fast Effective Rule Induction. *Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, California*.

Debole, F. and Sebastiani, F. 2004. Supervised term weighting for automated text categorization. *Text mining and its applications*: 81-97. Springer Berlin Heidelberg.

Gasanova, T., Sergienko, R., Semenkin, E., and Minker, W. 2014. Dimension Reduction with Coevolutionary Genetic Algorithm for Text Classification. *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO),* Vol.1:215–222.

Hecht-Nielsen, R. 1989. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*:593-605. IEEE.

Ko, Y. 2012. A study of term weighting schemes using class information for text classification. *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*: 1029-1030. ACM.

Lan, M., Tan, C. L., Su, J., and Lu, Y. 2009. Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721-735.

Salton, G. and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513-523.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*,

34(1):1-47.

Shafait F., Reif M., Kofler C., and Breuel T. M. 2010. Pattern Recognition Engineering. *RapidMiner Community Meeting and Conference*, 9.

Soucy P. and Mineau G. W. 2005. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*:1130-1135.

Suhm B., Bers J., McCarthy D., Freeman B., Getty D., Godfrey K., and Peterson P.. 2002. A comparative study of speech in the call center: Natural language call routing vs. touch-tone menus. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*: 283–290. ACM.

WARD SYSTEMS GROUP, INC. (1996) NeuroShell 2 User's Manual.

Whitley, D., Starkweather, T., and Bogart, C. 1990. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347-361.

Xu, H. and Li, C. 2007. A Novel term weighting scheme for automated text Categorization. *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*: 759-764. IEEE.

Yang, Y., and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. *ICML*, vol. 9:412-420.