Hierarchical Semantic Perceptron Grid based Neural Network

CAO Huai-hu, YU Zhen-wei, WANG Yin-yan
(Dept. Computer of China University of Mining and Technology Beijing, Beijing 100083, china)
chhu@cumtb.edu.cn

Abstract A hierarchical semantic perceptron grid architecture based neural network has been proposed in this paper, the semantic spotting is a key issue of this architecture, for finding solution to this problem, firstly we has formulated it, then a Semantic neural network classifier frame has been proposed. And experiment results of this scenario were presented, to evaluate the effectiveness of this scenario, we compare this scenario with the SVM, NNet, KNN and NB, the average experimental results of the scenario are obviously superior to other conventional approaches.
Key words: semantic perceptron, grid, neural network, semantic classifier, machine learning

1. Introduction

Knowledge and Semantic Web technologies are evolving the Grid towards the Semantic Grid to facilitate knowledge reuse and collaboration within a community of practice. The Semantic grid is about the way that knowledge is acquired, used, retrieved, published and maintained to assist e-Scientists to achieve their particular goals and objectives -knowledge is understood as information applied to achieve a goal, solve a problem or enact a decision. It involves all three conceptual layers of the Grid: knowledge, information and computation/data.[1,3]

For building a semantic perceptron grid, above all, semantic spotting must be exactly achieved to form basic semantic grid nodes. In this paper, we propose a hierarchical semantic perceptron grid architecture based neural network(SPGN). It uses context at the lower layer to select the exact meaning of key words, and employs a combination of context, co-occurrence statistics and thesaurus to group the distributed but semantically related words within a semantic to form basic semantic nodes. The semantic nodes are then used to infer the semantic within an input document. For finding solution to semantic spotting problem, we has formulated it, a semantic neural network classifier(SNNC) frame has been proposed. Experiments on 20000 data set demonstrate that the SNNC is able to capture the semantics, and it performs well on semantic spotting task.

2. Problem description and theoretical framework

Semantic spotting is the problem of identifying the presence of a predefined Semantic in a text document. More formally, given a set of n Semantics together with a collection of documents, the task is to determine for each document the probability that one or more semantics is present in the document. Semantic spotting may be used to automatically assign subject codes to newswire stories, filter electronic emails and online news, and pre-screen document in information retrieval and information extraction applications.

Semantic spotting, and its related problem of text categorization, has been a hot area of research for over a decade. A large number of techniques have been proposed to tackle the problem, including: regression model, nearest neighbor classification, Bayesian probabilistic model, decision tree, inductive rule learning, on-line learning, and, support vector machine (Yang & Liu, 1999; Tzeras & Hartmann, 1993)[6]. Most of these methods are word-based and consider only the relationships between the features and semantics, but not the relationships among features. Semantic spotting theoretical framework model is shown in Fig.1. The model has two basic components: gating networks and expert networks. The gating networks, located at the intermediate level nodes of the tree, receive the input $\mathbf{x}^{(m)}$ (a vector $\mathbf{x}$ of m input features representing a document) and produce scalar output that weights the contribution of the child networks. The expert networks, located at the leaf nodes, receive the input $\mathbf{x}^{(m)}$ to produce an estimate of the output. This model may be seen as a cascade of networks that works in a "bottom-up" fashion: the input is first presented to the experts that generate an output, then the output of the experts are combined by the second level gates, generating a new output. Finally the outputs of the second level gates are combined by the root gate to produce the appropriate result $\mathbf{y}^{(n)}$ (a vector $\mathbf{y}$ of n components where n is the number of outputs). The ∑nodes in the tree represent the convex sum of the output of the child nodes which is computed as $\mathbf{y}^{(n)} = \sum_j g_j y_j^{(n)}$ where g_j is the output of the gate and $y_j^{(n)}$ is the output of the corresponding child node.

In the original model proposed by Jordan and Jacobs all the networks in the tree are linear (perceptrons). The expert network produces its output $\mathbf{y}$ as a generalized linear function of the input $\mathbf{x}$:

$$\mathbf{y} = f(U_{\mathbf{x}}) \qquad (1)$$

where U is a weight matrix and f is a fixed continuous non linear function. The vector x is assumed to include a fixed component with value one to allow for an intercept term. For binary classification problems, f (·) is the logistic function, in which case the expert outputs are interpreted as the log odds of "success"

under a Bernoulli probability model. Other models (e.g., multi-way classification, counting, rate estimation and survival estimation) are handled by making other choices for f (·). The gating networks are also generalized linear functions. The ith output of the gating network is the "softmax" function of the intermediate variable ψ_i (Bridle 1989, McCullagh and Nelder 1989) [7].

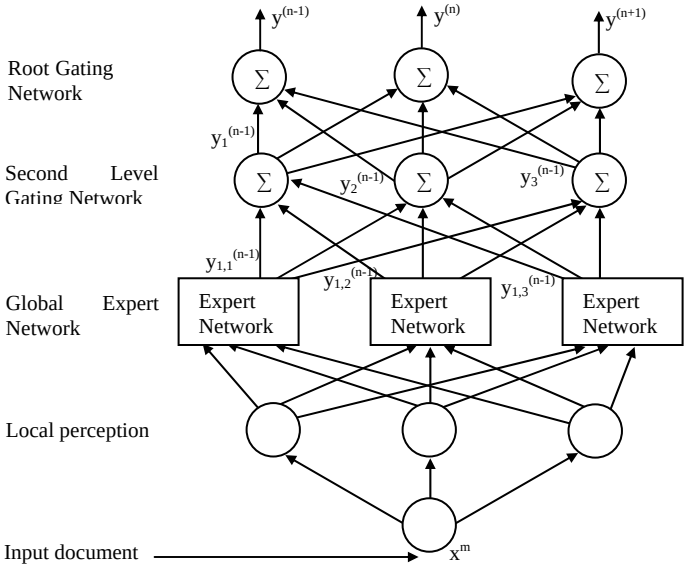

Fig.1. Semantic spotting theoretical framework model

$$g_i = \frac{e^{\Psi_i}}{\sum_{j=1,\ldots,l} e^{\Psi_i}} \qquad (2)$$

Where l is the number of child nodes of the gating network, and the intermediate variable ψ_i is defined as:

$$\Psi_i = V_i^T X \qquad (3)$$

Where v_i is a weight vector and ^{T} is the transpose operation. The g_is are positive and sum to one for each x. They can be interpreted as providing a "soft" partitioning of the input space. The output vector at each nonterminal node of the tree is the weighted sum of the output of the children below that nonterminal. For example, the output at the ith nonterminal in the second layer of the two-level tree in fig.1. is:

$$y_i^{(n)} = \sum_j g_{i \cdot j} y_{i.j}^{(n)} \qquad (4)$$

Where j =1, . . . , l, l is the number of child nodes connected to the gate, $y_{i.j}^{(n)}$ is the output of expert j which is a child of gate i, and $g_{i.j}$ is the jth output of the gate i. Note that since both the g's and the y's depend on the input x, the output is a nonlinear function of the input. The theoretical framework model is very flexible. One may choose a function f (Eq.1.) for the gate and expert decision modules that is appropriate for the application. Moreover, since gates and experts depend only upon the input x, one may choose either bottom-up or top-down processing whichever is appropriate for the problem. In our variation of the theoretical framework model we use a binary classification function at the gates. Also, we train and use our model top-down. The choice of this direction was made based upon the number of categories. As will be described in detail later, we have a collection of 216 categories in the dataset and moreover aim towards a scalable categorization procedure that can handle large classification schemes. Abottom-up approach would be very inefficient given that there is an expert module for each category. The computational requirements are especially severe when the decision module at each expert node is a neural network. In contrast our top-down approach along with a binary classification at each gate restricts the number of expert networks to be activated for a given document. It may be observed that in studies using the bottom-up approach the number of categories was small as for example 26 for handwriting recognition and 52 for speech recognition (Waterhouse 1997)[8]. In this study we choose the more efficient direction

and postpone the exploration of bottom-up processing for future research.

In our model, given a binary function a gate is trained to yield a value of 1 if the example document is categorized with any of its descendent concepts. During testing, the categorization task starts at the root node and its gate decides whether the most general concept is present in the document. If this is true, then all the second level nodes make their decisions and the process repeats until it reaches the leaf nodes. Observe that only the experts connected to gates that output the value 1 are activated thus reducing the response time during classification. A key difference depicted is at the gating networks which use binary functions. To give a statistical interpretation of our model we define $Z_1, \ldots, Z_j$ as the path of gates from the root node to the gate j that is the parent of an expert ε_k that assigns category k. Let $x^{(m)}$ be the input features that represent a document, and $y^{(n)}$ the output vector of the categories assigned to the document. The probabilistic interpretation of our hierarchical model is as follows:

$$P(y^{(n)} \mid x^{(m)}) = \sum_{k=1}^{n} P(Z_1 = 1 \mid x^{(m)}) P(Z_2 = 1 \mid Z_1 = 1, x^{(m)}) \ldots$$
$$P(Z_j = 1 \mid Z_1 = 1, \ldots, Z_{j-1} = 1, x^{(m)}) P(\varepsilon_k = 1 \mid x^{(m)}) \quad (5)$$

Given an appropriate set of features and a training set of manually categorized documents, the backpropagation network learns to make the appropriate decisions. Observe that for experts and gates the set of positive examples is different. The set of positive examples for the experts is a subset of the positive examples of any ancestor gate. As a consequence, two identical neural networks trained with these different subsets learn different probabilistic functions. The backpropagation neural network as an expert node learns to use the input to estimate the desired output value (category), while as a gate it computes the confidence value of the combined outputs of its children.

3. Implementation of hierarchical semantic perceptron grid based neural network

It may be observed that network architecture and feature selection methods must be studied in combination. In fact, at a basic level feature selection is one of the factors that define the configuration of the network. Given the many complex combinations in terms of feature selection methods and numbers of nodes in the different layers for both the expert and gating networks we approach the problem in stages[9]. We follow a top-down approach that optimizes the gates and then the experts. First we focused our attention on the gating networks. We used experts with 25 input features and 50 nodes in the hidden layers. We then explored 5, 10, 25, 50, 100, and 150 input features for the gating networks with hidden layer that had twice the number of input nodes. We also tried all three different feature selection methods (Mutual Information, Odds Ratio and Correlation Coefficient). This experiment was done only on the “high frequency categories” because they allow appropriate training of the neural networks and also because the variance between different training runs is smaller than the variance for lower frequency categories. Interestingly the differences between the three feature selection methods on the gating networks are not significant. It is possible for this slight decrease to be caused by the limit in the number of iterations (1, 000) that a network was allowed to run during training. Usually a larger network needs more iterations on the training set to converge to an optimal value.

3.1. Semantic neural network classifier frame

In order to assess the advantage gained by exploiting the hierarchical structure of the classification scheme, we built a flat neural network classifier. We decided to build a flat modular classifier that is implemented as a set of 103 individual expert networks. In this model the experts are trained independently using the optimal feature set and the category zone for each individual category [2]. The thresholding step is performed by optimizing the F1 value of each expert using the entire training set. These are the values that we report in the next section for the flat neural network classifier. Observe also that the use of this model allows us to assess the contribution of adding the hierarchical structure, i.e., our first research question. Neural networks can learn nonlinear mappings from a set of training patterns. The parameters are adjusted to the training data in a number of iterations. We have used a back-propagation algorithm that minimizes quadratic error. Fig.2. gives a pictorial representation of a neural network for document classification. The input layer has as many units (neurons) as features retained. The number of hidden units is determined by optimizing the performance on a cross-validation set. There is one output unit for each possible class. Hertz [11] and Ripley [12] give an introduction to neural networks that goes out of the scope

of this paper. This learning process needs to be monitored using a cross-validation set to avoid overfitting the data. We have used several cross-validation sets as described later.

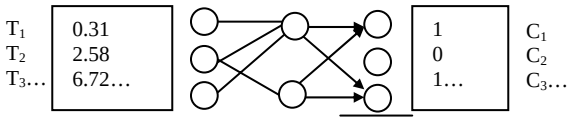

Fig.2. Neural Network for semantic classification

3.2 Feature selection

In text categorization the set of possible input features consists of all the different words that appear in a collection of documents. This is usually a large set since even small text collections could have hundreds of thousands of features. Reduction of the set of features to train the neural networks is necessary because the performance of the network and the cost of classification are sensitive to the size and quality of the input features used to train the network (Yang and Honovar 1998) [4,5]. A first step towards reducing the size of the feature set is the elimination of stop words, and the use of stemming algorithms. Even after that is done the set of features is typically too large to be useful for training a neural network.

Two broad approaches for feature selection have been presented in the literature: the wrapper approach, and the filter approach (John et al. 1994) [10]. The wrapper approach attempts to identify the best feature subset to use with a particular algorithm. For example, for a neural network the wrapper approach selects an initial subset and measures the performance of the network; then it generates an "improved set of features" and measures the performance of the network. This process is repeated until it reaches a termination condition (either a minimal value of performance or a number of iterations). The filter approach, which is more commonly used in text categorization, attempts to assess the merits of the feature set from the data alone. The filtering approach selects a set of features using a preprocessing step, based on the training data. In this paper we use the filter approach, but we plan to explore the wrapper approach in future research. We select three methods that have been used in previous works: correlation coefficient, mutual information, and odds ratio. We eliminate those stems that occur in less than 5 documents in the training collection. Since feature selection is done for each category, based on its zone (explained later) we also remove stems that occur in less than 5% of the positive example documents. We then rank the remaining stems by the feature selection measure and select a pre-defined number of top ranked stems as the feature set.

3.3 Training set selection

A supervised learning algorithm requires the use of a training set in which each element has already been correctly categorized. One would expect that the availability of a large training set (such as OHSUMED) will be beneficial for training the algorithm. In practice this does not seem to be the case. The problem occurs when there is also a large collection of categories with each assigned to a relatively small number of documents. This then creates a situation in which each category has a small number of positive examples and an overwhelming number of negative examples. When a machine learning algorithm is trained to learn the assignment function with such an unbalanced training set, the algorithm will learn that the best decision is to not assign the category. The overwhelming amount of negative examples hides the assignment function. To overcome this problem an appropriate set of training examples must be selected. We call this training subset the "category zone". The method for creating the category zone is a KNN approach in which the category zone consists of the set of K nearest neighbors for each positive example of the category. This method will produce variable sized category zones. We explored several values of K (10, 50, 100 and 200). Our main concern with this method was to obtain a training set large enough to train a neural network without overfitting.

4. Performance measures

Table1 Contingency table for class j

	Correct YES	NO
Assigned YES	a_j	b_j
NO	c_j	d_j

Table 1 describes the possible outcomes of a binary classifier. The system YES/NO results refer to the classifier output and the real YES/NO refers to what the correct output is. The perfect classifier would have a value of 1 for a_j jand d_j, and 0 for b_j and c_j. Using table 1 we define 3 performance measures common in the semantic classification literature.

The tradeoff between recall and precision is controlled by setting the classifiers parameters. To describe the performance both values should be provided. Another common performance measure is the F-measure defined by Rijsbergen [12]

$$F_\beta(r,p) = \frac{(\beta^2+1)pr}{\beta^2 p + r}$$

The most commonly used F-measure in text classification is F_1:

$$F_1(r,p) = \frac{2pr}{p+r}$$

When dealing with multiple classes there are two possible ways of averaging these measures, namely, macro average and micro average. In the macro averaging, one contingency table as Table 3 per class is used, the performance measures are computed on each of them and then averaged. In micro averaging only one contingency table is used, an average of all the classes is computed for each cell and the performance measures are obtained therein. The macro average weights equally all the classes, regardless of how many documents belong to it. The micro average weights equally all the documents, thus favoring the performance on common classes. Different classifiers will perform different in common and rare categories. Learning algorithms are trained more often on more populated classes thus risking local overfitting.

5 Experiments and Results

We summarize here the steps followed in the preparation of the experiments, and the results therein.

a. Preparing the data: A list of 571 stop-words were removed from the document collection. A standard stemming algorithm was also used.

b. Vectorization and weighting: The resulting documents were represented as vectors, using TF×IDF weighting. Other weighting schemes were tried and this was found optimum. The Smart text processing package [9] was used for this stage. All the experiments described use a cosine normalization.

c. Network architecture: The selected terms were used as input features to the network. We tried networks with 500, 1000 and 2000 input features. The network has 90 output units, one for each class.

d. Training: We generated 5 cross-validation sets with a number of random documents each. These documents were set aside and the network was trained on the remaining ones. As a rule of thumb, it is common to use 5-10% of the training set, we tried using 500 and 1000 documents, the smaller cross-validation set resulted in better performance. The learning rate (η) was reduced when the error found a minimum on the cross-validation set. Reducing η during learning is a commonly used technique, similar to "cooling down" in simulated annealing. In table 2 we show the micro and macro average F_1 performance for networks with 500, 1000 and 2000 input features. In all cases the best results obtained were for 100 hidden units. In table 3 the results obtained by other authors and by us are displayed. We show in this table the best results for each input dimensionality.

The results in table 2 and 3 were obtained using $\chi^2_{\max\times P_r}(t)$. We compare the performance of the classifier with the results of other authors that used Support Vector Machines (SVM), kth Nearest Neighbors (KNN) and Naïve Bayes (NB) classifiers. The micro averaged precision of neural networks is the best for all the methods compared, this is particularly important for many applications where precision is the most important variable. The F_1performance for neural networks, in particular M_aF_1, is not as good as the one for SVM or KNN, and this is due to a lower recall level. The tradeoff between high recall or high precision is a well known problem and must be considered for each application. Neural networks performance has a degree of randomness, inherent to the learning process and by distinct initial conditions, so it is not possible to determine the method performance by a single experiment. Several trainings are needed, preferably with different cross-validation sets and different initial weights. We have used 5 to 20 independent experiments to determine each value.

Table 2. Performance of the network with 500 and 1000 input features.

	50 hidden	100 hidden	150hidden
Macro F_1 500inputs	0.254+/-0.021	0.275+/-0.029	0.278+/-0.027
Micro F_1 500inputs	0.834+/-0.005	0.821+/-0.005	0.832+/-0.004
Macro F_1 1000inputs	0.293+/-0.024	0.320+/-0.010	0.305+/-0.044
Micro F_1 1000inputs	0.845+/-0.003	0.852+/-0.002	0.850+/-0008
Macro F_1 2000inputs	0.283+/-0.018	0.284+/-0.021	0.256+/-0.031
Micro F_1 2000inputs	0.839+/-0.003	0.844+/-0.002	0.842+/-0.005

Table 3 Performance of different classifiers

	miR	miP	miF_1	maF_1
SVM	0.912	0.811	0.861	0.512
NNet	0.780	0.904	0.838	0.290
KNN	0.835	0.882	0.856	0.525
NB	0.768	0.824	0.794	0.387
NN500inp.	0.787+/-0.010	0.910+/-0.002	0.844+/-0.004	0.281+/0.027
NN1000inp.	0.800+/-0.015	0.907+/-0.001	0.854+/-0.002	0.325+/-0.013
NN2000inp.	0.783+/-0.006	0.914+/-0.001	0.847+/-0.003	0.282+/-0.018

6. Conclusion

In this paper, we have proposed a hierarchical semantic perceptron grid architecture based neural network, we has formulated the semantic spotting problem, and we has proposed a semantic neural network classifier frame. The experiment results of this scenario were presented, to evaluate the effectiveness of this scenario, we compare this scenario with the SVM, NNet, KNN and NB, the average experimental results of the scenario are obviously superior to other conventional approaches. Future work includes trying these methods on new data sets, we are particularly interested in the classification of documents in other languages, other neural network schemes should also be tested.

REFERENCES

1. Foster, I., C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001. 15(3): p. 200-222.
2. SLONIM, N. AND TISHBY, N. 2001. The power of word clusters for text classification. In Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research (Darmstadt, Germany, 2001).
3. S. Lawrence and G. Giles. Accessibility and distribution of information on the web. Nature, 400:107–109, 1999.
4. Y. Yang. An evaluation of statistical approaches to text categorization. Information Retrieval Journal, May, 1999.
5. Y. Yang and X. Liu. A re-examination of text categorization methods. In ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), 1999.
6. Alani, H., Dasmahapatra, S., Gibbins, N., Glaser, H., Harris, S., Kalfoglou, Y., O'Hara, K., and Shadbolt, N. Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web. 14th International Conference on Knowledge Engineering and Knowledge Management, Spain, October, 2002.
7. Bontcheva, K. Tailoring the Content of Dynamically Generated Explanations. M. Bauer, P.J. Gmytrasiewicz, J. Vassileva (eds). User Modelling 2001: 8th International Conference, UM2001, Lecture Notes in Artificial Intelligence 2109, Springer Verlag, 2001.
8. Cerf, V. G., et al., "National Collaboratories: Applying Information Technologies for Scientific Research", National Academy Press: Washington, D.C., 1993.
9. Ciravegna, F.: "Adaptive Information Extraction from Text by Rule Induction and Generalisation" in Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001), Seattle, August 2001."
10. Andreyev Y V , Belsky YL , Dmittriev A S , et al. :neural networks implementation[J] Information processing using dynamical chaos. IEEE Trans Neural Networks , 7.(1996):2902299.
11. J. Hertz, A. Krogh, and R. Palmer. Introduction to the theory of neural computation. Addison-Wesley, Redwood, CA, 1991.
12. B.D. Ripley. Pattern recognition and neural networks. Cambridge Univeristy Press, Cambridge, 1996.