Using SVM and Error-correcting Codes for Multiclass Dialog Act Classification in Meeting Corpus

Yang Liu

The University of Texas at Dallas, Richardson, TX, USA
yangl@hlt.utdallas.edu

Abstract

Accurate classification of dialog acts (DAs) is important for many spoken language applications. Different methods have been proposed such as hidden Markov models (HMM), maximum entropy (Maxent), graphical models, and support vector machines (SVMs). In this paper, we investigate using SVMs for multiclass DA classification in the ICSI meeting corpus. We evaluate (1) representing DA tagging directly as a multiclass task, and (2) combining multiple binary classifiers via error correction output codes (ECOC). For the ECOC combination, different code matrices are utilized (e.g., the identity matrix, exhaustive code, BCH code, and random code matrix). We also compare using SVMs with our previous Maxent model. We find that for DA tagging, using multiple binary SVMs via ECOC outperforms a direct multiclass SVM, but neither achieves better performance than the Maxent model, possibly because of the small class set and the features currently used in the task.

Index Terms: dialog act classification, SVM, maximum entropy, error correction output code.

1. Introduction

Dialog act (DA) represents certain kind of discourse structure in conversations (either human-to-human or human-to-computer). In spoken dialog systems, automatic identification of the utterance type (i.e., DA) is an important task. With the growing interest in processing multiparty meeting corpus, accurate classification of DAs will play a vital role for automatically understanding and summarizing meetings.

Several machine learning techniques have been investigated for effective DA classification [1, 2, 3, 4, 5, 6]. Because of the success of the SVM in many applications, in this work we employ SVM for the DA classification task. Since SVM is mostly used in binary classification tasks, we investigate using error-correcting output codes (ECOC) for combining multiple binary SVMs, in addition to the direct multiclass SVMs. We also compare SVMs to the maximum entropy (Maxent) model used in our previous work for this task. Note that in this paper all the approaches use supervised learning, i.e., we used the training data that is labeled with DA tags. In [7], we have investigated the effect of active learning and weakly supervised learning using the HMM and Maxent for the task of DA classification.

The rest of the paper is organized as follows. Section 2 describes the related work and the techniques we investigate in this paper, including the SVM and the ECOC. Section 3 presents the experimental results for DA classification using various approaches. Discussion and future work appear in Section 4.

2. Methods

2.1. Previous Work

Many studies have been conducted for DA classification in order to process either human-human or human-computer dialogs [1, 2, 3, 4, 5, 6], using different machine learning techniques, such as Maxent, DBN, HMM, and SVM. These methods adopted different representations for the DA tagging task. For example, given a vector of features associated with a DA unit, the Maxent classifier directly estimates the conditional probability of the DA tag for each unit [1]. In HMM [3], the "hidden" states are the DA tags, which generate the sequences of words as observations. A DA grammar N-gram LM provides the transition probability between the DA tags, and a DA specific N-gram word-based LM provides the observation probability. Different knowledge sources have been explored in the previous work for DA classification, including both prosodic features and textual cues.

2.2. Support Vector Machines (SVM)

SVMs have achieved comparable or significantly better performance in many classification tasks. It was originally designed for a binary classification problem, in which a hyperplane is determined with the maximum margin to the 'support vectors'. For a multiclass tagging task,[1] many methods based on SVMs attempt to build multiple binary SVM classifiers (e.g., one versus else or a pair wise scheme), and then combine them to obtain the final class hypothesis for a test sample. The multiple classifier combination problem can also be solved in a general framework using ECOC, with more description in the next section.

Using multiple binary classifiers generally requires a number of classifiers, leading to more computational complexity. Therefore, learning a discriminative function that distinguishes multiple labels at the same time can eliminate the need of building many classifiers, and can potentially utilize the dependencies among different labels as well. Learning such a function can be implemented as maximizing the margin between the best hypothesis and the others. See [8, 9] for more details on the optimization algorithms and the implementation for the multiclass SVMs.

2.3. Combining Multiple Binary Classifiers via ECOC

The idea of using error correcting output codes (ECOC) [10] is to avoid the direct multiclass problem by breaking the multiclass task into several binary classification tasks and then combining the results from these indirect classifiers. ECOC has been used in several language processing tasks on top of different base classifiers, for example, naive Bayes and conditional random fields [11, 12].

[1] This means that there are more than 2 class tags in the data set; however, each data point only has one class associate with it.

The following matrix shows an example code matrix for a task with 4 classes (C_i), using 5 base classifiers (S_i).

	C_1	C_2	C_3	C_4
S_1	1	0	0	0
S_2	0	0	0	1
S_3	1	0	1	0
S_4	0	1	1	0
S_5	0	1	1	1

In the code matrix above, each class C_i is associated with a codeword (i.e., the column vector). Each classifier S_i is trained to perform a binary classification task, that is, to distinguish the two subsets of the classes labeled with 1 and 0, respectively.[2] During testing, a vector of scores $[o_1, o_2, ..., o_5]$ is generated by the 5 binary classifiers for each test sample. This vector is then compared to each codeword, and the one with the minimum distance is chosen as the hypothesis.

There are two problems in this framework, the design of the code matrix and the distance measure. To design a good code matrix, a general idea is to have large row and column separation. The columns in the code matrix[3] are the codewords (i.e., corresponding to different classes), hence the larger the distance among them, the more likely that a correct hypothesis is obtained even with errors from some classifiers during testing. In coding theory, the number of errors that a code is guaranteed to be able to correct is

$$\lfloor \frac{H_c - 1}{2} \rfloor$$

where H_c is the minimum Hamming distance between any pair of the codewords. The row vectors in the matrix represent different classifiers. In order to achieve an effective ensemble of multiple classifiers, it is better to have classifiers that have low correlations and make different errors, in the hope that they will combine effectively. However, there is no well-defined measurement for the diversity of different classifiers in an ensemble approach, especially when these classifiers conduct different classification tasks.

Two methods can be used for the distance measure. One uses the hard decision generated by each classifier (composed of 1 and 0, or -1 depending on the representations of the code matrix) and compares it to the template codeword using the Hamming distance. The other method preserves the soft decision from the binary classifiers. For example, in a 0/1 labeling task, if the classifiers generate the posterior probability or confidence for a test sample, the distance to the codeword C_i is:

$$d(i) = \sum_{k=1}^{N} |p(k,i) - M(k,i)| \quad (1)$$

where N is the total number of classifiers, $p(k,i)$ is the posterior probability generated by the classifier S_k corresponding to the class C_i (or more correctly, the class group which C_i belongs to), and $M(k,i)$ is the bit value in the code matrix.

3. Experiments

3.1. Experimental Setup

3.1.1. Data

We used the ICSI Meeting corpus that consists of 75 naturally-occurring meetings. Each is roughly an hour in length and has about 5 participants in average. DA units were manually labeled for this corpus using a fine grained set of tags [13]. In this paper, similar to our previous work [1], DA tags are grouped into 5 broad intuitive classes along the lines of [14] – Backchannels (B), Disruptions (D), Fillers (F), Questions (Q) and Statements (S), with the prior distribution of 13.3%, 14.1%, 7.2%, 6.4% and 59.0% respectively. We randomly split the entire corpus and used 55 meetings for training, 10 for development, and 10 for evaluation. Table 1 describes the data used in this experiment. Human transcriptions are used in this study in order to factor out the effect of speech recognition errors and focus on the machine learning aspect of the problem. Also we assume the word sequences are segmented into DAs and the task is to determine the type for each DA. In a real scenario of automatic processing of the meeting data, there is a negative impact from the speech recognition errors and the imperfect DA segmentation as well [1].

	training	test
number of meeting	55	10
number of DAs	82k	12k
number of words	606k	95k

Table 1: Data description.

3.1.2. Features for DA Classification

In the Maxent classifier [7], we used the features including those extracted from the current DA unit: the length, the identity of the first two words, the identity of the last two words, and a bigram of the first two words; the identity of the first word of the next DA unit, and a flag indicating whether or not there is a speaker turn change when this utterance starts. In our previous work, DA contextual information did not yield much gain, therefore for this study we choose to treat this task as a simple local classification task, rather than a sequence decoding problem, which relies to a great deal on the contextual tag information. Note that we do make use of contextual information extracted from the neighboring utterances (not their DA types).

In addition to these features, we investigated some features derived from the DA specific word N-gram LMs that are trained from the utterances with a particular DA. Given a test utterance, we calculate the perplexity (PP) using each of the DA LMs. Then based on the five PPs, we generate the following features: the best and the second best DA hypotheses, and the ordering of the five LM PPs. A jack-knife (or round-robin) paradigm is used to generate these features for the training set. The training set was split into n subsets, and for each subset, the PPs were generated using the LM trained from the other subsets. The PPs for the test set were calculated using the LM that was trained from the entire training set. In this study we do not use any prosodic features, which have been shown to slightly improve DA classification performance [1]. We plan to incorporate those features in our feature work.

3.1.3. Code Matrix

It is not straightforward how to design a code matrix to have good separation in both rows and columns. For classifier combination, the performance is also related to the classifiers and their performance, and the properties of the classes and their grouping. In our experiments, we evaluated four different code matrices, described below.

- Identity matrix.

  This is equivalent to using the one versus else framework

[2]Note that we use 1 and 0 in this code matrix. Typically -1 is used in stead of 0 in SVM classifiers.

[3]Note that in the literature, sometimes the code matrix is represented as the transpose of the matrix as shown in our example, therefore the row and columns are switched.

used in many multiclass tasks. In this approach, n (equal to C, the total number of class labels) binary classifiers are built, each of which learns to distinguish one class versus the others. The code matrix is an identity matrix, that is, 1 for all the diagonal items and 0 elsewhere. Due to space limit, we do not show the matrix here.

- Exhaustive matrix.

  For C classes, all the possible different classifier arrangements are exhaustively used in the code matrix. If C is large, the computational cost is a potential problem. In addition, there may also be redundancy among different classifiers, which affects the diversity and the effectiveness of their combination. For the DA classification task, there are 5 classes, resulting in $2^{C-1} - 1 = 15$ different arrangements in total. Each of the 15 rows in the code matrix corresponds to a classifier S_i. We set 1 for the code word corresponding to class C_1, and the rest 4 bits in the i^{th} row vector are the binary representation of i. For example, for classifier S_6, the row vector is [1 1 0 1 0].

- Random matrix.

  In this method, the code matrix is generated randomly. We used 5 classifiers for this case. The matrix with the largest sum of the row separation and column separation among T generations is selected. The code matrix we used is shown below.

	c_1	c_2	c_3	c_4	c_5
S_1	1	1	1	0	0
S_2	0	0	1	1	0
S_3	1	0	1	1	1
S_4	0	1	1	0	0
S_5	1	0	0	1	0

- BCH code matrix.

  BCH code is one specific type of Reed-Solomon codes in coding theory [15], which guarantees certain correction ability (i.e., via the column separation among the codewords); however, for the application of classifier combination using error-correcting codes, a good row separation is also critical. We selected 7 rows from the BCH (15,5) code as our code matrix, which is shown below.

	c_1	c_2	c_3	c_4	c_5
S_1	0	1	1	1	0
S_2	0	1	1	0	1
S_3	0	0	0	0	1
S_4	1	0	1	0	0
S_5	0	0	0	1	0
S_6	1	1	0	0	0
S_7	1	0	1	1	1

3.1.4. Distance Measure in ECOC

For classifier combination, we use hinge loss for the distance measure between the vector of the SVM scores and the template codewords. Let $v_k = l_k * M_{kj}$, where l_k is the SVM score from the classifier S_k, and M_{kj} is the bit in the code matrix (corresponding to classifier S_k and class C_j), the distance to the k^{th} element of the code word C_j is:

$$d(k,j) = \begin{cases} 1 - v_k & \text{if } v_k < 1 \\ 0 & \text{otherwise} \end{cases}$$

and the total distance $d(j)$ to the code word C_j is $\sum_k d(k,j)$.

Note that the decision for each DA unit is made individually, without considering the tags of the neighboring utterances. Contrary to this, if the ECOC approach is used in a sequence decoding problem (e.g., conditional random fields [12]), the decision for each point cannot be made separately.

3.2. Results

Performance for the DA classification task can be easily evaluated using the classification error rate (CER), i.e., the number of incorrectly classified DAs divided by the total number of DAs.

3.2.1. ECOC Code Matrix

Table 2 shows the results using different code matrices as described earlier, along with the distance for each code matrix. For all the base SVM classifiers, SVM-light [16] was used with the default parameters. Surprisingly the impact of different codes is marginal. The distance shown in the table is the minimum row/column distance. It does not take into account the classifiers' performance (e.g., for which class groups a classifier makes errors), the distribution of the different classes, or the confusion among different class sets. Apparently a large distance does not guarantee an overall good classification performance.

code matrix	CER (%)	row dist.	column dist.
Identity	23.44	2	2
Random	22.91	1	2
BCH	23.78	2	3
Exhaustive	23.16	1	8

Table 2: DA classification error rate using multiple binary SVMs combined via ECOC. The minimum row and column distances are also presented.

3.2.2. Multiclass SVM

Table 3 shows the DA classification results using the direct multiclass framework [9], along with the ECOC result using the identity matrix, which is the widely used method for combining binary classifiers. The multiclass implementation does not perform better than ECOC combination for the DA tagging task. It has been reported that the multiclass approach outperforms the combination of multiple binary classifiers in some applications. We will look into this in the future work and investigate whether some inherent property of a particular task explains the difference.

	CER (%)
ECOC SVM (identity matrix)	23.44
Multiclass SVM	23.93

Table 3: Comparison of the direct multiclass SVM and combining multiple binary SVMs via ECOC using the identity code matrix.

3.2.3. Maxent and SVM

In our previous studies, Maxent achieved better performance than the HMM [7], therefore we will only compare the SVM results to Maxent. Another reason that we choose to compare Maxent with SVMs is that both of them formulate DA tagging as a classification task, whereas HMM and DBN use sequence decoding.

For Maxent, a direct multiclass is straightforward, since it estimates directly the conditional probability $P(C_i|O)$. This is the approach used in our previous work [1, 3]. In order to compare

with SVMs, we also evaluated the ECOC combination of the Maxent classifiers using the same code matrices as used for SVMs. We examined two different distance measures: hard decision or soft decision that preserves the posterior probabilities as shown in Equation (1). The exhaustive code matrix was used for this comparison due to its comparable performance and the straightforward design of the code.

Table 4 shows the results comparing the Maxent and SVM for the DA classification task. The Maxent model compares favorably to the SVM method. The direct multiclass Maxent performs significantly better than ECOC combination of the Maxent base classifiers, suggesting that the direct Maxent approach may be able to capture some discriminative features that directly distinguish different classes.

		CER (%)
Maxent	Multiclass	20.94
	ECOC hard	22.76
	ECOC soft	22.10
SVM	Multiclass	23.93
	ECOC	23.16

Table 4: DA classification error rate (%) using the Maxent and SVM classifiers.

3.2.4. Features

In all the results reported above, the same features were used as those in our previous work [1, 7]. After adding PP-related features generated from the DA specific word N-gram LMs as described in Section 3.1.2, we found that there is a slight improvement in classification performance (error rate of 20.94% versus 20.54%) in the Maxent classifier. Similar performance is also observed in SVMs.

4. Discussion

We have investigated utilizing SVMs for multiclass DA classification in the multiparty ICSI meeting corpus. Various techniques are studied, either using a direct multiclass SVM or combining multiple binary SVM classifiers via the general ECOC scheme. For this task, we found that the code design does not have a significant impact on performance. The combination of multiple binary SVMs via ECOC achieves better performance than a direct multiclass SVM. We also compared SVM classifiers with our previous work using a Maxent model, and found that Maxent outperforms the SVM approach, and that in contrast to SVMs, combining multiple Maxent classifiers via ECOC is not as good as a direct multiclass Maxent model.

We believe that these experiments are still preliminary and that the results are task and features dependent. More analysis is needed for the errors made by different classification approaches and the impact of different training size. Due to the limited feature set used here and the inherent ambiguity of the task, there may not be much gain that can be possibly achieved by combining multiple binary SVMs via ECOC. However, the study in this paper sets a general framework that allows us to investigate the DA classification task using various features and different DA classes. In the current task setup, there are only five classes, which limits the choices of the code matrix. When there are more classes involved, it is likely that there is more room for performance improvement using better code matrix. For code design, linguistic knowledge may also provide helpful guidelines on how to group the DA class tags into subgroups for large discrimination between clusters. In addition, we believe that there is interaction between the base classifiers' performance and the distance of the codewords that also affects the overall performance.

In future work, we plan to investigate incorporating more features in the classifiers, in particular the prosodic features, which help resolve ambiguity in some cases (e.g., a question using a surface statement representation). Improving base classifier performance by utilizing more discriminative features will also help subsequent classifier combination. We will study different DA class sets (e.g., a more fine grained DA tags) and investigate the possibility of better code matrix design. Finally we will use speech recognition output for DA classification and segmentation.

5. Acknowledgment

The author thanks Anand Venkataraman, Elizabeth Shriberg, and Andreas Stolcke at SRI for the useful discussions. This work is partly supported by DARPA under Contract No. HR0011-06-C-0023. Any opinions expressed in this work are those of the author(s) and do not necessarily reflect the views of DARPA. Part of the work was conducted while the author was at the International Computer Science Institute at Berkeley.

6. References

[1] J. Ang, Y. Liu, and E. Shriberg, "Automatic dialog act segmentation and classification in multiparty meetings," in *Proc. of ICASSP*, 2005.

[2] G. Ji and J. Bilmes, "Dialog act tagging using graphical models," in *Proc. of ICASSP*, 2005.

[3] A. Venkataraman, L. Ferrer, A. Stolcke, and E. Shriberg, "Training a prosody based dialog act tagger from unlabeled data," in *Proc. of ICASSP*, 2003.

[4] N. Webb, M. Happle, and Y. Wilks, "Dialog act classification based on intra-utterances features," in *AAAI workshop on Spoken Language Understanding*, 2005.

[5] R. Fernandez and R. Picard, "Dialog act classification from prosodic features using support vector machines," in *Proc. of Speech Prosody*, 2002.

[6] M. Mast and R. K. et al, "Dialog act classification with the help of prosody," in *Proc. of ICSLP*, 1996.

[7] A. Venkataraman, Y. Liu, E. Shriberg, and A. Stolcke, "Does active learning help automatic dialog act tagging in meeting data," in *Proc. of Eurospeech*, 2005.

[8] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class kernel-based vector machines," *Machine Learning Research*, pp. 265–292, 2001.

[9] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector learning for interdependent and structured output spaces," in *Proc. of ICML*, 2004.

[10] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problem via error-correcting output codes," *Journal of Artificial Intelligence Research*, pp. 263–286, 1995.

[11] A. Berger, "Error-correcting output coding for text classification," in *IJCAI: Workshop on Machine Learning for Information Filtering*, 1999.

[12] T. Cohn, A. Smith, and M. Osborne, "Scaling conditional random fields using error-correcting codes," in *Proc. of ACL*, 2005.

[13] E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey, "The ICSI meeting recorder dialog act MRDA corpus," in *Proc. of 5th SIGdial Workshop on Discourse and Dialogue*, 2004, pp. 97–100.

[14] A. Clark and A. Popescu-Belis, "Multi-level dialogue act tags," in *Proc. of 5th SIGDIAL Workshop on Discourse and Dialog*, 2004.

[15] F. MacWilliams and N. Sloane, *The Theory of Error-correcting Codes*, 1977.

[16] T. Joachims, "Making large-scale svm learning practical," in *Advances in Kernel Methods - Support Vector Learning*, 1999.