

Dialog Act Tagging with Support Vector Machines and Hidden Markov Models

Dinoj Surendran, Gina-Anne Levow

Computer Science Department, University of Chicago,
1100 E. 58th St., Chicago IL 60637, United States
dinoj,levow@cs.uchicago.edu

Abstract

We use a combination of linear support vector machines and hidden markov models for dialog act tagging in the HCRC MapTask corpus, and obtain better results than those previously reported. Support vector machines allow easy integration of sparse high-dimensional text features and dense low-dimensional acoustic features, and produce posterior probabilities usable by sequence labelling algorithms. The relative contribution of text and acoustic features for each class of dialog act is analyzed.

1. Introduction

There are several possible cues that humans and machines can use to identify the meaning of an utterance, such as whether it is a acknowledgement or clarification. Several approaches have been proposed to integrate such cues [1] [2] [3], using a combination of neural networks, decision trees, hidden markov models (HMMs), principal components analysis, and nearest neighbor algorithms.

We use support vector machines (SVMs) [4] and HMMs to obtain dialog act classification accuracy on a standard corpus that is better than previously published results. With SVMs, we make no special effort, other than scaling, to combine sparse text representations and dense acoustic measurements of different kinds. Posterior probability estimates from SVMs are input to a Viterbi algorithm [5] to make use of sequential information. This SVM-HMM combination results in classification accuracies of 42.5% and 59.1% respectively using acoustic and text features separately and 65.5% together.

In this document, we briefly describe the task and classification algorithms used, followed by several measurements of performance in experiments using text features only, acoustic features only, and a combination thereof.

2. Task Description

The HCRC MapTask corpus [6] is a collection of 128 2-speaker dialogs, of which we used the 64 ‘no-eye-contact’ dialogs. Each dialog has a ‘giver’ giving directions on a shared map to a ‘follower’, and is segmented into parts called ‘Dialog Acts’ (DAs). We assume this segmentation has already been done manually. DAs do not span speaker turns; turns can consist of multiple DAs.

DAs are labelled with one of a number of tags; the HCRC labellers used twelve tags, and a thirteenth for ‘uncodable’. As this study was explorational, we simply took the task to be a 13-class classification problem. The tags, and their relative frequencies, are described in Table 1.

The experiments reported here were done with four-fold cross-validation. The 64 dialogs considered here are organized into

Table 1: Tags used by the HCRC MapTask Labellers, from the Dialog Structure Coding Manual. Also shown is the percentage of dialog acts of each type.

Tag	%age	Description
instruct	15.1	commands partner to carry out action
explain	7.5	states information that partner did not elicit
align	7.2	checks attention & agreement of partner, or their readiness for next DA
check	8.0	requests partner to confirm information that checker is partially sure of
query-yn	6.0	Yes/No question other than a check or align
query-w	3.0	any other question
ack	21.0	acknowledgement: minimal verbal response showing that speaker heard preceding DA
clarify	4.0	repetition of information already stated by speaker, often in response to a check DA
reply-y	12.6	affirmative reply to any query
reply-n	3.3	negative reply
reply-w	3.2	any other reply
ready	7.9	DA that occurs after end of a dialog game and prepares conversation for a new game
uncodable	1.2	None of the above

eight parts, q1 to q8. We used four splits, with the n -th split ($n = 1, 2, 3, 4$) having test data from $q\{2n - 1\}$ and $q\{2n\}$ and the training data from the other six parts. Text features differed in each split as they could only make use of the split’s training part.

In this way, each of the 14810 examples in the corpus was a test example in exactly one split. The confusion matrices presented here are sums of all confusion matrices from all splits. The final classification accuracy is the sum of all correctly classified test examples divided by 14810.

In the interests of result reproducibility, we have placed the data and data splits used in our experiments online¹. Using the exact splits is important, as results varied widely across splits. For example, our final classification accuracy of 65.8% is a weighted sum of the accuracies for each split of 63.4%, 61.9% 69.4% and 70.9%. This wide variance is disconcerting but, as will be seen, even 61.9% is better than previously reported results that do not assume knowledge of the previous DA or higher level information.

¹<http://people.cs.uchicago.edu/~dinoj/da>

3. Classification Algorithms

Our strategy is to use linear SVMs on individual data points, and then Viterbi decoding to make use of some contextual information. The Viterbi decoding procedure is a bit non-standard as we have posterior probabilities instead of output symbol probabilities.

3.1. Support Vector Machines

SVMs are binary classifiers. Given set of training examples, each a D -dimensional vector, with labels -1 or 1, a linear SVM learns weights $w \in \mathbf{R}^D$ and a threshold $b \in \mathbf{R}$ to produce a final decision function $f(x) = \text{sign}(w^T x + b)$. The value of $f(x)$ is between $-\infty$ and ∞ but can (and should) be Platt-scaled [7] to produce a value between 0 and 1 that is a good estimate of the probability that example x is in class 1.

There are several ways to reduce multiclass problems to binary problems [8]. A fast method is to split the problem into $n(n-1)/2$ binary classification problems and combine the results [9] [10]. The probability estimates from each binary classification problem can be combined to produce an estimate of the posterior probability for x over the n classes [11].

Label bias was handled by associating a weight of $\frac{1}{p}$ to each class with empirical training set probability p .

3.2. HMM Decoding

Previous work in this domain [12] [3] [13] reports improved recognition rates when knowledge of the previous dialog act is assumed. This is because the distribution of DAs differs greatly with the previous dialog act — Table 2 shows these distributions for part of the MapTask corpus. For example, the probability of an affirmative reply is about 0.5 after a binary question or check or alignment, but under 0.05 after other types of DAs. And while checks and alignments are also questions, the probability of a negative reply after an alignment is about 0.01, after a check 0.08 and after a binary question 0.27.

Table 2: Transition matrix showing which dialog acts followed which dialog acts. The (i, j) th entry is the percentage of DAs after one of class i that are of class j . For example, of the dialog acts immediately following a binary question (qy), 45% were affirmative replies (xy) and 27% negative replies (yn).

	in	ex	al	ch	qy	qw	ac	cl	ry	rn	rw	rd	un
in	4	5	8	13	9	4	48	0	1	0	1	5	2
ex	8	10	5	7	7	3	43	1	2	2	0	11	2
al	12	4	3	7	4	2	6	2	51	1	2	5	1
ch	3	3	2	2	2	2	4	10	55	8	2	4	2
qy	1	3	0	2	4	1	4	0	45	27	7	3	2
qw	4	2	4	3	4	2	5	11	3	2	48	10	2
ac	29	10	12	6	6	3	11	4	2	1	1	14	1
cl	4	5	11	15	3	2	47	4	4	1	0	4	0
ry	17	7	7	9	5	2	23	9	4	0	1	14	1
rn	7	19	3	6	5	3	34	12	1	1	4	7	0
rw	7	6	4	8	3	5	47	2	2	1	6	10	2
rd	46	11	8	7	12	3	3	2	1	0	2	3	2
un	15	10	4	6	7	4	18	3	10	3	4	9	6

If the computer is a participant in the dialog, knowledge of previous DAs on one conversation side can be assumed, as in Taylor et. al.[12]. In general, knowledge of the preceding DA cannot be assumed. Of course, it can be estimated, and Stolcke et. al. [2] found that a HMM improved performance.

The HMM assumption is that the sequence of observations x_1, \dots, x_T is generated by an underlying first-order Markov sequence of states q_1, \dots, q_T with each observation x_t generated by the corresponding state q_t only. Note that we are using the DA classes as states.

Suppose there are N states. If we know the transition probabilities $P(q_{t+1}|q_t)$, the output symbol probabilities $P(x_t|q_t)$, and the initial probability distribution $P(q_1)$, the Viterbi forward-decoding algorithm [5] is a dynamic programming algorithm that outputs the most likely state sequence q_1, \dots, q_T given an output sequence x_1, \dots, x_T .

We can estimate the transition probabilities from the training set. However, we do not know $P(x_t|q_t)$, but do know the posterior probability $P(q_t|x_t)$ for each x_t in the test set [11]. Bayes Rule and some other assumptions can be employed to give the slightly modified Viterbi algorithm below [14].

$$\delta_{jt} := \max_{q_1, \dots, q_{t-1}} P(x_1, \dots, x_t, q_1, \dots, q_{t-1}, q_t = j)$$

$$\psi_{jt} := \text{argmax}_{q_{t-1}} \delta_{jt}$$

We use $P(q_1|x_1)$ for the initial state distribution, and then compute δ and record ψ recursively:

$$\delta_{j,1} = P(q_1 = j|x_1)$$

$$\delta_{j,t \geq 2} = P(q_t = j|x_t) \sum_{n=1}^N P(q_t = j|q_{t-1} = n) \delta_{n,t-1}$$

$$\psi_{j,t \geq 2} = \text{argmax}_{q_{t-1}} P(q_t = j|q_{t-1} = n) \delta_{n,t-1}$$

The equation for $\delta_{j,t \geq 2}$ is probabilistically incorrect; it should be $P(x_t|q_t)$. As we only want to find the most likely state sequence, and not its probability as well, we can use $c \cdot P(x_t|q_t)$ instead of $P(x_t|q_t)$, as long as c is a positive real number independent of q_t . Bayes Rule says that $P(x_t|q_t) = P(q_t|x_t)P(x_t)/P(q_t)$. Since we reweighted the probabilities in SVM training so that all classes were equally likely, $P(q_t) = \frac{1}{N}$ is independent of q_t , as is $P(x_t)$. Thus we can substitute $P(q_t|x_t)$ for $P(x_t|q_t)$ in our modified Viterbi procedure; $\psi_{j,t}$ remains the same.

Finally, backtracking obtains the most likely state sequence π_1, \dots, π_T :

$$\pi_T = \text{argmax}_q \delta_{qT}$$

$$\pi_{t < T} = \psi_{\pi_{t+1}, t+1}$$

4. Classification Experiments

4.1. Classification with acoustic features

We used the acoustic features mentioned by Stolcke et. al. [1] that made use of duration (but not pauses), intensity, pitch, speaking rate [15], and speaker identity. Pitch was measured with the ESPS algorithm implemented in the Snack Toolkit [16]. Each feature were scaled to between -1 and 1 based on the minimum and maximum values of the feature among training examples.

This resulted in a classification accuracy of 41.4% with a linear SVM, which was increased to 42.5% after Viterbi decoding. However, as the corresponding confusion matrix in Table 3 shows, nearly half the classes were rarely recognized, and over half the examples were classified as the most common classes.

This indicates that our acoustic features simply did not separate the data and need to be improved. However, previous work on this dataset using only acoustic features (and no context) produced similar recognition rates, such as 42% in [12].

4.2. Classification with text features

We represented text features of each DA using a sparse bag-of- n -grams model. Our features included all unigrams, bigrams, and trigrams that appeared at least twice in the training set. We also had a feature for a unigram that was the only word in a DA.

Suppose F of these features occurred at least twice. F was between 9000 and 10000, depending on the data split. Then each DA x in the training and test set was represented by a $(F + 1)$ -dimensional feature $v = [v_1 v_2 \dots v_{F+1}]^T$ where v_j , $j = 1, \dots, F$ was the number of times x generated feature v_j while v_{F+1} was the number of times x generated a unigram feature that was not among the F features. This dealt, albeit crudely, with the out-of-vocabulary problem.

Applying a linear SVM to this resulted in 58.1% classification accuracy, and in 59.1% once Viterbi decoding was applied. The corresponding confusion matrix is in Table 4. This is much better than the 42.8% when using 1-nearest neighbors i.e. classifying each test DA with the label of the training DA with the highest cosine similarity [3]. While it is less than the 62.1% reported using Transformational Based Learning by Lager and Zinovjeva [13], their algorithm assumed knowledge of the previous DA.

Table 3: *Confusion matrix using acoustic features with a linear SVM and Viterbi decoding. Classification accuracy was 42.5%*

	inst	ex	al	ch	qy	qw	ac	cl	ry	m	rw	rd	un
in	1932	7	72	4	29	0	25	14	44	3	4	107	0
ex	308	211	25	245	11	17	178	4	52	3	7	43	0
al	441	15	171	43	29	1	83	5	90	4	3	184	1
ch	89	135	12	516	11	17	315	2	65	2	6	15	0
qy	368	37	89	154	89	11	81	5	28	2	6	19	0
qw	57	51	12	92	8	13	165	1	20	3	3	13	0
ac	59	55	21	159	6	6	2067	8	354	7	6	358	3
cl	448	12	18	7	11	1	12	8	48	2	7	37	0
ry	130	19	21	56	3	3	780	8	597	15	10	218	0
m	27	5	5	23	0	1	173	1	146	13	4	85	0
rw	165	39	12	79	8	0	55	13	63	0	11	24	0
rd	63	5	29	9	4	0	246	1	137	7	3	662	3
un	10	0	12	5	3	0	60	1	46	1	0	43	1

Table 4: *Confusion matrix using text features with a linear SVM and Viterbi decoding. Classification accuracy was 59.1%*

	inst	ex	al	ch	qy	qw	ac	cl	ry	m	rw	rd	un
in	1780	90	47	101	27	18	61	33	17	2	18	43	4
ex	162	591	17	93	35	11	84	22	21	20	26	20	2
al	119	29	382	54	44	8	327	9	9	2	7	80	0
ch	182	114	42	520	106	23	96	30	34	12	14	9	3
qy	66	27	39	129	550	15	30	8	8	5	11	1	0
qw	38	16	11	27	25	254	30	4	5	2	11	10	5
ac	49	42	86	53	13	8	2103	10	349	49	2	326	19
cl	332	60	12	54	8	8	35	44	11	3	33	5	6
ry	26	38	13	28	6	4	420	14	1234	3	11	59	4
m	2	12	0	3	6	0	33	0	3	419	2	0	3
rw	99	87	8	27	10	6	23	16	18	7	146	17	5
rd	28	6	36	12	3	6	355	4	10	2	5	692	10
un	12	3	2	18	3	7	59	3	2	4	5	31	33

Table 5: *Confusion matrix using acoustic and text features with a linear SVM and Viterbi decoding. Classification accuracy was 65.5%*

	inst	ex	al	ch	qy	qw	ac	cl	ry	m	rw	rd	un
in	1923	48	42	30	24	5	23	67	9	6	18	41	5
ex	122	598	36	120	32	12	64	31	22	21	23	18	5
al	126	24	587	34	49	7	71	15	9	2	8	137	1
ch	50	118	27	683	102	38	93	17	22	7	13	13	2
qy	35	35	54	154	534	18	16	19	3	5	8	6	2
qw	12	26	6	33	21	280	29	5	6	2	6	7	5
ac	19	54	51	52	10	8	2300	5	280	42	4	267	17
cl	356	36	19	23	5	3	7	87	15	0	40	18	2
ry	35	33	10	16	3	2	408	29	1282	6	10	22	4
m	6	13	0	9	6	0	33	2	4	404	1	2	3
rw	87	89	9	30	7	3	12	26	23	9	153	15	6
rd	32	7	39	5	1	4	229	6	7	2	5	824	8
un	16	4	2	9	2	10	56	4	2	6	4	26	41

4.2.1. Preprocessing Text with PCA

Serafin and di Eugenio [3] suggest using Principal Components Analysis on the sparse text features before applying a nearest neighbors classifier. We therefore ran a linear SVM on the first 100 principal components of the sparse text features and obtained classification accuracy of 55.7%, compared to 58.1% without using PCA. We thus performed no further experiments with PCA.

4.3. Classification with both features

Recall that we have so far for each DA a G -dimensional dense vector representing its acoustic properties and a F -dimensional sparse vector representing its textual features. $F \approx 10000$ is much larger than $G \approx 50$. The easiest way of integrating them is to concatenate the two vectors to form a $(F + G)$ -dimensional sparse vector and feed this to the SVM.

With vector concatenation, classification accuracy was 61.8% using a linear SVM and 65.5% after Viterbi decoding. The confusion matrix in Table 5 has more details. This compares with 59.1% and 42.5% using text and acoustic features separately.

This is better than previously reported results for this dataset. Higher accuracy, such as 73.9% in [3], has only been achieved by assuming knowledge of higher level discourse information such as game segmentation and game type for each DA.

A better sense of how different features helped can be found in Table 6. It has precision, recall and F-scores for the cases when acoustic and text features were used separately and together.

Unsurprisingly, all classes were better recognized with better precision using text than acoustic features. Bear in mind that we are using manually, not automatically, transcribed text features. Considering F-scores, acoustic features did not aid text features in recognizing binary questions, possibly because most `query-yn` DAs have helpful bigrams like “have you” or “do you” or “am i”. They help a little with recognizing complex queries, since though over 75% of most `query-w` DAs have the word “where”, “how” or “what”, so do other DAs. Acoustic features also help with recognizing questions that were `check` or `align`, but not with recognizing any replies. They did help with recognizing `ready`, `instruct` and `ack` DAs.

Table 6: *Precision, Recall, and F scores for each class using acoustic (A), text (T) or both (B) features. Values are percentages.*

Tag	Precision			Recall			F-score		
	A	T	B	A	T	B	A	T	B
instruct	47	61	68	86	79	86	61	69	76
explain	36	53	55	19	54	54	25	53	55
align	34	55	67	16	36	55	22	43	60
check	37	46	57	44	44	58	40	45	57
query-yn	42	66	67	10	62	60	16	64	63
query-w	19	69	72	3	58	64	5	63	68
ack	49	58	69	66	68	74	56	62	71
clarify	11	22	28	1	7	14	2	11	19
reply-y	35	72	76	32	66	69	34	69	72
reply-n	21	79	79	3	87	84	5	83	81
reply-w	16	50	52	2	31	33	4	38	40
ready	37	54	59	57	59	70	44	56	64
uncodable	13	35	41	1	18	23	1	24	29

4.3.1. Online Classification

The acoustic features we used were normalized by conversation side, which requires that one needs to see the entire dialog before classifying any dialog act. This makes online testing impossible.

We thus investigated the possibility of not normalizing the acoustic features. Scaling was still done as it can be applied online to individual test examples. The classification accuracy was then 42.4% and 65.3% using acoustic and acoustic+text features respectively, an absolute drop of only 0.1% and 0.2% respectively from the non-normalized case.

Since all other parts of our algorithmic framework are online in the test phase (including the Viterbi algorithm, since it is only a forward pass), this algorithm is suitable for online testing.

5. Conclusion

In this paper, we showed that linear support vector machines can easily integrate text and acoustic features, and that class posterior probabilities estimated from their outputs can be input to HMMs to produce a simple, fast dialog act classification algorithm.

Our acoustic features improved the quality of recognition for a few classes, namely instructions, acknowledgements, and all queries other than binary ones.

Future work will investigate the use of better prosodic features, Multiple Kernel Learning [17][18] to integrate different features, SVMs that directly incorporate sequential information [19], and methods for DA segmentation.

6. Acknowledgements

Funding for this project came from NSF Grant 0414919. We are very grateful to the Dialogue Group at the Human Communication Research Centre in Edinburgh for creating and releasing the Map-task corpus, and to Chih-Jen Lin and Chih-Chung Chang for their LIBSVM toolkit [10] used for our experiments. Nelson Morgan provided the source code to compute enrate parameters. Thanks also to Gunnar Rättsch, Arthur Gretton, and John Langford for very helpful discussions.

7. References

- [1] Elizabeth Shriberg, Rebecca Bates, Andreas Stolcke, Paul Taylor, Daniel Jurafsky, Klaus Ries, Noah Coccaro, Rachel Martin, Marie Meteer, and Carol van Ess-Dykema, "Can prosody aid the automatic classification of dialog acts in conversational speech?," *Language and Speech*, 1998.
- [2] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol van Ess-Dykema, and Marie Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Computational Linguistics*, vol. 26, pp. 339–373, 2000.
- [3] Riccardo Serafin and Barbara di Eugenio, "Flsa: Extending latent semantic analysis with features for dialogue act classification," in *Proceedings of the 40th Annual Meeting of the Assoc. for Computational Linguistics*, 2004, pp. 692–699.
- [4] Corinna Cortes and Vladimir Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.
- [5] Andrew Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–267, April 1967.
- [6] Jean Carletta, Amy Isard, Stephen Isard, Jacqueline C Kowtko, Gwyneth Doherty-Sneddon, and Anne H Anderson, "The reliability of a dialog structure coding scheme," *Computational Linguistics*, vol. 23, pp. 13–31, 1997.
- [7] John Platt, "Probabilistic outputs for support vector machines and comparison to regularized likelihood methods," in *Advances in Large Margin Classifiers*, A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, Eds., 2000, pp. 61–74.
- [8] Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, and Bianca Zadrozny, "Reductions between classification tasks," in *Proceedings of the 22nd International Conference on Machine Learning*, New York, NY, USA, 2005, ACM Press.
- [9] Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.
- [10] Chih-Chung Chang and Chih-Jin Lin, "Libsvm : a library for support vector machines," *Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>*, 2001.
- [11] Ting-Fan Wu, Chih-Jin Lin, and Ruby C. Weng, "Probability estimates for multi-class classification for pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.
- [12] Paul Taylor, Simon King, Stephen Isard, and Helen Wright, "Intonation and dialog context as constraints for speech recognition," *Language and Speech*, vol. 41, pp. 489–508, 1998.
- [13] Torbjorn Lager and Natalia Zinovjeva, "Training a dialogue act tagger with the μ -tbl system," in *Third Swedish Symposium on Multimodal Communication, Linkoping University Natural Language Processing Laboratory (NLPLAB)*, October 1999.
- [14] Gunnar Rättsch and Stefan Sonnenburg, *Accurate splice site prediction for Caenorhabditis Elegans*, pp. 277–298, MIT Press, Cambridge, MA, USA, 2004.
- [15] Nelson Morgan, Eric Fosler, and Nikki Mirghafori, "Speech recognition using on-line estimation of speaking rate," in *Proceedings of Eurospeech '97*, 1997, pp. 2079–2082.
- [16] Kåre Sjölander, "The snack sound toolkit," 1997.
- [17] Francis R. Bach, Gert R. G. Lanckriet, and Michael I. Jordan, "Multiple kernel learning, conic duality, and the smo algorithm," in *ICML '04: 21st International Conference on Machine Learning*, New York, NY, USA, 2004, ACM Press.
- [18] Stefan Sonnenburg, Gunnar Rättsch, and Christin Schäfer, "Learning interpretable svms for biological sequence classification," in *RECOMB 2005, LNBI 3500*, S. Miyano et al., Ed. 2005, pp. 389–407, Springer-Verlag Berlin Heidelberg.
- [19] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann, "Hidden markov support vector machines," in *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.