

Cooperative Sign Language Tutoring: A Multiagent Approach

İlker Yıldırım, Oya Aran, Pınar Yolum, Lale Akarun
Department of Computer Engineering
Boğaziçi University
Bebek, 34342, Istanbul, Turkey
{ilker.yildirim, aranoaya, pinar.yolum, akarun}@boun.edu.tr

Abstract. Sign languages can be learned effectively only with frequent feedback from an expert in the field. The expert needs to watch a performed sign, and decide whether the sign has been performed well based on his/her previous knowledge about the sign. The expert's role can be imitated by an automatic system, which uses a training set as its knowledge base to train a classifier that can decide whether the performed sign is correct. However, when the system does not have enough previous knowledge about a given sign, the decision will not be accurate. Accordingly, we propose a multiagent architecture in which agents cooperate with each other to decide on the correct classification of performed signs. We apply different cooperation strategies and test their performances in varying environments. Further, through analysis of the multiagent system, we can discover inherent properties of sign languages, such as the existence of dialects.

1 Introduction

Sign language is the natural means of communication for the hearing-impaired. These visual languages are based on *signs*, which are a combination of hand gestures, facial expressions, and head movements. A sign that is composed of hand gestures is called a *manual sign*, whereas the head movements or facial expressions are called *non-manual sign*. Teaching these signs to others is an important, but a difficult task. A person can improve her performance of signs only with frequent attempts and feedback. To automate the teaching of sign languages an automated sign language tutoring tool called *SignTutor* has been developed [1]. The aim of this system is to help users learn isolated signs by watching recorded videos and to enable them to try those signs on their own. The system records a user's video while she is performing a sign. After analysis of the user's sign performance, the system gives the user feedback on her performance. The system can recognize manual signs as well as non-manual, complex signs, which include hand movements and shapes, together with head movements, and facial expressions. The system uses a classifier for recognition, by which it can compute a similarity score with a level of certainty for the user's sign performance. This classifier is the key component in deciding whether the user has performed the sign correctly or not. SignTutor is specialized for Turkish Sign Language.

The current system is a stand-alone application. That is, each system has its own data and own set of classifiers and there is no communication between different instances of the system. Hence, two students that are practicing the same set of signs on different stations cannot use each other's data or feedback. This is an obvious disadvantage. Intuitively, different systems will have varying data and varying performances of classifiers. That is, a system may classify sign A correctly, but may be incompetent in classifying sign B , whereas a second system may have complementary expertise. It is most appropriate for these systems to cooperate with each other when making decisions.

This challenge can be addressed by a traditional approach in which there is a centralized database and a classifier, where every tutoring system acts as a client. Clients then need to be connected in order to facilitate tutoring. This architecture is not plausible in our situation, because first, we know that the clients may not be online all the time, and second, videos may belong to certain individuals who do not want to share them on a central database.

Accordingly, we propose to encapsulate each instance of the SignTutor in an agent. Agents are distributed geographically, but will be able to communicate with each other over the Internet, forming a cooperative multiagent system [2]. Each agent is associated with a local database of signs and a classifier. An agent can improve its classification performance due to its own experience. An agent may decide to include a practice sign in its training data or a sign language teacher may add new training data. Moreover, agents can help each other classify signs by exchanging classification requests. Thus, even when an agent's own classifier is not trained to classify a sign accurately, it can collect answers from others and decide autonomously. Since agents have their own local databases they can make a decision even when they cannot or do not want to communicate with other agents.

However, realizing this multiagent system comes with challenges. The most important one is that when agents have varying expertise in different classes, it is not immediately clear whom to ask for help. Accordingly, we study different cooperation strategies deeply to understand their strengths and weaknesses in different environments.

The rest of this paper is organized as follows: Section 2 formalizes the problem of identifying agents to cooperate with. Section 3 explains different cooperation strategies that have been developed to help agents decide on sign classification. Section 4 evaluates these cooperation strategies on real sign language data. Section 5 explains how important sign language properties can be inferred from a multiagent system. Finally, Section 6 discusses our work with comparisons to the literature.

2 Problem Definition

In this study, rather than focusing on how agents can improve the capability of their own classifiers, we are interested in the problem of cooperation for making better decisions for incoming classification requests. We illustrate the problems

of cooperation associated with our architecture on a simple setting. Figure 1 depicts a possible setting for cooperative multiagent architecture, which consists of three agents that can communicate with each other. Each agent has a classifier and this classifier is trained with data in the database associated with it.

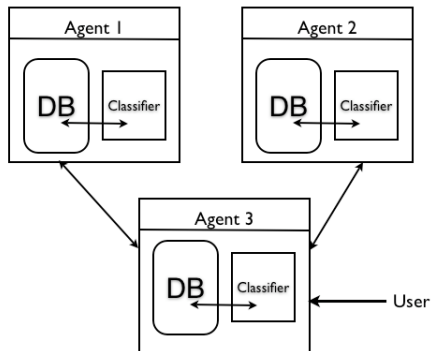


Fig. 1. A setting for multiagent architecture for Turkish sign tutoring tools

As described above, the tutoring tool aims to make people learn sign language on their own. The tutoring tool shows videos of signs as queried by the human learner. A sign language learner requests the agent to recognize her performance, V_{human} , and decide whether it is an instance of sign class C_{claim} . The agent then queries a subset of other agents, A_Q , in the system by communicating to them V_{human} , for their top m class predictions, $P_i^{(1)}, \dots, P_i^{(m)}$, with associated scores, $S_i^{(1)}, \dots, S_i^{(m)}$ and certainty values, $C_i^{(1)}, \dots, C_i^{(m)}$ for all $i \in A_Q$. Now, the agent has to decide if V_{human} belongs to C_{claim} by combining predictions, scores and certainty values of all queried agents. Therefore, the problem is to design cooperation strategies in our architecture, which enable agents to effectively combine results of other agents and to achieve better classification performance than it would do on its own.

3 Cooperation strategies

In our context, cooperation strategies are expected to enable each agent achieve better tutoring, which corresponds to increased predictive accuracy in recognizing performances of human learners. As discussed above in Section 2, predictions, scores and certainties of other agents are available to each agent by communication. Therefore, we develop cooperation strategies that exploit data gathered through communication.

A valid interpretation of the cooperation strategy problem is classifier combination. In this view, the agent that is to make the decision is responsible for

gathering predictions of other agents and applying any classifier combination method, such as voting and score fusion. In Sections 3.1 and 3.2, we further elaborate state-of-the-art classifier combinations methods [3].

Another suitable interpretation for the cooperation strategy problem is teammate modeling in cooperative multiagent learning, where agents model each other in order to make good guesses about their future behavior [4]. In our case, each agent in our system, by communicating predictions for human performances maintain probabilistic models of each other. Each agent, then, uses these models of other agents in the system to decide which agents to query for a given human performance, and how to combine responses of these agents to make a better prediction next time. We propose two probabilistic methods, one incorporates prior knowledge to build a model, and the other uses observations coming from interactions. The details of these methods are explained in section 3.3.

3.1 Voting

Voting is a common method for combining classifiers, and has proven useful many times in the literature [5, 6]. While applying voting schemes as cooperation strategies, we consider responses coming from other agents as votes, and the decision making agent which is responsible to respond to the user (the agent that is queried by the human learner) is responsible for counting the votes in terms of a specific scheme, and making the decision accordingly. We call this agent the decision maker. For instance, in our sample setting seen in Figure 1, the user performs a sign, which is captured by agent 3 as V_{human} , and the user also claims that she performed an instance of the sign class C_{claim} . Following this, agent 3 queries both agents 1 and 2 to collect their votes for the class of sign to which V_{human} belongs. Among several voting schemes we study majority voting, weighted voting and Borda count schemes. If at the end of counting of votes, agent 3 decides that the performance V_{human} belongs to the class C_{claim} , then it responds verbally as *OK*, and otherwise as *WRONG* to the user.

Majority voting: Majority voting is the application of majority rule, which selects the one of two choices with more than half of the votes to make a decision [7]. To apply the majority rule, the decision making agent, agent 3 in our sample setting, needs to retrieve the top predictions, $P_i^{(1)}$, for all $i \in A_Q$, the agents that are queried (in this case agents 1 and 2). There are two outcomes of majority voting, *OK* or *WRONG*. A vote $P_i^{(1)}$ is counted for *OK* if it is equal to C_{claim} , and counted for *WRONG* otherwise. The one which has more than half of the votes is the final decision. In case of a tie, the decision maker selects one of the possibilities randomly.

Weighted voting: The weighted voting scheme is based on the idea that not all voters are equal, but instead, each voter has an associated weight and her vote is counted according to this weight. This time, the decision maker collects certainty values (which corresponds to weights) and predictions for their top choices, $C_i^{(1)}$ and $P_i^{(1)}$ respectively for $i \in A_Q$. The decision maker needs to

count the weighted votes as follows.

$$R = \sum_{i \in A_Q} K * C_i^{(1)}$$

where

$$K = \begin{cases} 1 & \text{if } P_i^{(1)} = C_{claim} \\ -1 & \text{if } P_i^{(1)} \neq C_{claim} \end{cases}$$

and

$$\text{Final Decision} = \begin{cases} OK & \text{if } R \geq 0 \\ WRONG & \text{if } R < 0 \end{cases}$$

One problem associated with weighted voting is the assessment of weights. Without elaborating on this problem, we simply make agents assess a certainty value for their votes as $C_i^{(1)} = S_i^{(1)}/S_i^{(2)}$, where, $S_i^{(1)}$ is the score calculated for top prediction, $P_i^{(1)}$, by agent i and $S_i^{(2)}$ is the score of the second prediction, $P_i^{(2)}$.

Borda count: Borda count is a voting scheme, in which voters rank candidates (or a subset of candidates) in the order of preference. Each candidate’s score is the summation of points—the higher the position of the candidate in the rank of a voter, the higher the score—over all voters. The winner in Borda count is the candidate with the maximum score.

In Borda count, the decision maker collects the top k predictions, $P_i^{(1)}, \dots, P_i^{(k)}$ from each agent $i \in A_Q$, the subset of agents that are queried. Each position in a rank of k predictions has a specific score. For instance if $k = 3$, a possible scoring could be 15 points for the first position, 10 and 5 points for the second and third positions, respectively. After counting the total score for each class, the one with the maximum score is selected. If the selected class is C_{claim} , then feedback is generated as *OK*, otherwise it is generated as *WRONG*. In our experiments we set $k = 3$, and scores as 3 points for the first position, 2 points for the second position, and 1 point for the last position in the rankings.

3.2 Score level fusion

In score level fusion, instead of the predictions themselves, the scores (confidences, likelihoods, and so on) of the prediction, coming from different experts, are fused to give the final decision [8,9]. Several combination rules, such as sum rule or product rule, can be applied to combine data coming from different sources in order to achieve better inference. Here, we apply sum rule for score level fusion.

The decision making agent gathers predictions, $P_i^{(1)}, \dots, P_i^{(k)}$, and their associated scores, $S_i^{(1)}, \dots, S_i^{(k)}$ for the top k choices for all agents $i \in A_Q$. For instance, in our sample setting in Figure 1, once the decision maker, agent 3,

receives the data coming from agents 1 and 2, it can proceed to apply score level fusion methods to generate the user feedback. In our experiments we set $k = 3$.

Given that the decision maker has $P_i^{(1)}, \dots, P_i^{(k)}$ and $S_i^{(1)}, \dots, S_i^{(k)}$ for all $i \in A_Q$, score fusion by sum rule calculates new integrated score for each sign class as follows: $sumScores(P_i^{(j)}) = sumScores(P_i^{(j)}) + S_i^{(j)}$
for all $i \in A_Q$ and $1 \leq j \leq k$,

where $sumScores$ is a vector of size of the number of different sign classes, and it is initially all 0s. The position of the maximum value in $sumScores$ is the decision of the decision making agent. If the decision is the same as C_{claim} then the feedback is generated as *OK*, and otherwise as *WRONG*.

3.3 Modeling agents

As we have discussed earlier, the aim of cooperation strategies is to enable agents to achieve better tutoring, i.e. increased predictive accuracy in recognizing human performances. Agents in our architecture are heterogeneous in the sense that their classifiers are trained using different databases, and they aim to improve via experience. The heterogeneity of agents in the system could be of advantage if a proper cooperation strategy is designed.

Strategies we have proposed up to now do not actually take advantage of agents' being heterogeneous. For instance, in majority voting and Borda count schemes, the decision-making agent totally ignores differences among agents, and acts as if everyone is equally knowledgeable for all queries. Although in weighted voting each agent's vote is counted according to their certainty values, one's evaluating its own weight may not always be a good choice. For instance, among other reasons, weights gathered this way have no meaning in terms of relative certainty of two agents, because they are all generated independent from others by all means. Similarly score fusion techniques also suffer from the same case of being relatively unnormalized.

One can exploit the heterogeneity of the system by designing a strategy in which each agent models others explicitly in terms of what they know and how well they know what they know, or in other words a strategy by which each agent models expertise of other agents. Once an agent has modeled expertise of other agents, it can query them accordingly, and still end up as a better tutor than it would otherwise be. We now propose two different approaches for modeling other agents in the system, the Observation-based model, a simple model of counting success and failure times in previous predictions by each agent for each sign class, and the Bayesian model, on top of the Observation-based model, incorporating some prior knowledge related with success of others for each different sign class ¹.

Observation-based model: In multiagent games, a simple model for modeling other agents in the system to predict their future behavior and achieve

¹ In terms of our modeling approaches, we do not need to consider the exploration versus exploitation tradeoff, because agents can only explore when extra data for modeling others are available. When such data is available, agents use this data to update all models they maintain.

coordination is called fictitious play, in which agents maintain empirical distribution of previously observed behavior for each agent, and use these distributions to predict behavior of others. As shown previously and repeated many times, in many settings, fictitious playing agents can achieve coordination [10].

In our work, we use the same idea to model predictive accuracy of other agents in the system for each class of sign. But in our case, agents need to communicate with each others training data sets to build up models of predictive accuracy. More specifically, agent i has a data set D , which consists of performance instances (videos) for different sign classes. Agent i queries a set of other agents, A_T with D and collects their top predictions $P_i^{(1)}$ for each item in D . Using these predictions, agent i calculates an empirical distribution for each agent j in A_T in terms of how accurate agent j is in predicting an instance of sign class c , which is the reliability R_i^c . For each sign class c and for each agent i the reliability is calculated as follows:

$$R_i^c = \frac{TP}{TP + FA} \quad (1)$$

where TP is the number of times agent i predicted an instance of the sign class c correctly (number of true positives), and FA is the number of times the agent predicted c when it was not c (false accepts).

Agents can use their Observation-based models in combining responses to make a better decision. For instance, an agent can decide to stick to prediction of a particular agent in a particular sign class based on its reliability. Formally, an agent decides based on its Observation-based model as follows. The agent collects the top prediction, $P_i^{(1)}$, for all agents $i \in A_Q$, a subset of all agents that are queried. A value for each sign class is calculated using reliability of all agents in their predicted class:

$$Value(P_i^{(1)}) := Value(P_i^{(1)}) + R_i^{P_i^{(1)}} \quad (2)$$

where $R_i^{P_i^{(1)}}$ is the reliability of agent i in sign class $P_i^{(1)}$, and $Value$ is a vector of size of number of sign classes. The decision making agent then, averages each item in $Value$ depending on the number of agents that predicted that particular sign class. The agent makes its decision as the position of the maximum in $Value$.

To illustrate Observation-based modeling strategy we study an example on our sample setting as seen in Figure 1, where agent 3 models agents 1 and 2 for three different sign classes, namely “Study”, “Study regularly”, and “Study continuously”. For each of these three classes, agent 3 queries both agents for 5 instances (Hence 15 instances are used for modeling at total). Table 1 shows reliability values calculated by agent 3 according to Equation 2. For example, agent 1 responded correctly four out of five times for the “Study” class, whereas agent 2 only answered two out of five correctly for the same class.

The values in Table 2 are obtained after normalization of reliability values among agents. Now, having calculated normalized reliability values of others, agent 3 will decide for a test item by querying other agents. Suppose that, agent

Reliability	Agent 1	Agent 2
Study	0.8	0.4
Study regularly	0.6	1
Study continuously	0.8	0.6

Table 1. Reliability values of both agents for three classes before normalization.

1 decides that this test item belongs to the third class, “Study continuously”, whereas, agent 2’s decision is the second class, “Study regularly”. After collecting responses of both agents, according to its models, agent 3 favors agent 2’s decision (agent 2’s value of 0.62 is greater than agent 1’s value of 0.57).

Reliability	Agent 1	Agent 2
Study	0.67	0.33
Study regularly	0.38	0.62
Study continuously	0.57	0.43

Table 2. Reliability values of both agents for three classes after normalization.

Bayesian model: A rather important point in modeling is that the more accurate the models of others, the better decisions made out of others’ responses. Especially if observations are scarce, (i.e. due to cost or availability), agents can achieve better modeling if they rely on their prior information regarding other agents in the system. In our system, the observations between agents that are useful for modeling are really rare. Therefore, incorporating prior information turns out to be critical and could be of help.

In the Observation-based Model, agents only use their interaction history for other agents, and build their models only according to their observations. One can extend this approach by incorporating the available a priori information in a Bayesian fashion. For instance, a relevant prior information about different sign classes is their similarity with each other. On top of the observations collected, augmenting the prior information, which says that a particular pair of similar signs are more likely to be confused with each other, can increase accuracy of models.

In our model, several subsets of sign classes are very similar in each other.² This similarity a priori says that an agent is more likely to confuse the class of a sign with another class which is similar to the real class. More formally, let S be the class of all signs, and $Conf_1, \dots, Conf_\kappa, \dots, Conf_n$ be disjoint confusion sets, such that $S = Conf_1 \cup \dots \cup Conf_\kappa \cup \dots \cup Conf_n$. Then, we say that any instance of sign s in $Conf_\kappa$ is more likely to be confused with another instance

² In our Bayesian model, a priori, each agent takes it for granted that every other agent is likely to confuse a pair of similar signs. But instead, there could be other types of priors and each agent could have a specific distribution over these possible priors.

of a sign in $Conf_\kappa$, than it is to be confused with an instance of a sign in the set $S - Conf_\kappa$.

We quantify how much two classes of signs are confused with each other as follows: Let $Conf_\kappa = s_1, \dots, s_j, \dots, s_c$, and let $R_i^{(s_j)}$ be the reliability of agent i in class s_j as described in Section 3.3. We calculate the posterior reliability of agent i in sign class s_j as follows:

$$PR_i^{s_j} = R_i^{s_j} - \frac{\sum_{s_h \in Conf_\kappa} (1 - R_i^{s_h})}{|Conf_\kappa|} \quad (3)$$

where $|Conf_\kappa|$ is the cardinality of the set $Conf_\kappa$. This formula says that to calculate the posterior reliability, we decrease the reliability of agent i in sign s_j by the average unreliability of agent i in signs in $Conf_\kappa$. The underlying intuition is that if agent i is wrong in its prediction of s_j , then the reason is its unreliability in predicting the true class s_{true} where both s_j and s_{true} are certainly in $Conf_\kappa$.

Once posterior reliability values are calculated, the decision making agent can proceed as in the Observation-based Model. But this time, the decision maker adopts the prediction of the agent with the highest posterior reliability without averaging them. Therefore, the decision maker does maximum a posteriori (MAP) estimation over the posterior reliability distribution of other agents. In other words, the decision making agent is more likely to stick to the prediction of the agent which is not only reliable in terms of the sign class of its prediction, but also reliable in the sign classes that are likely to be confused with that of prediction.

Having formally described Bayesian model, we now illustrate how it works by following our example above at the end of Observation-based model, Section 3.3. Firstly, notice that the three classes we selected, “Study”, “Study regularly”, and “Study continuously” all belong to the same confusion set. Table 3 shows posterior reliability values calculated by agent 3 according to Equation 3 and Table 1.

Posterior Reliability	Agent 1	Agent 2
Study	0.5	0.2
Study regularly	0.4	0.5
Study continuously	0.5	0.3

Table 3. Posterior reliability values of both agents for three classes before normalization.

Values in Table 4 are obtained after normalization of posterior reliability values in Table 3 again for each sign and over all agents. Again, agent 3 is to test the same data item as in Observation-based model. Before studying what Bayesian model predicts, here we reveal that the test item belongs “Study regularly” class. Therefore, Observation-based model actually fails to predict the test item correctly (its prediction is “Study continuously”).

Same as the Observation-based model example, agent 1 decides that the test item belongs to the third class, “Study continuously”, whereas agent 2 decides it belongs to the second class “Study regularly”. Our Bayesian model believes that a posteriori agent 1 is more reliable than agent 2 in terms of their decisions, hence it predicts that the test item belongs to the third class, “Study continuously”, which is the correct prediction (agent 1’s value of 0.63 is greater than agent 2’s value of 0.56).

Posterior Reliability	Agent 1	Agent 2
Study	0.71	0.29
Study regularly	0.44	0.56
Study continuously	0.63	0.37

Table 4. Posterior reliability values of both agents for three classes after normalization.

4 Evaluation of cooperation strategies

We evaluate the performance of our cooperation strategies on a dataset of 19 signs (We use exactly the same dataset used in [1]). For each sign, the dataset consists of five repetitions from eight different subjects. We measure the performance of each cooperation strategy by the predictive accuracy of the decision making agent. We also compare the performance of decision making agent with the performance of each single agent queried. We perform our analysis in several experiments. All results reported are average of five runs for each experiment (See Table 5 for a summary of each experimental setting).

Setting	# of				Total
	Trained agents	Random agents	Semi-oracles	Quarter-oracles	
One	2	-	-	-	2
Two	2	2	2	-	6
Three	2	4	-	4	10

Table 5. Summary of experimental settings, see text for details.

In the first set of experiments, we examined how cooperation strategies perform given there is a very limited number of agents to query. In this setting, there are two agents which are trained with performance instances of several subjects. And there is one decision making agent, which is responsible to query others and combine their responses (See Figure 1 for an instance of the first setting). In this setting, we make up test sets using performance instances of one subject. We train the first of two agents using instances of four other subjects, and the second agent is trained by performances of three subjects. There is one

overlapping subject that we use to train both agents, therefore at total, data from six subjects are used to train classifiers of the two agents. Performance instances of one of the remaining two subjects are used by the decision making agent to model other agents. The instances of the remaining last subject is left for the test. We apply eight-fold cross-validation to generate test and training sets. For example, in a fold, we use instances from subject one (95 instances) for test, and use instances from subjects two, three, four and five (380 instances) for training first agent, and use instances from subjects five, six and seven (285 instances) for training the second agent, and use instances from subject eight (95 instances) to train the decision making agent for the models of the first and second agents.

Subjects	Agent 1	Agent 2	Majority Voting	Weighted Voting	Borda Count	Sum Rule	Obs.-based M	Bayesian Model
1	0.68	0.68	0.64	0.66	0.67	0.67	0.69	0.71
2	0.77	0.78	0.65	0.81	0.76	0.79	0.78	0.80
3	0.76	0.60	0.54	0.62	0.67	0.63	0.73	0.75
4	0.76	0.73	0.59	0.77	0.78	0.78	0.72	0.79
5	0.59	0.47	0.34	0.57	0.58	0.60	0.40	0.47
6	0.81	0.72	0.65	0.79	0.80	0.80	0.71	0.76
7	0.62	0.68	0.49	0.68	0.71	0.64	0.66	0.62
8	0.77	0.64	0.52	0.80	0.72	0.77	0.68	0.74
Avr	0.72	0.66	0.55	0.71	0.71	0.71	0.67	0.70

Table 6. Test results for the first setting

Table 6 tabulates the performances of cooperation strategies as they are employed by the decision making agent by querying two agents. First, we see that although the decision making agent has no valid classifier to recognize any instance of 19 signs, it achieves equally successfully as the other two trained agents. Second, our results suggest that all cooperation strategies perform comparably, except the majority voting scheme. But in terms of teammate modeling strategies, we can still see the availability of prior information results in a slightly higher performance (Bayesian modeling is slightly better than Observation-based model). Actually, modeling approaches come with a cost, which is the cost of extra training data and the necessary interactions to build models of others. Given this cost, we can infer that voting and fusion methods are more preferable when compared to modeling in this particular setting.

In real settings, it is more likely that there will be more than two agents to query, and their predictive accuracy (reliability) in any given sign class is expectedly variant. In a second set of experiments we run our cooperation strategies on a more realistic setting to see the performance of our cooperation strategies and whether modeling pays off. For this purpose, this time in addition to a single decision making agent, and two trained agents (among which the dataset distributed as described for the first experiments), two random agents and two

Subjects	Majority V	Weighted V	Borda C	Sum Rule	Obs-based M	Bayesian M
1	0.17	0.17	0.80	0.74	0.94	1.0
2	0.15	0.14	0.85	0.80	0.95	1.0
3	0.14	0.15	0.88	0.79	0.89	0.98
4	0.17	0.15	0.87	0.78	0.94	0.99
5	0.07	0.12	0.80	0.71	0.79	1.0
6	0.17	0.17	0.87	0.79	0.94	1.0
7	0.08	0.14	0.88	0.73	0.77	0.96
8	0.15	0.13	0.86	0.77	0.94	1.0
Avr	0.14	0.14	0.85	0.76	0.89	0.99

Table 7. Test results for the second setting

semi-oracle agents are also involved. A random agent responds to a given query by one of 19 classes randomly. It also generates second or more predictions, as well as corresponding scores and certainty values again randomly. In contrast, a semi-oracle agent can recognize a set of sign classes perfectly, whereas it performs just like any random agent for the rest of the signs. In our setting, the first semi-oracle is perfect in the first ten signs, whereas the second semi-oracle is perfect in the remaining nine signs.

Note that our second setting is different from the first setting in that there are significant number of agents in the system who are not reliable at all for a given sign. In Table 7, since in majority voting scheme we blindly count the votes, majority voting ends up with a poor performance due to votes coming from unreliable agents. The situation is the same for weighted voting as well.

Interestingly, Borda count and score fusion by sum rule achieve relatively better performance. In Borda count, since each agent responds with its top three choices, and since the higher a sign class in a rank the higher its counted score, the effect of random agents is less, whereas the correct predictions (both from semi-oracles and trained agents) are reinforced due to scoring. In the light of our results, we can say that Borda count scheme is more robust to unreliable sources of information when compared to other common voting schemes. See [11] for a detailed discussion on robustness of the Borda count and its variants.

A similar effect is also observed for score fusion by sum rule. In this case, again each agent reports its top three choices alongside with its corresponding scores. The difference from Borda count is that the scoring is calculated by each agent on its own. Since the higher a sign class is in a rank, the higher its score, this method can also eliminate unreliable information and continue with more reliable ones.

The best performing strategies are teammate modeling methods, namely Observation-based and Bayesian modeling, in which the decision making agent explicitly models other agents in terms of their predictive accuracy for all sign classes. The explicit modeling enables the decision maker to avoid unreliable responses, and only consider information coming from reliable sources. For instance, in the Observation-based model, the decision maker is more likely to decide an instance in one of top ten signs following the first semi-oracle agent,

and more likely to decide as the second semi-oracle for an instance from the last nine signs. Bayesian model brings it further by including the prior, and achieves to reveal the oracle for most of the time. We conclude that it definitely pays off to explicitly model others in the system, if not every agent in the system are comparably reliable, and there exists unreliable ones.

To examine further effects of the size of the system, and information distribution among agents, we test our methods in a third set of experiments. In this case, in addition to one decision making agent, and two trained agents, there are four random agents and four quarter-oracles. A quarter-oracle perfectly recognizes one fourth of 19 signs, in particular the first quarter-oracle is perfect in sign classes between 1 and 5, the second is perfect in 6 to 10, the third is perfect in 11 to 15, and the fourth quarter-oracle is perfect in 16 to 19. Quarter-oracles respond randomly for the signs in which they are not perfect.

Subjects	Borda	C Sum	Rule	Observation-based	M Bayesian	M
1	0.77	0.61		0.93		0.85
2	0.72	0.58		0.98		0.96
3	0.73	0.60		0.97		0.95
4	0.77	0.58		0.96		0.95
5	0.69	0.53		0.81		0.82
6	0.79	0.59		0.95		0.89
7	0.69	0.57		0.81		0.82
8	0.71	0.52		0.93		1.0
Avr	0.73	0.57		0.92		0.91

Table 8. Test results for the third setting

This time, we only compare Borda count and score fusion by sum rule with Observation-based modeling and Bayesian modeling. The results in Table 8 show that it becomes more and more preferable to pay the cost of modeling as the size of the system increases and variations in the reliability of agents increase.

5 Discovering sign language properties

Languages have inherent properties that are useful to discover. One such property is the existence of dialects. The term dialect corresponds to a specific form of a language that belongs to a particular geographic region or social group [12]. Sign language contains many dialects. Although dialects are mostly due to geographical separation, dialects—in terms of particular signs—can as well be observed among different signers in the same region. Before continuing, to avoid ambiguity, we consider dialects at the level of a sign, and if there are dialects for a particular sign, then it means this particular sign is signed in at least two different ways.

Since dialects are very common in sign language, it is an important question for linguists to understand the nature of dialects. Actually the first step in the

study of dialects is to discover them. Here we propose and evaluate an automated method to discover dialects using the models generated by agents in Observation-based and Bayesian modeling strategies.

First we assume that for a particular sign, an agent only knows one way of signing it, even if the sign has dialects. Now consider a sign with two different dialects. Since an agent, which can recognize one dialect, cannot recognize the other dialect, we can separate the set of agents in the system in two in terms of which dialect they recognize. More formally, the set $Dial_1$ has the agents that can recognize the first dialect, and the set $Dial_2$ contains the agents that can only recognize the second dialect. Due to our assumption that $Dial_1$ and $Dial_2$ are separate, if one can find such disjoint sets of agents in terms of recognizing one particular sign class, it can be inferred that this sign has at least two different dialects known in the system.

A way to extract such patterns of sets is to use models generated by agents for Observation-based and Bayesian modeling. We can generate set of reliable agents for each sign using a threshold reliability value out of model of a decision making agent. Formally, T_i^c is the set of reliable agents for the sign class c for the decision making agent $i \in A_D$, the set of decision making agents. If the sign c has only one dialect known in the system, and each decision making agent $i \in A_D$ models the same set of other agents A_Q , then we expect $\cap_{i \in A_D} T_i^c \neq \phi$. But on the other hand, if there are at least two dialects of the sign c known in the system, then it follows that $\cap_{i \in A_D} T_i^c = \phi$.

To experimentally test our method for discovering dialects in the system, we run an additional fourth set of experiments. In this case, there are two decision making agents, two trained agents, two random agents and two semi-oracle agents. In addition to the 19 signs, we include a 20th sign, which has two different dialects in the system. One trained agent is trained for the first dialect and one semi-oracle is perfect in the first dialect. On the other hand, the other trained agent is trained for the second dialect and the other semi-oracle is perfect in the second dialect.

For each decision making agent, any agent i is in the set of reliable agents for a particular sign if the reliability of that agent i for that particular sign is greater than a threshold value, which we set as 0.6. For all 20 signs, we find the cardinality of the intersection of set of reliable agents for the decision making agents 1 and 2, $T_1^c \cap T_2^c$, where $1 \leq c \leq 20$. We observe that this cardinality is 0 only for the sign 20. Therefore we infer that the sign 20 has at least two dialects known in the system.

6 Discussion

We introduced a cooperative multiagent system for sign language tutoring. Each agent in our architecture represents a SignTutor [1] that can improve itself due to experience and by exchanging decisions with each other. On this architecture, we proposed several cooperation strategies that differ on whom to request experiences from and how to combine incoming answers.

As a cooperation strategy, firstly, we adopted several voting methods (majority voting, weighted voting, and Borda count), which are widely and successfully used in combining classifiers. Secondly, we applied score fusion by sum rule, which has proven useful in many pattern recognition problems. And lastly, we proposed two teammate modeling methods. Observation-based modeling strategy explicitly accounts for the reliability of other agents in the system using previous on purpose communication (On purpose in the sense that they interact to model each other using a portion of training data). The other teammate modeling method, the Bayesian modeling strategy, incorporates the available a priori information in order to model others better.

At first, we observed that the decision making agent performs comparably for all cooperation strategies in a toy setting, where there are two agents that are more or less equally reliable. But in more realistic settings, where there are more agents and they are not equally reliable, modeling strategies—although they are costly when compared to other strategies—perform the best. Therefore, for the decision making agent, it pays off to model others when there are agents with unknown reliabilities. We also showed that by analysis of the set of reliables of different agents, we can discover dialects of a sign known in the system.

Yolum and Singh [13] show that agents by modeling and updating models of others can achieve trustworthy service selection. In their model, agents model others in the system and direct their queries accordingly. Similar to our model, a model of an agent has several components, and they are updated depending on the interaction with that particular agent. But, rather than a statistical model, they assign a value between 0 and 1 for each component for each agent, and updating these values corresponds to an increment or a decrement over them.

To make agents cooperate more effectively, Chalkiadakis and Boutilier [14] employ Bayesian learning in order to better model others in the system and predict their future behavior more accurately. They show that Bayesian agents achieve better coordination in stochastic environments when compared to fictitious play, which is similar to our Observation-based model, and several previously proposed heuristic strategies.

Parker [15] also uses confusion matrix of a classifier as prior information to combine rankings coming from several non-homogeneous learners. He assumes that the confusion matrix of a classifier is a predictor for this classifier's future behavior, and he uses these behavior predictions in combining and achieves better error rates. In this study, predictions about future behavior of classifiers are done only upon the prior, but, in contrary to our modeling approaches, there neither exists an explicit statistical model nor the prior is updated due to observations.

Currently, we are collecting a larger database of human performances for a larger collection of signs in Turkish sign language. Therefore, we will have the chance to test our cooperation strategies also on this upcoming database. We are also working on other possible priors in addition to the confusion matrix. Lastly, we are searching for other properties of sign languages and methods to discover these properties on our architecture.

Acknowledgement

This work has been partially supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grants 107E021 and 105E073.

References

1. Aran, O., Ari, I., Akarun, L., Sankur, B., Benoit, A., Caplier, A., Campr, P., Carrillo, A., Fanard, F.: Signtutor: An interactive system for sign language tutoring. *IEEE MultiMedia* **16**(1) (2009) 81–93
2. Wooldridge, M., Jennings, N.R.: *Intelligent agents: Theory and practice*. Knowledge Engineering Review **10**(2) (1995) 115–152
3. Kuncheva, L.: *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience (2004)
4. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems* **11**(3) (2005) 387–434
5. Kittler, J.: Combining classifiers: A theoretical framework. *Pattern Analysis & Applications* **1**(1) (1998) 18–27
6. Melvin, I., Weston, J., Leslie, C.S., Noble, W.S.: Combining classifiers for improved classification of proteins from sequence or structure. *BMC Bioinformatics* **9**(1) (2008) 389
7. Weiss, G.: *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press (1999)
8. Aran, O., Burger, T., Caplier, A., Akarun, L.: A belief-based sequential fusion approach for fusing manual signs and non-manual signals. *Pattern Recognition* (2008)
9. Gokberk, B., Akarun, L.: Comparative analysis of decision-level fusion algorithms for 3D face recognition. In: *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*. Volume 3. (2006) 1018–1021
10. Robinson, J.: An iterative method of solving a game. *Annals of Mathematics* (1951) 296–301
11. Van, M., Erp, Schomaker, L.: Variants of the borda count method for combining ranked classifier hypotheses. In: *in the Seventh International Workshop on Frontiers in Handwriting Recognition*. (2000) 443–452
12. Jewell, E.J., Abate, F.: *The New Oxford American Dictionary*. Oxford University Press (2001)
13. Yolum, P., Singh, M.P.: Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man and Cybernetics, Part A* **35**(3) (2005) 396–407
14. Chalkiadakis, G., Boutilier, C.: Coordination in multiagent reinforcement learning: A bayesian approach. In: *Proceedings of the 2nd AAMAS*. (2003) 709–716
15. Parker, J.R.: Rank and response combination from confusion matrix data. *Information Fusion* **2**(2) (2001) 113–120