COOPERATIVE SIGN LANGUAGE TUTORING: A MULTIAGENT APPROACH

by

İlker Yıldırım

B.S., Computer Engineering, Boğaziçi University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2009

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Pınar Yolum for her guidance and support during the last three years. I owe most of the ideas in this thesis to her able and insightful guidance. More than being a great advisor, she was always available when I got into trouble with my graduate school applications or anything else about my life. It is a privilege to be involved in her group, and I feel it more intense as I get closer to moving to the University of Rochester for my PhD studies.

I share many ideas with Lale Akarun and Oya Aran in this thesis. I would like to thank Lale Akarun for her exteremely helpful and fruitful discussions. I would like to thank Oya Aran for providing all of the SignTutor codes and being always available for my never ending questions.

I would like to thank Murat Saraçlar for his feedback and ideas which resulted in important contributions to this thesis.

I have surely benefited a lot from our work with Haluk Bingol apart from this thesis. I would like to thank him for his guidance and support throughout these last two years.

I would like to thank Suzan Usküdarlı for her friendly and mind opening discussions.

Several friends and colleagues made these last two years unforgettable for me. I would like to thank Akın Günay for his every time available great consultancy that I frequently applied every time I got confused about my what to dos and how to dos. I would like to thank Özgür Kafalı for showing me how to be calm in face of any problem. I shall not forget our discussions with Haşim Sak after which I always felt the rebel of my exhausted neurons. I would also like to thank Reyhan Aydoğan for her guidance that eased my adaptation to being a graduate student. I would like to thank all other

members of AI lab that I didn't name here for helping me to have a two year period of time that was full of fun.

I am by all means grateful to my family. In our daily phone calls, my family always encouraged me to always keep being motivated and positive in whatever I was doing. Their unconditional support and love made me to feel always comfortable with each and every issue I had in my life.

This work has been supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under gran 107E021 and Boğaziçi University Research Fund under grant BAP09A106P.

ABSTRACT

COOPERATIVE SIGN LANGUAGE TUTORING: A MULTIAGENT APPROACH

Sign language is the natural means of communication for the hearing-impaired. Sign languages are based on signs, which are a combination of hand gestures, facial expressions, and head movements. Teaching these visual languages to others is an important, but a difficult task. 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 her previous knowledge about the sign. The experts role can be imitated by an automatic system, which uses a training set as its knowledgebase 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 represent sign language experts, and they 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. We further study the robustness of our strategies. Our results also show that the best performing strategy is our proposal, the Bayesian modeling strategy. Further, through analysis of the multiagent system, we discover inherent properties of sign languages, such as the existence of dialects.

ÖZET

KOOPERATİF İŞARET DİLİ ÖĞRETİMİ: ÇOK ETMENLİ BİR YAKLAŞIM

İşaret dili duyma engellilerin doğal iletişim aracıdır. İşaret dili el hareketleri, yüz ifadesi ve kafa hareketlerinin bir karışımı olan işaretlere dayalıdır. Bu görsel dilleri başkalarına öğretmek zor fakat önemli bir iştir. İşaret dillerinin verimli öğrenilmesi bir uzmanın sıkça geri bildirimde bulunmasıyla mümkündür. Uzman kimse, gerçekleştirilen işareti izlemeli, ve işaret hakkındaki varolan bilgisine göre işaretin doğru gerçekleştirilip gerçekleştirilmediğine karar vermelidir. Uzmanların rolü, gerçekleştirilen işaret hakkında karar vermek için bir eğitim kümesini bilgi dağarcığı olarak kullanıp bir sınıflayıcı eğiten, otomatik bir sistem tarafından taklit edilebilir. Fakat, otomatik sistemin, yeterli eski tecrübesi olmadığı takdirde, vereceği kararlar isabetli olamayacaktır.

Bu problemi çözmek için etmenlerin işaret dili uzmanlarını temsil ettiği ve kendi aralarında işbirliği yaparak gerçekleştirilen bir işaretin doğru sınıfına karar verdikleri, çok etmenli bir mimari öneriyoruz. Geliştirilen mimari üzerinde çeşitli işbirliği stratejileri uyguluyor ve bunların performansını değişen koşullarda test ediyoruz. Daha sonra, bu stratejilerin ne kadar sağlam olduklarını inceliyoruz. Sonuçlarımıza göre en iyi performansı sergileyen strateji bizim önermiş olduğumuz Bayesçi modelleme stratejisidir. Bir sonraki adım olarak, çok etmenli sistem üzerinde gerçekleştirdiğimiz analizlerle, işaret dilinin içerdiği dialektlerin varlığı gibi bir takım özellikleri keşfediyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Sign language is the natural means of communication for the hearing-impaired. Sign 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 signs*. Teaching these visual languages to others is an important, but a difficult task. It is important, simply because as the number of non-native signers increase, hearing-impaired will communicate easier and achieve her potential better. It is as difficult, as it is important, because the learner needs to practice and receive feedback frequently. A person can improve her performance of signs only with frequent trials 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 (See Figure 1.1 for the SignTutor GUI). 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. We briefly explain how SignTutor works. SignTutor consists of several consecutive modules: hand and face detection, followed by analysis stage, and finally sign classification module.

To robustly detect hands and their movements, the performer is required to wear colorful gloves, and the system is trained for the colors of each of these gloves. Following training for glove colors, hand detection reduces only to detect pixels of the trained glove colors, and to construct a connected component of detected pixels for each hand. Following the detection of hands, hand motion analysis is done by estimating center of mass, position and velocity of each hand for each frame. However there are several

Figure 1.1. Sign Tutor GUI: a) Human learners can view videos recorded in the database for training. b) Performance of the human learner is recorded through a webcam, and then displayed. c) The system analyses the performance and provides feedback verbally as OK or $WRONG$. d) System can also provide an animated feedback.

sources of uncertainty due to segmentation noise and occlusion of hands, which make motion analysis difficult. To smoothly estimate trajectory of each hand, two independent Kalman filters are used which assume a constant velocity, ignoring acceleration. In addition to hand motion, head movements are also extracted using an algorithm inspired by the human visual system [1].

Once hand shape is detected in every frame of the camera input, it is necessary to extract features to proceed to the recognition of signs. SignTutor is intended to work with a single low resolution camera to facilitate its usage. This goal constrains the SignTutor only to extract low-level features (appearance based features such as best fitting ellipse, compactness, etc.), rather than high level features.

At the classification stage, HMMs are used to model each sign and classify according to the maximum likelihood principle. Outcomes of the two independent HMMs, $\text{HMM}_{Manual\&Nonmanual}$ and $\text{HMM}_{NonManual}$, are fused to arrive at the final decision. $\text{HMM}_{M\&N}$ uses both hand and head features, but it suffers from high dimensionality. Therefore, in another dedicated model, HMM_N, only head features are used. Firstly, both HMMs are trained independently using both hand and head cues, and only head cues respectively. Then the trained models are used for prediction by fusing their decisions in sequential manner. Each HMM is a continuous, four state, left-to-right model, and trained for each sign by the Baum-Welch algorithm.

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.

To address this challenge, we propose to encapsulate each instance of the SignTutor in an agent. An agent is a persistent computation (i.e. a software process) that can perceive, reason, act and communicate [2]. Agents usually represent some principal or some other entity (i.e. a stand alone software) and they need to cooperate in order to achieve their goals. These entities or principals may often have contradicting goals, beliefs, and so on. Hence, agents that are representing those principals may have contradicting goals or beliefs. In order to achieve coordination in such an environment of agents, specific cooperation mechanisms must be developed.

Agents can communicate, for instance they can request other agents to perform an action. Agents can reason based on their knowledge and the state of the world. And they adapt in the sense that they can learn to act differently according to changes in the outside world. All in all, agents are autonomous, they can act freely, choose their own actions, and go into interactions with others.

Such a strong sense of autonomy can only be achieved in a world of heterogeneous agents. These heterogeneous agents can be designed and implemented by different parties with different intentions, and agents do not need to expose their internal properties. Although they are heterogeneous they may still share some requirements to comply with. As a natural consequence of being autonomous and possibly living in an environment, which involves heterogeneous agents and many other objects, agents are adaptive. They can learn what and who is good for them and act accordingly. Agents are self-interested in the sense that they work for their owners.

A multiagent environment contains multiple agents, which can interact each other. Each agent's knowledge of the world is usually limited, i.e., an agent usually does not know everything, which are known to other agents in the environment [3]. One important challenge associated with multiagent systems is related with the communication among agents. Communication protocol among agents must be well designed such that it will allow agents to achieve their desired level of interaction. One other important challenge is the coordination in the multiagent system. Agents often have conflicting or shared goals, and they need to coordinate in order to achieve their aims.

If properly designed, agents can communicate and achieve coordination effectively, at the same time they exploit the inherent heterogeneity and autonomy [4].

Today's agent systems enable us to design and develop automated interactive systems either between machines, people or both. Real world application areas of multiagent systems range from space research to film production sector. For example, in a space research project called eSTAR (eSpace Telescope for Astronomical Research), intelligent agents are being used to predict very special events in the universe, such as massive supernova explosions.

In many other remarkable examples, multiagent systems are being used to represent complex and dynamic real world environments, which range from simulation of economies, societies to biological environments. For example, many studies have investigated the understanding of social structures such as trust, reputation, dependence, norms, and so on in human notion [5].

Another important application area is e-commerce, where automated negotiation systems work effectively. Agents, representing business parties, negotiate according to preferences of their owners. Furthermore, service providing agents can learn customer preferences and offer accordingly to achieve better selling rates. Another striking example of success of agent systems in economics context is autonomous trading agents. By now, autonomous trading agents outperform human commodity traders by more than 7% [6].

Hollywood is another practitioner of multiagent systems. Autonomous software agents are used to model individual combatants in Peter Jackson's triology The Lord of the Rings. Rather than manually or centrally controlling each combatant in each and every war scene, Peter Jackson and his team modeled each combatant as an autonomous agent which had its own beliefs, goals and so on. Then agents autonomously acted in the battlefield which resulted in unforgettable war scenes of the triology.

In our system, we represent each SignTutor by an agent. These representing agents are distributed geographically, but will be able to communicate with each other over the Internet, forming a cooperative multiagent system [7]. 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 thesis is organized as follows: Chapter 2 discusses possible architectures and formalizes the problem of identifying agents to cooperate with on our proposed cooperative multiagent architecture. Chapter 3 provides the necessary background for further reading. Chapter 4 explains different cooperation strategies that have been developed to help agents decide on sign classification. Chapter 5 evaluates these cooperation strategies on real sign language data. Moreover, Chapter 5 explains how important sign language properties can be inferred from a multiagent system and how robust our strategies are. Finally, Chapter 6 discusses our work with comparisons to the literature.

2. PROBLEM DEFINITION

Current SignTutor [1] is a stand-alone application. However, in reality the same application will need to be run in a geographically distributed environment with different data. This obviously calls for a distributed architecture. A possible architecture would be to have a collection of stand-alone applications in which there is no interaction between agents. In such a system although agents may improve their classification capabilities due to learning and experience, they will not be able pass their experiences to others and they will be not be able benefit from experiences of others.

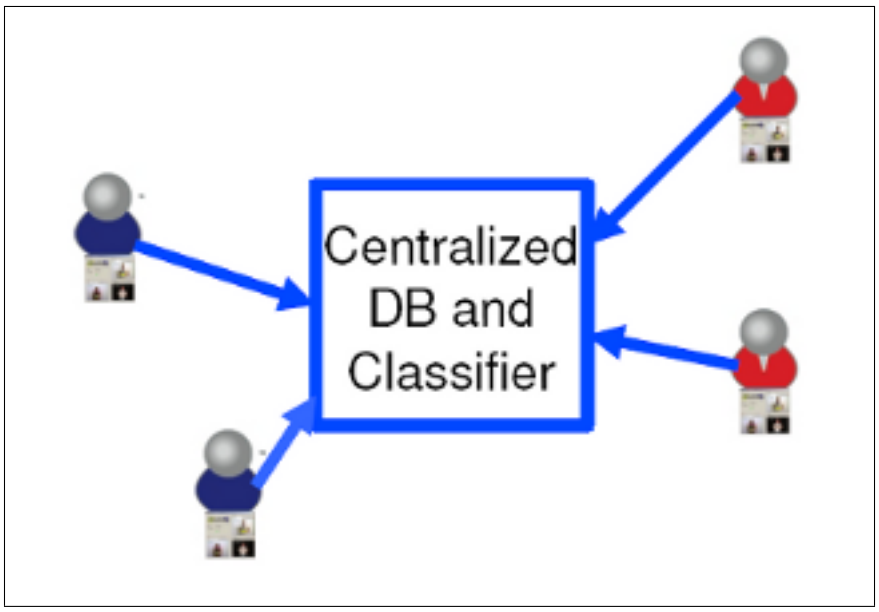

Figure 2.1. A centralized architecture

Another possible architecture would be a centralized one, in which agents are geographically separated, but they share a common database and a common classifier as seen in Figure 2.1. But in reality we know that agents may not be online all the time, hence they cannot access to the database and the classifier. In addition to that, videos may belong to a certain individual who do no want to share it.

Taking all these into account, we propose a cooperative multiagent system of SignTutors, which consist of many agents distributed geographically [7]. Each agent

represents a sign language tutor. Agents are connected via Internet. Each agent is associated a local database and a classifier. An agent can improve its classification performance due to its own experience. Bearing in mind that each agent has a trained classifier for sign recognition, agents may decide to include a sign performed by a human learner in its training data or a sign language teacher may add a new training data. Besides an agent's own classifier, agents can help each other classify signs by exchanging classification requests. For example, when an agent needs help in a classification request, it can request a subset of training data from other agents in the system to train its classifier better for the request. But, this method may obscure privacy concerns of a training data provider (a sign language learner or teacher). Another method could be that: the agent, rather than asking for training data of others, directly requests other agents in the system to classify the performance and only responds with its decisions. But, in this case, combining these decisions coming from other agents in the system may turn out to challenging.

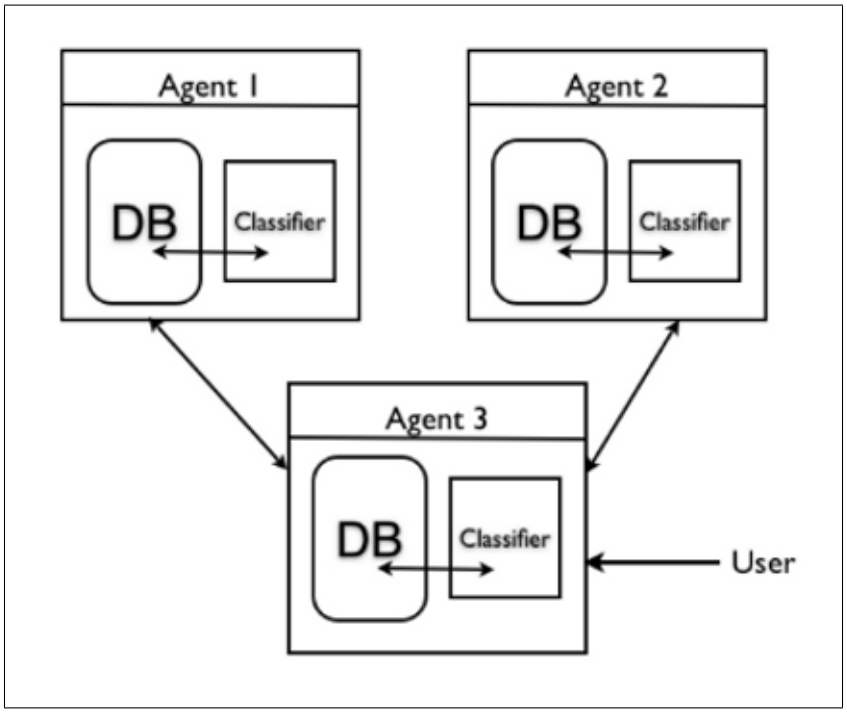

Figure 2.2. A setting for multiagent architecture for sign tutoring tools

To better understand possible challenges with our architecture, we will study a few examples before a formal definition of the problem. Figure 2.2 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. In our example setting in Figure 2.2, a user queries agent 3 for a particular sign, say "school", and requests to learn whether she performed the sign correctly or not. In this setting, we assume agent 3 is dummy, in the sense that it has no trained classifier for recognizing "school". But still agent 3 can make a plausible decision by querying other agents in the system, an obvious advantage of our architecture. Therefore, agent 3 goes ahead and query agents 1 and 2 in the system. The three examples are given below.

(i.) Agent 3 queries agents 1 and 2 for whether the performance of the user is an instance of sign "school". In this case, agents 1 and 2 can answer with one of two possible outcomes, OK or $WRONG$. For example if agent 1 believes that the most likely class for the human performance is "school", then it responds as OK, and otherwise as $WRONG$.

In this example, suppose that agent 1's response is OK, whereas agent 2 responds as $WRONG$. After collecting these responses from other agents, agent 3 must make a decision to give a feedback to the user. But, in this case, it is not obvious how to make a decision in favor of one of the two possibilities.

(ii.) Agents 1 and 2 may provide a ranking over the possible classes instead of a single outcome of OK or $WRONG$. For example, agent 1 may produce a ranking over all possible classes in terms of how likely the user performance is an instance of each of these classes. Agent 1 may believe that the user performance is most likely to be an instance of "university", then "teacher", and only after then "school".

Note that in the former example, agents respond for either OK or $WRONG$, and agent 3 seeks for the majority. But in this example, other agents in the system provide a ranking over the possible classes. In this case, more information is available to agent 3. But still, conflicting rankings coming from different agents make it difficult for the decision making agent. If on the other hand, agents 1 and 2 have agreed on a ranking, it would have been obvious for agent 3 to draw a decision.

(iii.) Although our example setting in Figure 2.2 only consists of three agents, suppose

that there are many more agents in the system. In this larger environment, agent 3 may collect either answers for OK and $WRONG$, or other agents in the system may provide a ranking or scores over the possible candidate classes.

In this example, we have many more agents in the system, relative to the other scenarios. In such an environment, it is likely that there are a lot of unreliable agents in terms of their responses for a given user performance. In this case, it may not result in accurate decisions to query everyone and combine their responses. Instead locating an expert in the system may be more useful. But, being aware of unreliable information sources, and finding an expert are challenging problems indeed.

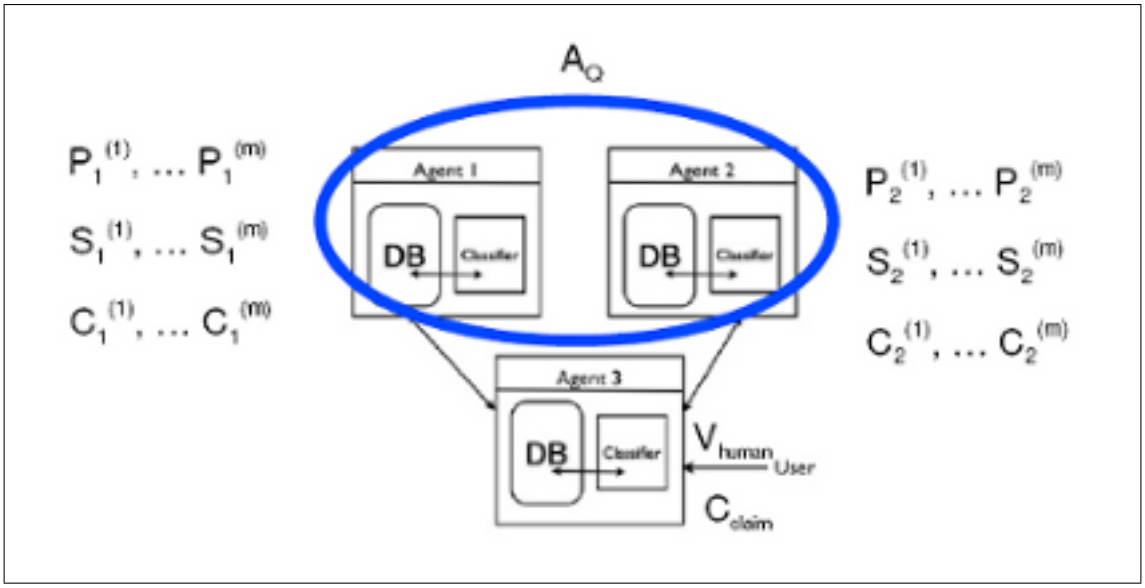

Figure 2.3. Problem definition

In this thesis, 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. After developing insights about nature of the problem, now we formalize it. Recall that 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, $P_i^{(1)}, ..., P_i^{(m)}$, with associated scores, $S_i^{(1)}, ..., S_i^{(m)}$ and certainty values, $C_i^{(1)}, ..., C_i^{(m)}$ 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. See Figure 2.3 for a sketch of the problem definition. 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. BACKGROUND

In this chapter, we introduce a few concepts that will ease reading of the rest of this thesis. First we introduce and span the literature on combining classifiers, and then study teammate modeling.

3.1. Combining Classifiers

It is worth visiting the problem again. We have a number of classifiers that possibly classify an item differently. Given all these classification results, how do we decide? The aim of combining classifiers is to achieve increased accuracy with the expense of computational burden. There are several reasons why a classifier ensemble might be better than a single classifier and Dietterich suggests three of them [8] as statistical, computational and representational.

Firstly, to understand statistical advantage, suppose that several classifiers and a labeled data set is available. Picking a single classifier to make predictions may result in large errors, because the classifier we picked may not be a suitable one for the given prediction problem. On the contrary, using several classifiers and averaging over their predictions will decrease or wipe out the risk of picking an unsuitable classifier, even though it may not be better than a single classifier.

Secondly, in terms of computational advantage, a classifier on its own may converge on a local optima. For instance, hill climbing may stay on a locally good solution due to its local optimization technique [9]. But, on the other hand, some way of combining these local optimas as an ensemble may take us closer to the global optima.

Lastly, in the representational aspect, we can benefit ensembles of classifiers as follows. Suggested classifiers for a problem may not contain the optimal classifier, i.e. the problem can be nonlinear, whereas we have all linear classifiers. In that case, we either increase complexity of a single classifier, i.e. having more weights to learn and

tune in a neural network, or we could work on how to combine these available classifiers. It is a lot likely that a complex classifier will converge to local optima, but in case of combination, we could avoid that local optima, and achieve the global optimum by an ensemble.

There are many successful methods proposed for classifier combination. These methods can be classified in terms of their approach to the combination. The first approach, which we will focus mostly, is to work at the level of designing different combiners. In this first approach, the big issue is to build up combiners that will exploit the possible advantages, i.e. statistical, mostly that the resulting ensemble is accurate as possible.

A second approach is related with using different base classifiers. In this case, the combining method is independent from the base classifiers, and aim is to design base classifiers such that their combination will result in higher accuracy, i.e. bagging and boosting. In bagging (bootstrap aggregating) [10] small training sets are generated using the the original data, and several base classifiers are trained using each of these different training data sets. Then decisions of the base classifiers are averaged to make the final decision. The success of the ensemble changes according to the set of base classifiers chosen. Similarly, in boosting, several weak classifiers are combined to construct a strong classifier. The idea is to reweight data items according to whether they are classified correct or wrong by a weak classifier. Among others, AdaBoost, short for adaptive boosting, is the most popular boosting method [11].

Other levels of combining classifiers are at feature level and data level. At feature level classifiers are trained with different feature subsets, the important question is how to select different features. Lastly, at data level, classifiers are trained with different data sets to increase accuracy again. Bagging and boosting methods can also be thought as data level combination methods, because each classifier is trained with a different subset of data.

As background, we will explore three methods, which are instances of the first

approach, in that they are different in their designs for combining several classifiers. The methods that we cover here with different designs are voting, mixture of experts and score level fusion by sum rule.

Another categorization of classifier combination methods is based on the level of available classifier output as follows:

- *The Abstract level:* In this category, only the label of the decided class is available, and nothing is known about the certainty of the classifier. As it is in the majority voting, the classifiers provide, which class they think is most likely, and all votes are aggregated accordingly.
- *The Rank level:* At this level, classifiers provide a sorted list of class labels according to themselves, where the most preferred class is at the top and the least preferred at the bottom. The lists that are provided can be partial (i.e., only a subset of classes are sorted) or full (i.e., all classes are in the ranking). For instance, in Borda count, classifiers rank their top n candidates, and candidates at a higher rank receive higher scores while aggregating all ranks.
- *The Measurement level:* At measurement level, classifiers also provide their measures of their decisions. For instance, in weighted voting, classifiers' votes are counted according their level of certainty that they have provided. Furthermore, in sum rule, agents provide their scores for all or a subset of classes (in terms of how similar the data item is to each of classes).

3.1.1. Voting and Sum Rule

Majority voting and sum rule are the most common and effective ways of combining classifiers [12, 13]. Suppose that there are N learners (or agents that represent learners), and the problem is to assign a given data item to one of Ω classes by combining decisions of learners. Each learner i responds with a vector of size Ω, $d_{i,1}, ..., d_{i,\Omega}$, and $d_{i,\omega} \in \{0, 1\}$ for all $i \in 1...N$ and $\omega \in 1...\Omega$. After collecting responses of all

learners the ensemble classifier is built as follows.

$$e_\omega = \sum_{i=1}^{N} g_{i,\omega} d_{i,\omega} \tag{3.1}$$

where $g_{i,\omega}$ is a factor assigned by the learner i as its weight or ranking or score for the class ω. For instance, in weighted voting, $g_{i,\omega}$ is the weight or certainty of learner i in class ω.

In majority voting all $d_{i,\omega}$ is set to 0 except the decided class d_{i,ω_k}, which is set to 1. And each $g_{i,\omega}$ is set to 1, for all learners $i \in 1...N$ and classes $\omega \in 1, ..., \Omega$, so all learners are equally weighted. In terms of level of input available for combining, majority voting is an abstract level combining method, because learners only provide the class label, which they decided. Actually in majority voting the number of classes to vote for is two and the class, which collects more than half of the votes wins. On the other hand, in plurality voting, a variant of majority voting, there can be more than two classes, and the winner is the class with the most number of votes.

Borda count is a rank level combining method: learners provide a ranking over their decisions. Ranking is provided in $g_{i,\omega}$ s, with the highest value for the top ranking class ω_k and lower values for the proceeding rankings, and eventually 0s for the rest of classes. For instance, if learners only rank their top three choices, then g_{i,ω_k} is three, and two other corresponding g_{i,ω_x} and g_{i,ω_y} are set to two and one, and the rest is set to zero.

In weighted voting, similar to majority voting the decided class, d_{i,ω_k}, is 1, and the rest is zero, but this time each learner provides its certainty, therefore the corresponding g_{i,ω_k} is the weight of the voter i. The underlying intuition is that not everyone is as knowledgeable as others for all data items or different classes. The resulting ensemble is more likely to be greater at the classes that are decided by more weighted learners. In terms of level of input available for combining, weighted voting is a measurement level combiner, because learners provide their measures of their decisions.

In addition to voting methods we have discussed, another popular and effective method of combining classifiers is the sum rule. This adds the scores, which are provided by the learners. Each learner scores each class in terms of how similar it considers a given data item to this class. In terms of our notation introduced above, each $d_{i,\omega}$ is set to 1, and each $g_{i,\omega}$ is the calculated score, for all learners $i \in 1...N$ and for all classes $\omega \in 1...\Omega$. Sum rule is also a measurement level combiner, since all learners provide their self calculated scores.

Once the ensemble e_ω is built, we choose the maximum a posteriori class

$$\omega^* = \arg\max_{\omega} e_\omega. \tag{3.2}$$

Choosing the maximum a posteriori class is very intuitive. For instance, in plurality voting, it corresponds to choosing the class with the maximum number of votes. Furthermore, the maximum a posteriori class of a sum rule ensemble corresponds to the class with the overall maximum aggregated score.

Lam and Suen motivate majority voting due to its successful applications in the literature and at the same time its being simple. They then analyze majority voting formally (rather than experimentally) to understand its behavior under several conditions. Their theoretical analyses predict the experimental data well, and they conclude majority voting is a simple yet as effective as other rather complicated methods [14, 15]. When each learner in an ensemble, independently from others, is more likely to decide correctly rather than incorrectly (error probability less than 0.5), majority voting is shown to perform better and better with the increasing size of learners.

Lin *et al.* study theoretical analysis of plurality voting, in which the class, which receives most of the votes is the winner (Note the difference with respect to majority voting, in which the winning class is the one with more than half of the votes). They show that plurality voting results in better error/rejection rates in several real world problems when compared to majority voting [16].

Ruta and Gabrys bring analyses further and show that structuring classifiers into relevant multistage organizations can decrease error rates of majority voting [17]. They also analyse sensitivity of these better error bounds and show that they are robust by introducing random discrepancies into votes of learners.

A critical point in combining classifiers using majority voting (and also many other combining methods) is deciding the nature and the quantity of classifiers that will be combined. When there are several classifiers available,it is usually intractable to select the optimum set of classifiers, because the problem of finding the optimum set requires an exhaustive search over all possible combinations. Therefore many computationally simple, but suboptimum methods have been suggested for selecting the classifiers to combine. Using a large set of classifiers and data sets, Ruta and Gabrys experimentally show that it is more appropriate to select classifiers based on combiner error (the error rate under a specific combiner), rather than diversity among selected classifiers [18].

In combining classifiers, either the combined classifiers use the same type of pattern representations, or each use a distinct representation, i.e. combining k-nn classifiers with different parameters, versus classifiers using its own input representation. Kittler *et al.* propose a common theoretical framework for combining classifiers each having its own way of pattern representation [12]. Besides showing that many popular combining classifier methods can be obtained using this theoretical framework, they analyse these methods experimentally, and reach an interesting result. Sum rule, the method obtained under most restrictive assumptions achieve the best. Furthermore, they verify their experimental results, that is sum rule is better, by employing theoretical sensitivity analysis on estimation errors of these combining methods.

In an early work on combination of classifiers, Ho *et al.* propose several rank level combining methods and study their performance on several real data sets [19]. The methods they propose are two fold: reduction (i.e., using intersection of ranks) of classes and reranking the classes (i.e., using Borda count). They conclude that combining methods can significantly decrease error rates, and more work is necessary

to understand the dynamics of combining classifiers.

Van Erp and Schomaker study Borda count, and two variants of it, median rule and Nanson's Borda elimination. They use artificial data to analyse abilities of Borda count, for example ranking list of a learner is simulated as a random permutation of the true ranking such that the response is not severely wrong, but not fully correct as well. After extensive analysis, they conclude that Borda count is strong if there are many ranking errors, but it fails to reconstruct the true ranking when there is a limited but large ranking errors [20].

Kittler and Alkoot compare two widely used combining methods, namely majority voting and sum rule [21]. They analytically show that sum rule outperforms voting when estimation error of each classifier is independent and comes from a Gaussian distribution. But on the other hand, if the error distribution has heavy tails (that is a significant probability mass being on the tails of the distribution), voting may overcome the sum rule. They represent their analytical results on synthetic data. They also show that real data approve their general findings, but also may be in contrary due to the independence and being identically distributed assumptions made in the theoretical analysis.

3.1.2. Mixture of Experts

Up until here, in voting and sum rule, votes are associated with static weights, and they are usually assigned by the classifier itself. For instance, in weighted voting, classifier i provides a certainty value for class ω, $g_{i,\omega}$ and certainty assignment by the classifier i is never reinforced according to the success of any trial.

But in the mixture of experts, proposed by Jacobs *et al.*, the factor $g_{i,\omega}$ is a function of the input [22]. In mixture of experts setting, there are several experts (classifiers) and one gating expert. The gating expert is responsible to evaluate weights of other experts for combining their decisions.

One crucial aspect of the mixture of experts is that the gating expert and other experts (classifiers) are all trained in parallel and in a coupled manner. The coupled training is designed in such a way that in order to maximize the likelihood of the mixture (which is the aim of training), each expert has to compete and not overlap with others in its expertise. Therefore, each expert specialize on separate local regions on the input space. This is achieved by the cost function, which dictates and results in a competing and non-overlapping mixture of experts. Once the gating expert has weights, $g_{i,\omega}$, according to input data, it can calculate the resulting combined decision in the same as in voting and sum rule:

$$e_\omega = \sum_{i=1}^{N} g_{i,\omega} d_{i,\omega} \tag{3.3}$$

Alpaydin and Jordan show that it is especially important that the experts are coupled during training, which results in faster and yet accurate convergence behavior [23].

3.2. Teammate Modeling

Cooperative multiagent systems are systems in which multiple agents attempt to solve a problem or maximize utility jointly through their interaction. Since there are several agents, the complexity of the problem to solve or utility maximization often become complex, that requires agents to learn autonomously. The approach to learning in cooperative multiagent learning is usually concurrent, in other words, each agent simultaneously learns to solve the problem or search for the optimum in realm of other agents in the system. A common method in concurrent learning is teammate modeling, in which each agent models others in the system in order to better predict their future behavior and act accordingly.

Boutilier [24] and Chalkiadakis and Boutilier [25] employ Bayesian agents for updating models of other agents. Bayesian agents use their models to predict most likely a posteriori behavior of other agents in the system. Then they act according to their predictions to achieve better coordination in stochastic environments. They

show that Bayesian agents achieve coordination and cooperate better than several other teammate modeling proposals.

In teammate modeling, one crucial aspect is that, the agents that are being modeled are as well modeling in turn, which results in an infinite recursion. For instance, "Agent A is taking action X, because it thinks that agent B thinks that agent A thinks that...". Vidal and Durfee [26] categorize agents in terms of the level of recursion they employ in modeling others. A 0-level agent have no model of others at all, in other words, agents are not aware of others. On the other hand a 1-level agent models others as 0-level agents. In general, a N-level agent models other agents as (N-1)-level agents. Since N is finite the infinite recursion is avoided and agents can decide in a finite amount of time.

In open multiagent systems, to achieve their goals, agents need to interact with many other agents. An important challenge in such dynamic systems is to select which agents to trust. Kafalı and Yolum study several modeling strategies, where the aim is to accurately model others in the system in order to decide whom to trust [27]. In their proposals each agent in the system updates its models of others after every interaction it is involved. These updates are done according to feedback (payoff, outcome) received by the agent. A positive outcome reinforces the current model, whereas a negative feedback requires agents to find other partners to trust.

In another work, Kafalı and Yolum propose modeling environment in terms of actions of agents and effects of these actions [28]. Such an approach in modeling and updating models of others to figure out whom to trust is quite different from previous methods, because instead of modeling agents one by one, this model propose to model actions and effects directly. Their results suggest that in certain conditions, action-based modeling outperforms traditional methods not only in terms of level of accuracy but also in terms of speed in achieving accurate models.

An important dimension in locating a trusted agent (i.e., a service, a partner) is the available reputation information. Reputation systems are used to encourage

adherence to commitments in open multiagent systems (i.e., e-commerce Web sites such as eBay.com). Josang and Ismail propose a reputation system which is based on beta probability density functions [29]. On contrary to most existing reputation systems, it is theoretically founded, and yet simple and flexible. By employing statistical methods, the reputation information can easily be updated by feedback and transformed into intuitive reputation ratings.

Teacy *et al.* propose a trust and reputation system called TRAVOS (Trust and Reputation in the Context of Inaccurate Information Sources) [30]. In TRAVOS, trust is calculated using probability theory, where ingredients are past interactions with an agent, and reputation information gathered from third parties (reputation is necessary when there is an absence of past interaction). TRAVOS enables agents to compare how trustworthy others are by providing trust metrics to agents. It enables agents also to have a level of confidence in their trust metrics, so that agents can make more informed decisions (i.e., a non-confidently trusted agent may be as bad as a confidently untrusted agent). TRAVOS also provide agents to have accuracy measures over the reputation information provided by third parties, with which they can normalize information coming from third parties accordingly.

Şensoy *et al.* propose a context-aware service selection method, POYRAZ, which works accurately even under deception [31]. Since a base level ontology is shared among agents, they can communicate their experiences as if they are objective. For example, if an agent is happy with a slow but high quality service, that information will also be known to other agents. Enabling communication of experiences is a step forward to traditional rank based mechanisms, since there is no ontological information (semantic information). Furthermore, POYRAZ integrates a mechanism to filter out deceptive information, which is common is dynamic open systems. Authors experimentally show that POYRAZ outperform the state-of-the-art trust and reputation methods even when there are a significant number of liars and agents preferences (satisfaction criteria) differ significantly.

Beta Reputation System [29], TRAVOS [30], and POYRAZ [31] all assume a sig-

nificant number of feedback obtained from previous interactions with providers (training data). In all these models, the immediate availability of feedback is exteremely vital for proper calculation of trust in providers. But in our case, this is not possible due to the relatively very high cost of training data. Therefore, we need to investigate a different solution.

4. COOPERATION STRATEGIES

In our context, cooperation strategies are expected to enable each agent to achieve better tutoring, which corresponds to increased predictive accuracy in recognizing performances of human learners. As discussed above in Chapter 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 4.1 and 4.2, we further elaborate state-of-the-art classifier combinations methods [13].

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 [3]. In our case, each agent in our system, by communicating predictions for human performances in between, 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 incorporating prior knowledge to build a model, and the other merely using observations coming from interactions. The details of these methods are explained in section 4.3.

4.1. Voting

Voting is a common method for combining classifiers, and has proven useful many times in the literature [32, 33]. 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 2.2, 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.

4.1.1. Majority Voting

Majority voting is the application of majority rule, which selects one of the two choices with more than half of the votes to make a decision [34]. 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 subset of agents that are queried, which, in this case, are 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.

4.1.2. 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.

$$R = \sum_{i \in A_Q} K * C_i^{(1)} \tag{4.1}$$

where

$$K = \begin{cases} 1 & \text{if } P_i^{(1)} = C_{claim} \\ -1 & \text{if } P_i^{(1)} \neq C_{claim} \end{cases} \tag{4.2}$$

and

$$\text{Final Decision} = \begin{cases} OK & \text{if } R \geq 0 \\ WRONG & \text{if } R < 0 \end{cases} \tag{4.3}$$

One problem associated with weighted voting is the assessment of weights. As a heuristic, 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, $P_i^{(2)}$.

4.1.3. 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 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, $P_i^{(1)}, ..., P_i^{(k)}$ 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.

4.2. Score Level Fusion

In score level fusion, instead of the predictions themselves, the scores (confidences, likelihoods, etc.) of the prediction, coming from different experts, are fused to give the final decision [35, 36] . 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, $P_i^{(1)}, ..., P_i^{(k)}$, and their associated scores, $S_i^{(1)}, ..., S_i^{(k)}$ for the top k choices for all agents $i \in A_Q$. For instance, in our sample setting in Figure 2.2, 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 $P_i^{(1)}, ..., P_i^{(k)}$ and $S_i^{(1)}, ..., S_i^{(k)}$ for all $i \in A_Q$, score fusion by sum rule calculates new integrated score for each sign class as follows:

$$sumScores(P_i^{(j)}) = sumScores(P_i^{(j)}) + S_i^{(j)} \text{for all } i \in A_Q \text{ and } 1 \leq j \leq k, \qquad (4.4)$$

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 $WRONG$.

4.3. 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 agents' 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]

4.3.1. Observation-based Model

In multiagent games, a simple model for modeling other agents in the system to predict their future behavior and achieve 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 [37].

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

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 training data sets with each other 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 collect 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:

$$R_i^c = \frac{TP}{TP + FA} \tag{4.5}$$

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). [2]

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:

$$Value(P_i^{(1)}) = Value(P_i^{(1)}) + R_i^{P_i^{(1)}} \tag{4.6}$$

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 how many agents predicted that particular sign class. The agent makes its decision as the position of the maximum in $Value$.

[2] We normalize R_i^c s before using, therefore the modeling agent has a discrete probability distribution over the reliability of agents for each class.

To illustrate Observation-based modeling strategy we study an example on our sample setting as seen in Figure 2.2, 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 4.1 shows reliability values calculated by agent 3 according to Equation 4.5. 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.

Table 4.1. Reliability values of both agents for three classes before normalization.

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

Values in Table 4.2 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 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 $0.62 > 0.57$ agent 1's value).

Table 4.2. Reliability values of both agents for three classes after normalization.

Reliability	Agent 1	Agent 2
Study	0.67	0.33
Study regularly	0.38	0.62
Study continuously	0.57	0.63

4.3.2. 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 problem, 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 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.[3] 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 $Conf_1, ..., Conf_\kappa, ..., Conf_n$ be disjoint confusion sets, such that $S = Conf_1 \cup ... \cup Conf_\kappa \cup ... \cup Conf_n$. Then, we say that any instance of sign s in $Conf_\kappa$ is more likely to be confused with another instance 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 class of signs are confused with each other as follows: Let $Conf_\kappa = s_1, ..., s_j, ..., s_c$, and let $R_i^{(s_j)}$ be the reliability of agent i in class s_j as described in Section 4.3.1. We calculate the posterior reliability of agent i in sign class

[3] 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.

s_j as follows:

$$PR_i^{s_j} = R_i^{s_j} - \frac{\sum_{s_h \in Conf_\kappa}(1 - R_i^{s_h})}{|Conf_\kappa|} \tag{4.7}$$

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 $Conf_\kappa$. [4]

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 know illustrate how it works by following our example above at the end of Observation-based model, Section 4.3.1. Firstly, notice that the three classes we selected, "Study", "Study regularly", and "Study continuously" all belong to the same confusion set. Table 4.3 shows posterior reliability values calculated by agent 3 according to Equation 4.7 and Table 4.1.

Values in Table 4.4 are obtained after normalization of posterior reliability values among 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

[4] For calculating $PR_i^{s_j}$ we use unnormalized $R_i^{s_j}$ values. But we then normalize posterior reliability values, therefore the modeling agent has a discrete probability distribution over posterior reliability of other agents for each class, as it is in the Observation-based model.

Table 4.3. Posterior reliability values of both agents for three classes before normalization.

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

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 $0.63 > 0.56$ agent 2's value).

Table 4.4. Posterior reliability values of both agents for three classes after normalization.

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

5. EVALUATION

We evaluate our proposed architecture in several ways: First, we study the performance and robustness of our cooperation strategies under different settings. Then, we study our architecture in terms of aiding the study of the sign language itself. We particularly focus on discovery of sign language properties.

5.1. Performance 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 performed our analysis in several experiments. All results reported are average of five runs for each experiment (See Table 5.1 for a summary of each experimental setting).

Table 5.1. Summary of experimental settings, see text for details.

	# of				
Setting	Trained agents	Random agents	Semi-oracles	Quarter-oracles	Total
One	2	-	-	-	3
Two	2	2	2	-	7
Three	2	4	-	4	11
Four	4	2	-	-	7

Before discussing our experiments and their results, we stop here to describe our environment. We developed our agents in MATLAB, which is actually not common among multiagent people. The reason we choose it as follows. As we have described, each agent in our system represents a SignTutor [1], which is developed in MATLAB.

Table 5.2. Test results for the first setting

Subjects	Agent 1	Agent 2	Majority V	Weighted V	Borda C	Sum rule	Observation-based M	Bayesian M
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

Although we could develop our agents in other platforms, in order to facilitate communication of the representing agent and the SignTutor, we developed our agents also in MATLAB.

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 5.1 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.

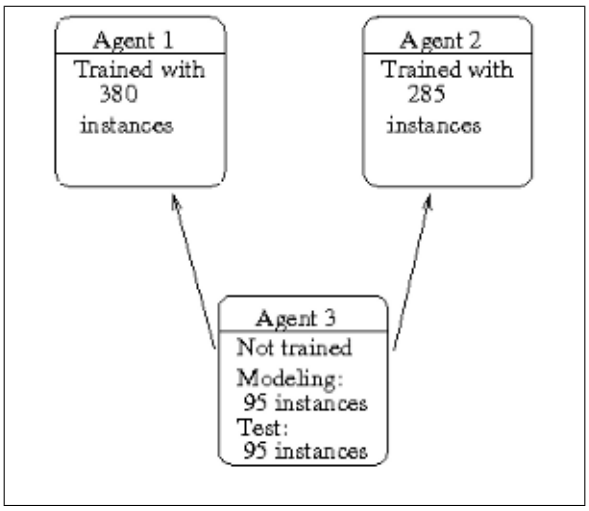

Figure 5.1. Sketch of the first experimental setting.

In Table 5.2 we see the performance of cooperation strategies as they are employed by the decision making agent by querying two agents. On the average, agent 1 predicts correctly 72% of the time, whereas agent 2 performs 66% correctly. 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 (Weighted voting 71%, Borda count 71%, Sum rule 71%, Observation-based modeling 67%, and Bayesian modeling 70%, all average of eight cross-fold validation). Second, our results suggest that all cooperation strategies perform comparably, except the majority voting scheme which performs correct for 55%. The reason behind the poor performance of the Majority voting is that ties are resolved randomly, and since there are two voters, many ties have occured. But in terms of teammate modeling strategies, we can still see the availability of prior information results in a slightly higher performance (Bayesian modeling (70%, on the average) is slightly better than Observation-based model (67%, on the average)). 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 semi-oracle agents are also involved (See Figure 5.2 for a sketch of the second setting). 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 10 signs, whereas the second semi-oracle is perfect in the remaining 9 signs.

Table 5.3. Test results for the second setting

Subjects	Majority V	Weighted V	Borda C	Sum Rule	Observation-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

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 5.3, we see that majority voting and weighted voting performs correct only for 14% on the average. Since in the majority voting scheme we blindly count the votes, it ends up with such 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 (Borda count 85%, and sum rule 76% on the average). 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

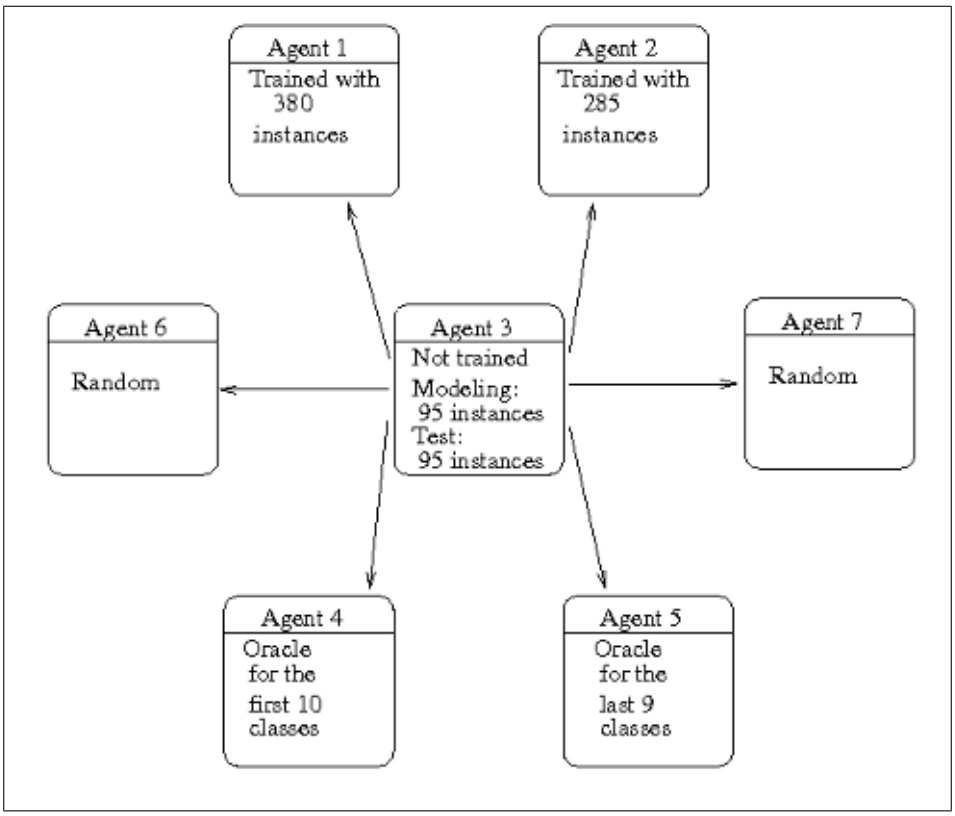

Figure 5.2. Sketch of the second experimental setting.

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 [20] 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 (89%, on the average) and Bayesian modeling (99%, on the average), 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 the top 10 signs following the first semi-oracle agent, and more likely to decide as the second semi-oracle for an instance from the last 9 signs. Bayesian model improves this 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 (See Figure 5.3 for a sketch of the third setting).

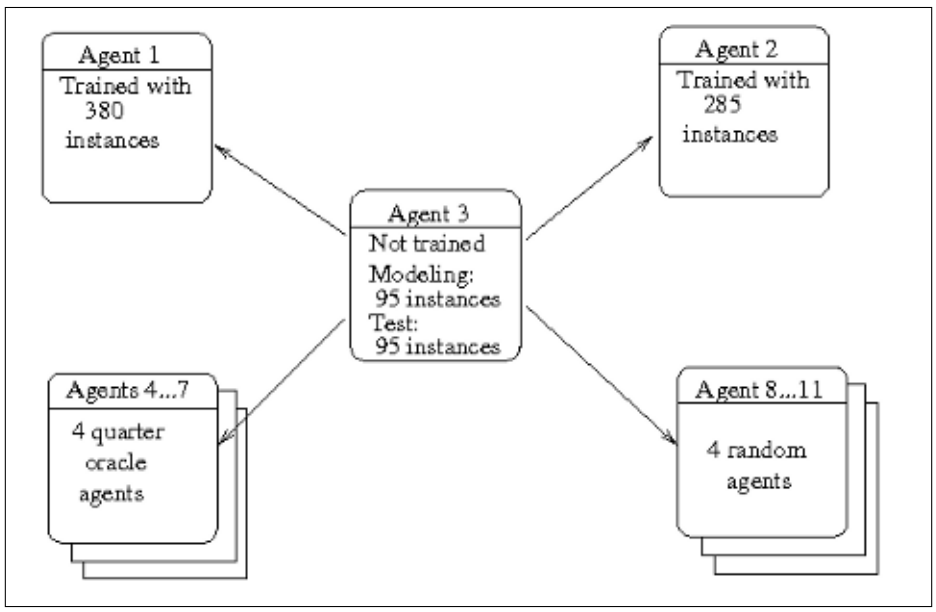

Figure 5.3. Sketch of the third experimental setting.

This time, we only compare Borda count and score fusion by sum rule with Observation-based modeling and Bayesian modeling. In Table 5.4 we see that Borda

count achieves a 73%, whereas sum rule only performs 57% correct. On the other hand Observation-based modeling and Bayesian modeling perform 92% and 91% respectively. These results 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.

Table 5.4. Test results for the third setting

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

Note that oracle agents are perfect in recognition of a subset of all sign classes. A more realistic substitute for oracles is an agent trained with more data. In a fourth set of experiments we replaced the semi-oracle agents in setting two by two agents that are trained with 665 instances. But the agent replaced the first semi-oracle is only trained for the first 10 sign classes, whereas the agent replaced for the second semi-oracle is trained for the last 9 instances. See Figure 5.4 for the depiction of this setting.

In Table 5.5, we see the performance of our cooperation strategies when there are agents trained with more data instead of semi-oracles. In Table 5.5, agent 4 is the agent trained with 665 instances for the first 10 sign classes, whereas agent 5 is trained with the same instances but for the last 9 sign classes. Since agents 4 and 5 respond randomly to the signs that they are not trained for, their average performance is relatively small (46% and 39%, respectively on the average). Although not seen in the

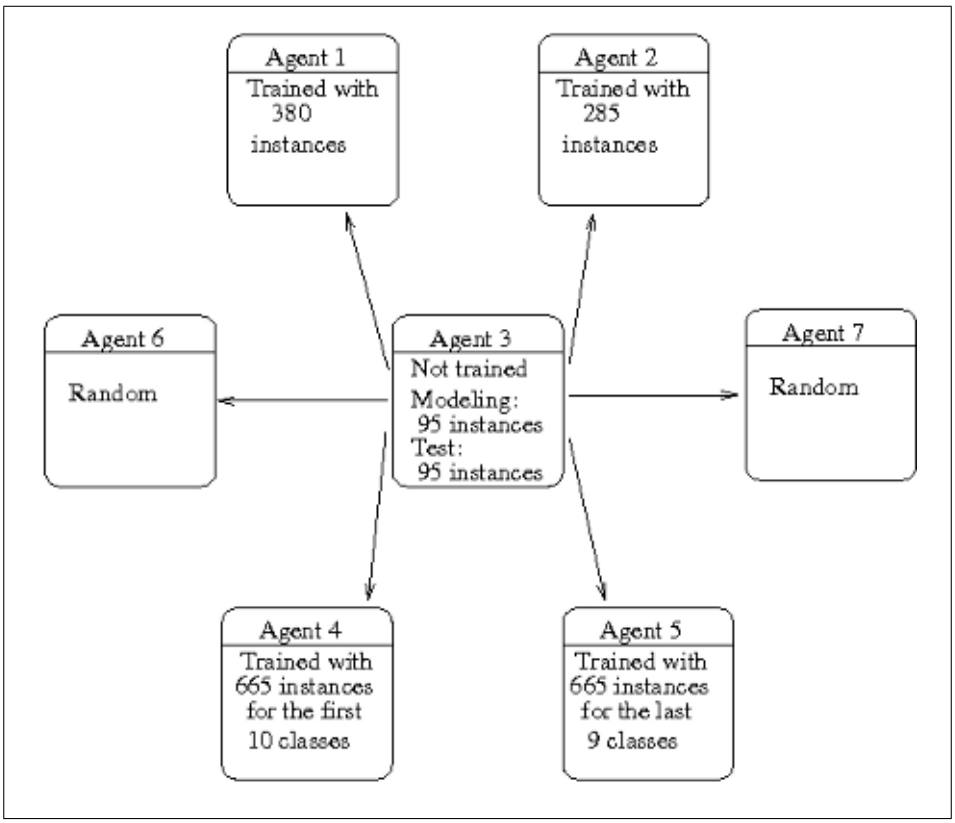

Figure 5.4. Sketch of the fourth experimental setting.

table, if we would only consider agent 4's performance for the first 10 signs, and agent 5's performance only for the last 9 sign classes then the overall average performance is 75%. We observe that Borda count (69%) again performs better than the sum rule (57%). But again, modeling strategies outperform all the rest. Even further, the Bayesian modeling strategy obtains the most accurate predictions available in the system, and achieves a performance as high as the overall performance of agents 4 and 5, which is 75%.

In sign language recognition, similar to face recognition, training data is scarce. Therefore it is important to study how the performance of modeling methods change with decreasing data available for modeling others.

To observe performance of our modeling methods, we run additional experiments using the first experimental setting. Recall that there are two agents and one decision making agent in the first setting. Previously the decision making agent used 95 instances to model the other two agents, which consists five repetitions for each of 19

Table 5.5. Test results for the fourth setting

Subjects	Agent 4	Agent 5	Borda C	Sum rule	Observation-based M	Bayesian M
1	0.48	0.35	0.66	0.54	0.70	0.72
2	0.43	0.48	0.79	0.67	0.83	0.80
3	0.53	0.40	0.67	0.57	0.78	0.80
4	0.46	0.47	0.75	0.56	0.83	0.82
5	0.34	0.35	0.54	0.53	0.52	0.57
6	0.52	0.38	0.77	0.60	0.76	0.83
7	0.42	0.36	0.62	0.56	0.65	0.60
8	0.55	0.38	0.74	0.58	0.78	0.84
Avr	0.46	0.39	0.69	0.57	0.73	0.75

different classes. This time we gradually decrease the number of instances to train models of other by one repetition for each class. Therefore, at the beginning there are 95 instances, but then 76, 57, 38, and 19 (one repetition for each class) instances are available only.

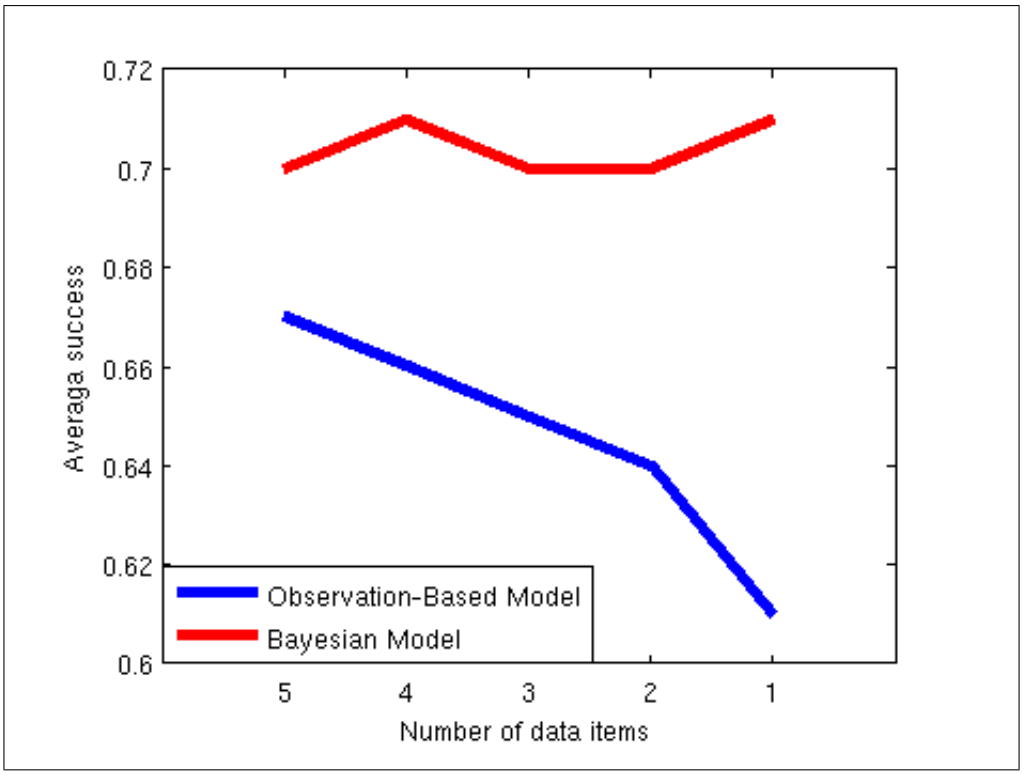

Figure 5.5. Average success of Observation-based and Bayesian models when there are different amounts of data available for modeling other

In Figure 5.5 we see number of repetitions for each sign class on the horizontal axis, and the average performance of Observation-based and Bayesian modeling

strategies again over eight-fold cross-validation. Expectedly, the performance of the Observation-based model decreases as the amount of data available to model others decrease, because there is no other information for building models, but only observations. But, to our surprise, even though there is very limited amount of data for modeling, Bayesian modeling strategy's success is nearly constant, with just small changes. Incorporating the prior information in Bayesian model, by regulating the available observations, achieves to maintain its success regardless of the amount of data used. Therefore, we can say that, in case of availability of very limited data, although Observation-based model gets worse, Bayesian model still stands as a very good option for cooperation.

These results suggest that Bayesian modeling strategy works well even there is very scarce data due to the prior knowledge available. Note that, in the Bayesian modeling strategy, we provide the decision making agent with the prior. So, it is crucial to evaluate performance of the Bayesian modeling strategy when the available prior is distorted, not the full prior is available.

To understand behavior of the Bayesian modeling strategy, we distort the prior in a few stages and observe the average success of the strategy. Recall that the Bayesian modeling strategy incorporates the prior information, which says that several subsets of sign classes are very similar in each other, and an agent is more likely to confuse the class of a sign with another class which is similar to the real class (Section 4.3.2).

To produce distorted priors we run a merge operation over confusion sets. A merge operation is as follows. Select two confusion sets at random and merge them into one confusion set. In a consecutive merge operation, we select one more confusion set at random, and merge this confusion set to previously merged sets. Therefore, after two merge operations, there is one confusion set which is a merging of three confusion sets, and other non-merged confusion sets.

In Figure 5.6, we see the effect of distortion of prior on the average success of Bayesian modeling strategy for the second and third experimental settings. In

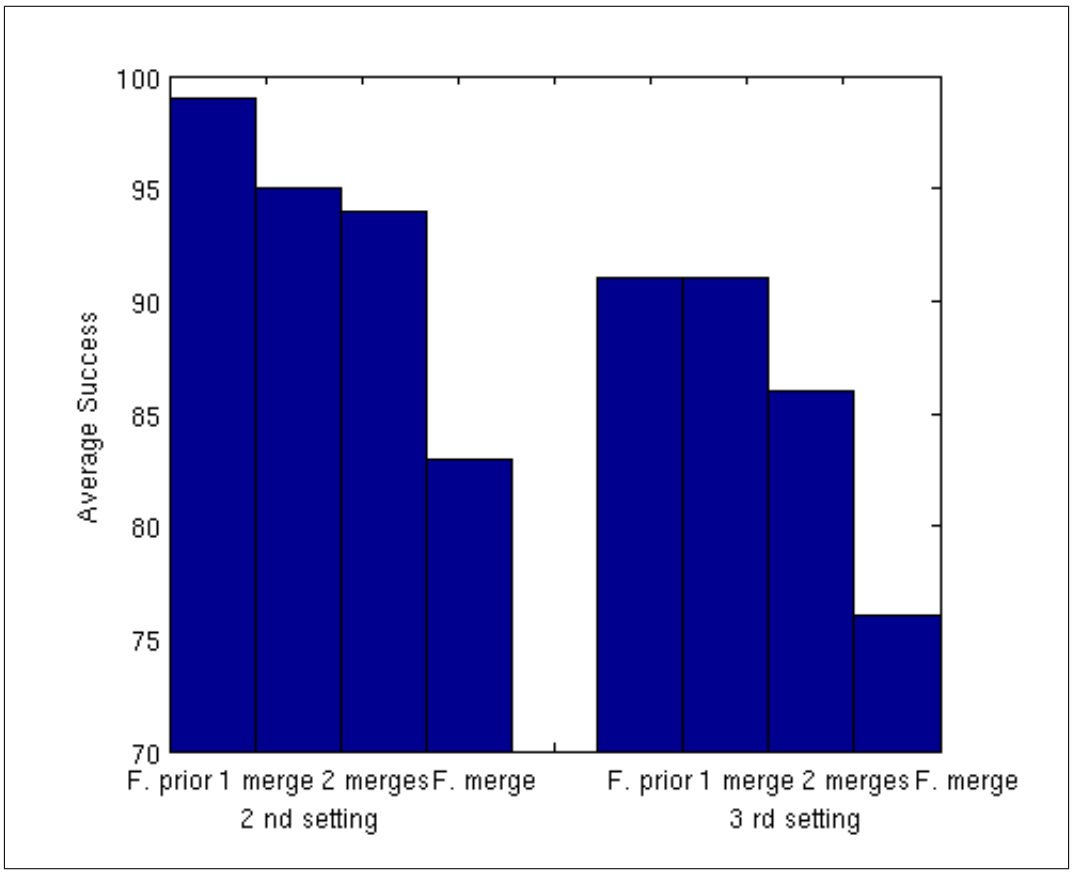

Figure 5.6. Effect of distortion of prior on the success of the Bayesian modeling strategy. Bars show the performance of Bayesian modeling when the prior information is full (F. prior), one time merged (1 merge), two times merged (2 merges), or fully merged (F. merge).

the second setting, expectedly success of the strategy decreases for consecutive merge operations. When full prior is available Bayesian modeling achieves 99%, but as the prior gets more distorted, the success of the strategy also decreases to 95% and then 94%.

Interestingly, for the third experimental setting, as seen in Figure 5.6, the performance of the strategy only decreases with the second merge operation. Recall that in the third setting, there are four quarter oracles. Due to sharing of knowledge of signs among quarter oracles, some oracles may not behave as it is expected by the prior. For instance, although prior predicts that an agent will perform similarly for sign classes five, six and seven, the first quarter oracle in the third setting is only perfect at fifth class, and random at sixth and seventh. Such distribution of quarter oracles causes

the observed average success in the first merge operation. This sharing is also the underlying reason that the Bayesian modeling strategy is slightly less successful than the Observation-based model in the third setting as seen in Table 5.4.

In addition to one and two merge operations we also consider a worst case scenario, where the decision making agent believes that all signs are in the same confusion set. In Figure 5.6, this is the full merge case, where all confusion sets are merged into one set. When the available prior information is fully merged, the decision making agent can still perform comparably with that of Borda count, and even better under the third setting (Borda count performs 73%, whereas Bayesian modeling with fully merged prior performs 76% on the average).

5.2. Robustness

Multiagent systems can either be open or closed. In contrary to a closed one, agents can arbitrarily enter or leave in an open multiagent system. In a closed multiagent system, the set of agents are predetermined beforehand and it does not change through out the life of the system. Since open systems allow entrance or exit of agents, we call open systems dynamic systems, and closed systems static systems.

Although we discussed our cooperation strategies and results on static systems, they can easily be extended to dynamic systems. Here we discuss these challenges and how our architecture would accommodate them.

One of these challenges is the entrance of a new agent into the system. There are two problems associated with this. On one hand, the new agent must model others in the system in order cooperate them, and on the other hand, already existing agents may model the new agent to possibly query for a future decision making task. The former problem can be solved by making agent query other agents in the system with data for modeling, and then continue just like any other already existing agent. Although the latter problem is not as crucial as the former one, a possible solution is to make agents model a querying agent, if this is the first time they are queried by this agent. This

solution is valid, because a new agent will query others to model them, and therefore already existing agents will become aware of that agent as being queried for the first time, they will query in turn to model this new agent.

Table 5.6. Leaving of an agent from the system

Subjects	Majority V	Weighted V	Borda C	Sum Rule	Observation-based M	Bayesian M
1	0.02	0.13	0.79	0.69	0.94	1.0
2	0.04	0.06	0.83	0.75	0.95	1.0
3	0.01	0.08	0.83	0.73	0.92	1.0
4	0.01	0.09	0.85	0.74	0.97	0.99
5	0	0.06	0.75	0.63	0.79	1.0
6	0.03	0.12	0.84	0.75	0.94	1.0
7	0.01	0.05	0.84	0.72	0.75	0.89
8	0.02	0.05	0.80	0.72	0.93	1.0
Avr	0.02	0.08	0.82	0.71	0.89	0.99

To experimentally evaluate how performance of each cooperation strategy changes by entrance of a new agent, we work on the second experimental setting. In the second setting, there are two semi oracles, 2 trained agents, and 2 random agents. In addition to these six agents we added a seventh random agent. Results in Table 5.6 show that majority voting and weighted voting get even worse as new unreliable agents enter into the system (with performances 2% and 8% respectively). On the average, performances of both Borda count and sum rule decrease from 85% and 76% to 82% and 71% respectively, by the entrance of a new agent. But on the other hand, both Observation-based and Bayesian modeling strategies maintain their success without any change (Observation-based at 89%, Bayesian modeling at 99%). This is due to the modeling of others in the system. Since modeling strategies update their models for newly coming agent, they can count responses from that agent accordingly, and they maintain to make better final decision.

A second challenge is the leaving of an agent. Just making the leaving agent notify others in the system before exiting the system will enable others to be aware of the update. The remaining agents can alter their models to overcome this change. To experimentally evaluate how performance of each cooperation strategy changes by entrance of a new agent, we again work on the second experimental setting. This time,

we remove one of the random agents, therefore the number of agents is reduced to five. According to results in Table 5.7, modeling strategies maintain their success as an agent leaves the system. (Observation-based modeling 91%, and Bayesian modeling 99%). As an unreliable agent leaves the system sources of unreliable information decreases in the system. Accordingly, we observe that Borda count (from 85% to 88%) and the sum rule (from 76% to 83%) achieve better predictions. Furthermore, although majority voting fail to increase its success, weighted voting jumps to 38%, which is still unacceptably poor performance.

Table 5.7. Entrance of a new agent to the system

Subjects	Majority V	Weighted V	Borda C	Sum Rule	Observation-based M	Bayesian M
1	0.13	0.40	0.84	0.81	0.94	1.0
2	0.14	0.44	0.91	0.87	0.96	1.0
3	0.09	0.37	0.89	0.84	0.94	1.0
4	0.11	0.40	0.91	0.83	0.97	0.99
5	0.04	0.25	0.82	0.78	0.81	1.0
6	0.13	0.43	0.92	0.86	0.95	1.0
7	0.07	0.33	0.88	0.81	0.77	0.91
8	0.11	0.42	0.91	0.84	0.94	1.0
Avr	0.10	0.38	0.88	0.83	0.91	0.99

When Tables 5.6 and 5.7 are both considered, we can conclude that teammate modeling strategies are robust to entrance or leaving of an agent. But voting and sum rule methods just reflect any change in the multiagent environment, and we account these results to their being not robust.

Lastly, although we have argued that agents may be improving or worsening in time we did not discuss how to update models of agents accordingly. It is necessary to keep models of others up to date by periodically querying them using data for modeling. Although the solution seems simple there is a tradeoff right here. It is costly to query other agents to update their models, because training data is scarce. But it is possible that a previously reliable agent starts malfunctioning. The optimum way of resolving this tradeoff remains as a challenge for us.

5.3. 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 [38]. 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 dialects. Since an agent, which can recognize one dialect cannot recognize the other, 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. And due to our assumption $Dial_1$ 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 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 a set of reliable agents for each sign using a threshold reliability value out of a 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. And 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. Whereas 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 evaluated 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 observed 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. CONCLUSION

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 and they 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 experimentally confirmed that when there are unreliable sources of information majority voting performs bad [14, 13], but on the contrary Borda count remains to be a good combiner. Hence Borda count is a more robust way of combining classifiers.

Although it is worthy to pay the cost of modeling others in many settings, we further analyzed how performance of modeling strategies change if there is less data available for modeling purpose (in other words smaller cost to pay for modeling).

As we expected Observation-based model's performance decreased as data lessened. But, surprisingly, Bayesian modeling's performance remained constant with a little increase. Therefore, even there is very small amount of data available for modeling others, Bayesian modeling maintains its performance due to prior information it employs.

Additionally, we analyzed how the performance of Bayesian model changes with distorted, i.e. not full, prior. Expectedly the average success of the Bayesian modeling strategy decreases, but it still performs comparably when compared to Observation-based model.

We also showed that our teammate modeling strategies are easily and robustly extendable to dynamic systems where agents can enter or leave the system. But, on the contrary, voting and sum rule methods just reflect any change in the multiagent environment, and we account these results to their being not robust.

We also showed that by analysis of the set of reliable partners of different agents, we can discover dialects of a sign known in the system. This way of discovering dialects is an important contribution, because sign language dialects carry a high value for linguistics researchers, and we can automatically mine dialects for them.

Having discussed our results, let's revisit the examples in Chapter 2 to better understand where we have arrived. In Example **??**, the decision making agent has to decide between two choices, none of which is majority. We showed that the decision making agent can resolve either by randomly selecting one of the choices or it can employ a modeling strategy to favor the choice which is coming from a more reliable resource. In Example **??**, the decision making agent need to combine conflicting rankings or scores coming from different agents. We showed that under Borda count scheme and sum rule, agents can effectively and relatively robustly arrive at good decisions. In Example **??**, the decision making agent has to draw an accurate decision out of many unreliable sources of information. Our experiments show that, despite failure of other strategies, modeling strategies, specifically Bayesian modeling strategy, exhibit impressive performance even under significant amount of unreliable information.

Recall that modeling is costly, it requires extra effort (modeling data and interactions). With every arrival of a new agent, the decision making agent, which uses a modeling strategy, has to decide whether to model this new agent or not. Either the decision making agent models a new agent and query it in the future if it is reliable, or just ignore this new agent and rely on the agents that it has already modeled. Therefore, in terms of scalability, the decision making agent is confronted with an exploration vs exploitation tradeoff. The decision making agent either pays the cost of modeling new agents (explore more) or just query according to its current models (exploit what you have). One solution could be to set a threshold on the minimum reliability a decision making agent seeks. And until it achieves this threshold in its models, the agent keeps exploring new arrivals, but once it achieves, then the agent is satisfied with its models and it exploits what it has in hand, and ignore new agents entering to the system.

In our architecture, we assume all agents respond according to their internal classifiers whenever they are queried. But, in certain conditions, it may not be reasonable for agents to respond to each and every query, in other words, agents may decide to reject responding to an incoming query. For instance, suppose an agent A is modeled to be reliable for all signs by all agents in the system. In this case, agent A will be very busy with answering others' queries, and therefore agent A will be unavailable for the users waiting for feedback from it. In such a case, it is more reasonable for agent A to reject incoming queries. Although we do not consider this problem explicitly, we can enable our agents to opt for rejection whenever necessary as follows. An agent that is very busy can make a response with a certainty level of zero, which will mean that this agent rejects to respond, or it is incompetent to have a positive certainty value for the given query. The querying agent can infer that its request is rejected just by checking up the corresponding reliability of the responding agent.

In most trust models, where modeling of others in the system is a crucial aspect, feedback out of any action of the agent is immediately available to that agent [27, 28, 29, 30]. In all these models, the immediate availability of feedback is exteremely vital for proper calculation of trust. On the contrary, in our Observation-based and Bayesian modeling strategies, agents can estimate reliability of other agents in the system, but

still update their models when any sorts of feedback is available to them.

Yolum and Singh [39] 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 [25] 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 pcoordination in stochastic environments when compared to fictitious play, which is similar to our Observation-based model, and several previously proposed heuristic strategies.

Parker [40] 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. And lastly, we are searching for other properties of sign languages and methods to discover these properties on our architecture. These are interesting topics that we defer to future work.

REFERENCES

1. Aran, O., I. Ari, L. Akarun, B. Sankur, A. Benoit, A. Caplier, P. Campr, A. Carrillo, and F. Fanard, "Signtutor: An interactive system for sign language tutoring", *IEEE MultiMedia*, Vol. 16, No. 1, pp. 81–93, 2009.

2. Singh, M. and M. Huhns, *Service-oriented computing: semantics, processes, agents*, Wiley, 2005.

3. Panait, L. and S. Luke, "Cooperative multi-agent learning: The state of the art", *Autonomous Agents and Multi-Agent Systems*, Vol. 11, No. 3, pp. 387–434, 2005.

4. Claus, C. and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems", *Proceedings of the National Conference on Artificial Intelligence*, pp. 746–752, 1998.

5. Yıldırım, I. and P. Yolum, "Hybrid Models for Achieving and Maintaining Collaborative Symbiotic Groups", *The Fifth Conference of the European Social Simulation Association*, 2008.

6. Kephart, J., "Software agents and the route to the information economy", *Proceedings of the National Academy of Sciences*, Vol. 99, No. 90003, pp. 7207–7213, 2002.

7. Wooldridge, M. and N. R. Jennings, "Intelligent Agents: Theory and Practice", *Knowledge Engineering Review*, Vol. 10, No. 2, pp. 115–152, 1995.

8. Dietterich, T., "Ensemble methods in machine learning", *Lecture Notes in Computer Science*, pp. 1–15, 2000.

9. Russell, S. and P. Norvig, *Artificial intelligence: a modern approach*, Prentice Hall Englewood Cliffs, NJ, 1995.

10. Breiman, L., "Bagging predictors", *Machine learning*, Vol. 24, No. 2, pp. 123–140, 1996.

11. Freund, Y. and R. Schapire, "Experiments with a new boosting algorithm", *International Conference on Machine Learning*, pp. 148–156, 1996.

12. Kittler, J., M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 226–239, 1998.

13. Kuncheva, L., *Combining pattern classifiers: methods and algorithms*, Wiley-Interscience, 2004.

14. Lam, L. and C. Suen, "A theoretical analysis of the application of majority voting topattern recognition", *Pattern Recognition, Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, Vol. 2, 1994.

15. Lam, L. and S. Suen, "Application of majority voting to pattern recognition: an analysisof its behavior and performance", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 27, No. 5, pp. 553–568, 1997.

16. Lin, X., S. Yacoub, J. Burns, and S. Simske, "Performance analysis of pattern classifier combination by plurality voting", *Pattern Recognition Letters*, Vol. 24, No. 12, pp. 1959–1969, 2003.

17. Ruta, D. and B. Gabrys, "A theoretical analysis of the limits of majority voting errors for multiple classifier systems", *Pattern Analysis & Applications*, Vol. 5, No. 4, pp. 333–350, 2002.

18. Ruta, D. and B. Gabrys, "Classifier selection for majority voting", *Information fusion*, Vol. 6, No. 1, pp. 63–81, 2005.

19. Ho, T., J. Hull, and S. Srihari, "Decision combination in multiple classifier systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16,

No. 1, pp. 66–75, 1994.

20. Erp, M. V. and L. Schomaker, "Variants of the Borda Count Method for Combining Ranked Classifier Hypotheses", *in the Seventh International Workshop on Frontiers in Handwriting Recognition.*, pp. 443–452, 2000.

21. Kittler, J. and F. Alkoot, "Sum versus vote fusion in multiple classifier systems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 1, pp. 110–115, 2003.

22. Jacobs, R., M. Jordan, S. Nowlan, and G. Hinton, "Adaptive mixtures of local experts", *Neural computation*, Vol. 3, No. 1, pp. 79–87, 1991.

23. Alpaydin, E. and M. Jordan, "Local linear perceptrons for classification", *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, pp. 788–794, 1996.

24. Boutilier, C., "Learning conventions in multiagent stochastic domains using likelihood estimates", *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 106–114, 1996.

25. Chalkiadakis, G. and C. Boutilier, "Coordination in multiagent reinforcement learning: A bayesian approach", *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 709–716, ACM New York, NY, USA, 2003.

26. Vidal, J. and E. Durfee, "Agents learning about agents: A framework and analysis", *Working Notes of the AAAI-97 workshop on Multiagent Learning*, Vol. 7176, 1997.

27. Kafalı, O. and P. Yolum, "Trust strategies for ART Testbed", *Ninth International Workshop on Trust in Agent Societies, AAMAS*, pp. 43–49, 2006.

28. Kafalı, O. and P. Yolum, "Action-Based Environment Modeling for Maintaining Trust", *Trust in Agent Societies: 11th International Workshop*, pp. 81–98,

Springer, 2009.

29. Jøsang, A. and R. Ismail, "The beta reputation system", *Proceedings of the 15th Bled Electronic Commerce Conference*, pp. 324–337, 2002.

30. Teacy, W., J. Patel, N. Jennings, and M. Luck, "TRAVOS: Trust and reputation in the context of inaccurate information sources", *Autonomous Agents and Multi-Agent Systems*, Vol. 12, No. 2, pp. 183–198, 2006.

31. Şensoy, M., J. Zhang, P. Yolum, and R. Cohen, "POYRAZ: Context-aware service selection under deception", *Computational Intelligence, accepted*, 2009.

32. Kittler, J., "Combining classifiers: A theoretical framework", *Pattern Analysis & Applications*, Vol. 1, No. 1, pp. 18–27, 1998.

33. Melvin, I., J. Weston, C. S. Leslie, and W. S. Noble, "Combining classifiers for improved classification of proteins from sequence or structure", *BMC Bioinformatics*, Vol. 9, No. 1, p. 389, 2008.

34. Weiss, G., *Multiagent systems: a modern approach to distributed artificial intelligence*, MIT press, 1999.

35. Aran, O., T. Burger, A. Caplier, and L. Akarun, "A belief-based sequential fusion approach for fusing manual signs and non-manual signals", *Pattern Recognition*, Vol. 42, No. 5, pp. 812–822, 2009.

36. Gokberk, B. and L. Akarun, "Comparative Analysis of Decision-level Fusion Algorithms for 3D Face Recognition", *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, Vol. 3, pp. 1018–1021, 2006.

37. Robinson, J., "An iterative method of solving a game", *Annals of Mathematics*, pp. 296–301, 1951.

38. Jewell, E. J. and F. Abate, *The New Oxford American Dictionary*, Oxford Univer-

sity Press, 2001.

39. Yolum, P. and M. P. Singh, "Engineering self-organizing referral networks for trustworthy service selection", *IEEE Transactions on Systems, Man and Cybernetics, Part A*, Vol. 35, No. 3, pp. 396–407, 2005.

40. Parker, J. R., "Rank and response combination from confusion matrix data", *Information Fusion*, Vol. 2, No. 2, pp. 113–120, 2001.