Creating of a Crowdsourcing Application to determine the Semantics of Shapes

Ruben Oostinga
r.j.s.oostinga@student.utwente.nl

ABSTRACT

This paper describes the creation of an online crowdsourcing application to determine the semantics of shapes by making a game. The game is played by two players who are both presented with the same image of a shape and the same question about that shape. The players have to answer the questions and get points based on the similarity of the answers. The questions have multiple types. The answers submitted are stored and can be analysed using statistical methods. This research can be used as a framework to making similar applications.

Keywords
crowdsourcing, semantics of shapes, image labelling, rating semantics

1. INTRODUCTION

1.1 Crowdsourcing

The word crowdsourcing was coined by Jeff Howe in a June 2006 Wired magazine article [5]. It describes a model for problem solving or production using a crowd of people. The problem or assignment is broadcasted to a group of people. Some of the people within the crowd submit a solution or participate in the assignment. In some cases this labour is well compensated, either monetarily, with prizes, or with recognition. In other cases the only rewards may be kudos or intellectual satisfaction [19]. Asking the crowd usually costs less than having a single person work on the problem and as stated by James Surowiecki “under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them”[15]. This is because the average or the most picked answer of multiple persons is more accurate than what a single person answers[15].

1.2 Project overview

The goal of the project is to make an online crowdsourcing application that can determine what certain shapes mean to people. This paper describes the design process and can be used as a reference for developing similar applications. Because this application uses a crowdsourcing approach it will also produce useful information for the research on crowdsourcing.

The global idea of the application is as follows. Administrators submit images of 3-dimensional shapes using the application and users are asked what they think of the shapes. This will generate data which can be analysed and used to further the research on the semantics of shapes. An example of a shape can be found in Figure 1.

The reason to make a crowdsourcing application is because the meaning of shapes is highly subjective and therefore can’t be determined by a single person. Taking the average or the most used rating or description is far more useful and accurate [15].

The data generated by the application will give an idea of the semantics of shapes. This information can be used in designs to convey certain messages. For example if people find a shape to be beautiful it could be used in advertising. Or if a shape is perceived as futuristic it could be used in the design of a certain product.

The scope of this project is just building the application and presenting the data. The analysis of the data will be done in another project. Of course some of the data will be analysed in order to test the system and detect possible problems early.

2. BACKGROUND

An important problem to be solved is how to ask people what the shapes mean to them. The application is a combination of Google Image Labeler [4] or the ESP game [1, 16] and Facestat [7].

Google image labeler is the licenced version of the ESP game developed by Luis von Ahn [20]. This means that
the two applications are essentially the same.

The game works as follows. Two users, or players, connect to the website and start the game. The players are paired up randomly from everyone playing. Each game has a two minute time limit. The goal is to get as much points as possible by giving the same label or tag to an image.

At the start of the game the time is set and both players are presented with the same image. The players submit their labels without being able to see the other player's labels. When both players guess the same label points are rewarded and the players move on to the next image. The labels don't have to be guessed at the same time, therefore the server maintains lists of all the labels guessed by the players. Commonly guessed labels for images become off-limits (google) or taboo (ESP game). This makes sure new labels will be provided for the images. When the time runs out the player can see the labels of their partner.

When the application isn't used very often it might be a problem that there are not enough users to form a pair. The ESP game solves this by making it able to play against previously recorded sessions when no one is available [17]. Although the game is then not played live anymore it still provides new labels and new recorded sessions.

The reason for using this type of labelling for the shapes is that it is already well researched and it has been shown to work. Players find the game fun to play and are therefore willing to participate for free.

Another part of the game will be similar to Facestat. Facestat is a crowdsourcing application which is used to determine how photos of people are perceived by the crowd. Users can upload their own photos and have the photo judged by the crowd. Also they can look at and judge photos of other users. The judgements are based on various pre-selected questions. The creators like to call it “multivariate Hot-or-Not” [7]. This is based on the website hotornot.com where people can rate photos of other people with a mark between one and ten based on whether they find that person attractive or not. Facestat works similar but asks multiple questions like: How old do you think this person is and describe this person in one word. Rating of the faces in Facestat is, besides the regular users, also done by workers of Amazon's Mechanical Turk (MTurk) [12][2]. MTurk is a marketplace that enables coordination of the use of human intelligence to perform micro-tasks in return of a small payment [18]. The advantage of paying users is that the application does not have to be well-known to generate a lot a data. The major disadvantage is that users might be more interested in completing as much sessions as possible than to actually give accurate information. A way to overcome this is to train workers by mixing in the known items with unknown items [6] or by letting workings check the tasks of other users. However both solutions increase the complexity.

3. RESEARCH QUESTIONS

The main research question is how to develop a crowdsourcing application to determine the semantics of shapes. The solutions are found in section 4 design choices and section 5 implementation.

Another important part of the research focusses on motivating users to participate. This determines a big part of the success of the application. The game will be designed to attract as much users as possible. When the user is persuaded to play he or she must enjoy the experience to make sure that the most amount of input will be gathered per user. Especially returning users are useful because they will participate multiple times and understand the game better which makes their input more accurate.

Part of the more detailed research questions include how the points are rewarded and how the recorded sessions are used. Google Image Labeler awards more points for more detailed labels [20]. An algorithm has to be developed to be able to detect more detailed answers. A simple solution would be to award more point for longer answers. Also part of the research is how the recorded sessions are going to be generated to ensure the player will not notice when their partner leaves or when there is no partner from the start.

It is also necessary to develop a way to add shapes and questions. If a new question is added answers need to be gathered for every shape. If a lot of answers from other questions are already gathered the newest questions will remain to have fewer answers if the questions are presented at random. It would make sense to ask question with less gathered answers more often.

Also the order images are presented in has to be determined. It would make sense to give shapes which have less information gathered priority over shapes with more gathered information.

A way has to be found to present the gathered data for analysis. Considered solutions are raw data, sorted lists or even graphs. The chosen solution will be discussed in the section 4 design choices.

Also attention will be given to validating the input of users. Some users are going to be trying to cheat which will make the data less useful. The ESP Game has already taken measures to reduce cheating. These measures can be adapted for this project. Google Image Labeler also had problems with cheating [20]. Every image was given the same labels using words from a list. This could be done by entering the first word of the list repeatedly to see if someone was also using the list. If such a player was found both players would enter the next word on the list. Repeating this process they would guess a lot of the same labels in a short amount of time. This together with the fact that the words from the list are worth many points causes cheating players to get a lot of points. Because players can see what their partner submitted at the end of the game the words from the list were being used by other players to increase their score [20]. The solution was blocking the words from that list. Therefore it is good to regularly check what people are submitting to recognise cheating quickly.

An attractive and clear appearance of the website is also needed to attract more users. If it is unclear what the goal of the application is people are less likely to participate. Having a nice layout also makes more people interested.

4. DESIGN CHOICES

In this section are the possible solutions to various design problems discussed including the pros and cons of each solution and the motivation of the chosen solution. A model of the solution space has been made using a Morphological chart see Table 1. The name of the applications is Rating Game.

4.1 Motivation

The biggest problem with this project is how to motivate people to use this application. Because the user’s task is relatively easy the best way to this is to make it fun to do. That way people will participate for free. Google Image
Labeler and the ESP game have already established that people like to label random images just for fun. The goal is to make the application as attractive to users as possible. This is kept in mind while making every design choice.

Facetstat motivates part of its users by being fun just as the labelling games, however Facetstat also pays part of it’s users. Both labelling games have a point system whereas Facetstat does not.

As already mentioned returning users are especially useful. To increase the chance a user will return there must be something to achieve by returning. This is done just as in Google Image Labeler and the ESP game by keeping the high scores. Each player is identified by his or her ip-address and gets a unique number. This number is shown while playing a game. It has been considered to allow users to make an account with their own name, but as this is not a crucial feature this is something for the future.

### 4.2 Gameplay

Multiple possibilities for the design of the game were considered. One considered possibility is to make two separate games. The first game will be just labelling the shapes and the other game will be rating and answering questions about the shapes. The problem with this solution is that the labelling game might be more popular while the rating game collects less data. This could be solved by also using MTurk workers just as in Facetstat, but this would make it necessary to keep paying to ensure enough input. Another possible solution is to use only the labelling game or the rating game. If only the labelling game is used, the only output of the application will be labels describing the shapes, which is less useful and harder to analyse to determine the actual meaning of the shapes. The rating game will generate data that is easier to analyse and labelling the shapes can also be part of the questions, but because this game is not played with another player and it involves rating shapes instead of people it is unlikely that enough people will enjoy it and play it regularly to generate enough data.

Because of these problems both parts of the application are combined into a single game. There are multiple ways to combine both parts. A possibility is to ask the user to rate the image at the end of each round of the labelling game. Points can be awarded when users give a similar rating. Another possibility is to have just one round per image where the user has to answer multiple types of questions. A question which asks the user to label the image could be included. The second approach has been chosen because it generates more rating data which is easier to analyse statistically. Also, having one round makes the gameplay more coherent.

### 4.3 Players

Another choice that has to be made is whether or not to use a synchronized game with two players, or just make a single player game. The advantage of making a single player game is that it is easier to implement and the game would look virtually the same if the points are calculated using the average of previously played games. However, this does not take advantage of the fact that users like to play against other players. Some users even got the idea that they got to know the other player by playing the ESP game [1]. Therefore making a two player game could greatly increase the user’s interest and participation. Therefore it has been chosen to make a two player game.

Making a two player game does however create multiple problems which have to be solved. Implementation problems are discussed in section 5 of this paper. For other problem design choices have to be made. A choice has to be made to decide what happens after one of the player leaves. In the ESP game [17] this problem has been solved by using a prerecorded session to imitate a user. This has also been done in the Rating Game.

Another possibility is to have more than 2 players in a game at one time. This has not been done before. There might be a small advantage in that it is more fun to play in groups. However, this greatly increases the complexity of the system while it is not even visible for the players because there is no communication allowed because it would defeat the purpose of guessing.

### 4.4 Questions

For this application questions concerning the 3d-shapes will have to be established. These can be simple multiple choice questions, but can also be ratings on a scale or even open questions. Multiple choice questions and ratings are easier to analyse than open questions because the user’s answers are limited, but open question can give more creative and original results.

In order to give an idea of how the questions will look, an example will be given for each type of question. Multiple choice question: Which of the following words describes the shape most accurate? The answers that can be picked can either be labels of the shape given in the labelling game or answers to an open question. Rating question: How smooth is this shape? A rating from one to five or one to ten can be given with one being rough and five being smooth. Open question: Describe this shape in one word. This question is also used in Facetstat to describe a photo.

The rating questions are actually a type of semantic differentiation [10]. In semantic differentiation a respondent is asked to choose where his or her position lies, on a scale between two bipolar adjectives (for example: "Adequate-Inadequate", "Good-Evil" or "Valuable-Valueless") [21]. The adjective in the rating game will be chosen to apply to the shapes.

Administrators are given the possibility to edit and add questions. This can be done to collect new information...
when enough data on the existing questions have been collected. However after a question has been added there are no answers collected yet. So in order to make sure every question has a similar amount of answers the questions with the least answers are given priority when picking the questions.

4.5 Points
For the way in which points are rewarded also multiple options are considered. One option is to give points per rated image, another option is to give point based on the similarity of the answers of both players or the similarity to the average answer. The problem with only giving points for completing the questions is that it does not persuade the user to think about their answers and actually rewards filling in random answers. Giving points for the similarity to the average actually defeats the purpose of having two players, because users don’t have to guess what the other player is guessing. Therefore the score is calculated using the similarity of the answers of the players. This however does make it possible to cheat in a similar way as it has been done with Google Image Labeler. The data will therefore be monitored by an administrator to detect cheating.

4.6 Visual design
The goal of the visual design is to make an accessible yet attractive application. This is done by keeping the visual elements to minimum while using enough colour to make it appealing. An example of an in-game screen in given in Figure 2. To ensure compatibility the design has been tested in multiple browsers including Internet Explorer 6 - 8 and Firefox 3. The design is fluid which means it scales with different resolutions.

Since the user will perform only a small task it is important to make sure the application is easy to use. If users have to learn how the application works or have to sign up to participate they are more likely to lose interest. Therefore the application will have an easy-to-spot start button which starts the game immediately.

4.7 Solution space
All these design choices give an idea of the solution space. All the researched applications are mapped in the Morphological chart of Table 1. It is easy to see that the labelling games are most similar to this application. The important difference being that this application has more question types. The differences with Facestat is that this application awards points for completing a game and that it is a two player game and it supports more question types.

5. IMPLEMENTATION
To make the application widely available and easy to start it is web based. For the interface there are multiple options available. One option is to use browser plugins such as Sun’s Java or Macromedia’s Flash another option is to use html pages as the interface. Because plugins have to be installed it has been chosen to make the application available without them. To make the application more responsive JavaScript and XML are used in an AJAX approach [11]. This makes it possible to run the game without having to refresh the page. Communication with the server after the loading of the initial page is done by the JavaScript.

5.1 Storage
To determine the state of the game information about the users has to be stored. The way in which this is done is to use PHP sessions [13]. PHP Sessions allow the server to store variables for each client. The information is stored in a temporary file on the server and the users are identified, among others, using a browser cookie. The results of the games are stored in a MySQL database. To achieve communication between different PHP Sessions the database is used to store messages. What is stored in the PHP Sessions and the game sessions is discussed in the following sections.

5.2 Database
At the start of the implementation a model of the database has been made. See the UML class diagram in Figure 3. This model has been translated into actual tables in the database (see Appendix A). What follows is a description of how the database is modelled.

\[ \text{Figure 3: Class diagram database} \]
For each game an entry in the Sessions table is made. Each session consists of multiple ShapeSessions which are made for each shape. A session of the game is played by two players. A ShapeSession is linked to an entry in the Shapes table. A player has to answer 3 questions for each shape. A ShapeSession therefore has at most 6 answers. Each answer answers one question. There are multiple types of questions namely multiple choice, open and rating.

For each session the start time is recorded and for each ShapeSession the time of submission is stored for each player. The score is stored for each Session and ShapeSession. Other classes and attributes will be discussed in the following sections.

5.3 Matching
At the beginning of each game two clients have to be matched. This involves communication between PHP sessions therefore the first step is to put the client’s ip-address and the PHP session-id in the table queue. In order to ensure mutual exclusivity a lock on the table queue is acquired first. After the client is added to the queue the queue is checked for another client with a different ip-address. If a client is available the ip-addresses of both clients are added to the players table. From now on clients are called players. A session is made with a reference to both player entries. The player who found the match removes himself from the queue and sets the match value of the other player’s entry in the queue to the id of the newly created session. The id of the session is also stored in the player’s PHP session variables.

If no client is available the lock is given up and the thread waits for half a second an then retries to match. The client is not removed from the queue. This retrying is done for approximately 5 seconds. Before one client checks for another client in the queue it first checks if the match value is set to see if it is already matched by another thread. If it is already matched the thread stores the id of the already created session in it’s PHP session variables.

After each thread has received a session id it gives up the lock on the queue and goes on to retrieve the ShapeSession.

5.4 Shape sessions
To get a ShapeSession to a player it has to be determined whether there is a ShapeSession for which the player has not submitted his answers yet. If there is such a session the shape and the questions associated with that shape have to be retrieved. If there is no such ShapeSession a new ShapeSession has to be made.

When a new ShapeSession is made the Shape has to be selected first. All shapes are ordered according to how much answers to questions about them they have, with the fewest answers first. When they have the same amount of answers they are ordered at random. From this list the first shape is picked which hasn’t been picked in this game yet.

Question also have to be picked when a ShapeSession is created. A question is picked according to how few answers it has. For each ShapeSession the 3 questions with the least answers are picked. The question, the type of question and for multiple choice question also the answer choices are returned to the player.

5.5 Player activity
To avoid that one player keeps waiting when his or her partner has left, the system has to know when a player leaves the game. Because a player may need some time to answer the questions a simple timeout between getting the session and submitting the answer won’t suffice. Because of the internet’s design it is not possible for the server to know if the client is watching the page. It is however possible to let client generate requests periodically. When it is stored when the client last did a request to the server a timeout can be used to see if the player is still active.

Every time a player does a request to the server it’s own last active time is set and the other player’s last activity time is checked to see whether both players are still matched. If the player’s partner has timed out the partner’s answers are replaced by previously stored answers. See the next section.

5.6 Prerecorded Sessions
When there is no partner available or a partner leaves during a game, a player has to be simulated using prerecorded sessions. First it is checked whether there is a ShapeSession about the same shape which has the same questions. If there is such a ShapeSession it’s answers and time of submission is used as the other player. The time of submission is time between the creation of the ShapeSession and the time of submission of the player. If there are no ShapeSessions with the same shape and all the same questions various ShapeSessions are used. Of course it is possible that the question has not been answered for this shape yet when that’s the case a different shape is used. If that is also not available a random answer will be generated.

The answers of the prerecorded sessions are not stored. While the answers of the player are. This way user data can still be collected when there is only one player at a time.

5.7 Points
As said in the design choices games are given points based on the similarity of the answers of both players. Rating questions can be answered with a value between 0 and 4. If the answers are the same the players get 100 points if there is a difference of 1, 50 point and of 2, 25 points. Multiple choice answers only get 100 points if both answers are the same. Answers to open question are compared using a function which gives a percentage of similarity. In particular the PHP function similar_text [13] which uses a method described by J. Oliver[8]. If the percentage is higher than 80% both player get that percentage of 100 points. In order to award more complex answers with more points answers with an average length of 8 letter or more get double the points.

5.8 Data representation
Although the analysis of the data is outside of the scope of this project the data is shown to the administrators. This is done by storing several useful queries and displaying the results in tables. It is also possible to make and store your own queries.

These useful queries include a query which shows the average or most picked answer for every shape and question. The most picked answers are shown for multiple choice and open questions and the average answer is shown for rating questions. Another query shows the highest scores and the ip-address and number of sessions of the players. There are also several statistical queries which show for example how many question are answered for each shape, what the average time is a user plays a game and how many shapes are rated per session.
5.9 Administration

As already mentioned an administrator must be able to add shapes and to edit questions. Because a complicated uploading mechanism for the shapes is not necessary a simple approach has been chosen. An administrator with access to the server files can simply put images in or remove images from a folder on the server and press the refresh shapes button to update the database. The database remembers the filenames of the images and stores them in the shapes table. If a shape is removed it is still contained in the database because the table shapesessions refers to a shape. This is done by setting the value of the column available to false. When an image with a filename that is already in the database is added the shape will become available again.

Questions can be added and edited using a simple form. The text of the questions can be changed as well as the text of the choices of multiple choice questions. Questions can be turned off while maintaining the answers to the question by disabling it. A disabled question can also be enabled. Questions and their answers can be removed permanently. It is not possible to add new choices to multiple choice questions because this would make their answers inaccurate. This is because the players that have already answered did not see the new choice and therefore have not picked it even if they would have otherwise.

6. TESTING

Because this application is web based and uses asynchronous JavaScript the best way to test it is to use a browser and check if it shows the correct information. Different browsers have different JavaScript engines so it is necessary to test if it works in multiple browsers. Because a lot of tests have to be done it is easier to automate the testing process. A good way to do this is to use Selenium Remote Control (RC) [14]. Selenium RC consists of a server which controls browsers and client libraries which allows you to use different computer languages to communicate with the server.

The difficult part of the testing of this particular application is that it needs two players to play against each other simultaneously in order to test it properly. Fortunately there is a tool called Selenium Grid [9] which makes it easier to run tests in parallel. Based on the excellent Selenium web testing tool, Selenium Grid allows you to run multiple instances of Selenium Remote Control in parallel. Even better, it makes all these Selenium Remote Controls appear as a single one, so your tests do not have to worry about the actual infrastructure. Selenium Grid cuts down on the time required to run a Selenium test suite to a fraction of the time that a single instance of Selenium instance would take to run.

To test a two player game it is necessary to be able to synchronize easy and to allow communication between threads to compare questions and answers. The way this is done is by using the TestNG framework [3]. TestNG is a Java based Unit Testing framework. TestNG can invoke the same test method multiple times is different threads. This allows you to make a single test method for a player and to run it twice in parallel to test a two player game. Because TestNG also instantiates each test class only once communication between the two player methods can be achieved by using the class variables.

6.1 Testing a game

As said in previous sections each player is presented with an image of a shape and has to answer questions about them. When two players are playing against each other they have to get the same shape and the same questions. So the first step in testing (after the game is started) is to store the shape and the questions. The score is also stored to make sure it isn’t decreasing and it is the same for each player. After this is done the test class will fill in random answers and also store them.

It might be possible that the time runs out before the questions and the shape are retrieved from the browser. This has to be taken into account when comparing the sessions of the players. Also when an answer is submitted when there is a small amount of time left it might be rejected by the server because there is always a small time discrepancy between the server and a player. This is because there are several delays in the communication between client and server. There is a delay because messages have to travel between server and client, the propagation delay and it also takes time for the browser to execute the JavaScript and render the website. Therefore the client might think there is time left to submit an answer but when it reaches the server the time is already up.

When a game is completed both players are presented with all the asked questions, the shapes, their own answers, their partner’s answers and the score. All this information is retrieved from the browser and compared with the stored information. Before the comparison is made the stored questions are appended with the last questions from the result page to make sure both values contain the same amount of questions. The stored information must be the same as the information presented in the result page.

After that comparison is made the sessions of each player must be compared to each other. The test method of each player adds it’s stored values to a class variable of the test case. When the thread of one of the players has done this it waits for the other thread to add it’s values and it is notified that so it is ready to compare the sessions. The sessions have to be equal and when this is true the test is completed successfully.

Testing a single player game is done by running the same test method for two players with the only difference being that the comparison of the values of both threads is not made. A similar approach has been taken to test when a player leaves a game. After it is confirmed both players are matched one player leaves while the other continues playing. The test is successful when the player left finishes a game.

7. FUTURE OF THE SYSTEM

Of course the first thing the application should do is to collect data. After enough data is collected it can be analyzed for the purposes already mentioned in this paper. There are however new features that could be added.

Just as with the ESP game [1] there could be issues with cheating players. How this problem would be solved depends on the type of cheating. If people would always enter the same answer to each question it might be considered to present the questions different to each of the two player. For rating and multiple choice questions it might be done by changing the order of answers people can choose. Also the order of the three questions can be
changed. For the open questions a blacklist with words that are not allowed as an answer could be introduced. If people were cheating by exploiting flaws in the system (e.g. SQL injection, cross site scripting) they must be corrected. Another feature that can be added is the usage of accounts to identify players. At the moment high scores are calculated based on the ip-address of the players. When a player has an account it can add points from different locations which makes the game more flexible.

8. REFERENCES
APPENDIX

A. DATABASE OVERVIEW

**CREATE TABLE** `queue` (  
`ip` varchar(16),  
`sid` varchar(32),  
`match` int(11) DEFAULT NULL,  
PRIMARY KEY (`ip`, `sid`),  
FOREIGN KEY (`match`) REFERENCES `sessions` (`id`) );

**CREATE TABLE** `sessions` (  
`id` int(11),  
`player0` int(11) DEFAULT NULL,  
`player1` int(11) DEFAULT NULL,  
`player0_active` double,  
`player1_active` double,  
`score` int(11),  
`time` double,  
PRIMARY KEY (`id`),  
FOREIGN KEY (`player0`) REFERENCES `players` (`id`),  
FOREIGN KEY (`player1`) REFERENCES `players` (`id`) );

**CREATE TABLE** `players` (  
`id` int(11),  
`ip` varchar(32),  
PRIMARY KEY (`id`) );

**CREATE TABLE** `shapes` (  
`id` int(11),  
`image` varchar(32),  
`available` tinyint(1),  
PRIMARY KEY (`id`) );

**CREATE TABLE** `shapesessions` (  
`id` int(11),  
`session` int(11),  
`shape_id` int(11),  
`questions` varchar(32),  
`time0` double DEFAULT NULL,  
`time1` double DEFAULT NULL,  
`score` int(11),  
PRIMARY KEY (`id`),  
FOREIGN KEY (`session`) REFERENCES `sessions` (`id`),  
FOREIGN KEY (`shape_id`) REFERENCES `shapes` (`id`) );

**CREATE TABLE** `answers` (  
`shapesession_id` int(11),  
`player` tinyint(1),  
`question_id` int(11),  
`answer` varchar(32),  
PRIMARY KEY (`shapesession_id`, `player`, `question_id`),  
FOREIGN KEY (`question_id`) REFERENCES `questions` (`id`),  
FOREIGN KEY (`shapesession_id`) REFERENCES `shapesessions` (`id`) );

**CREATE TABLE** `questions` (  
`id` int(11),  
`available` tinyint(1),  
`type` tinyint(4),  
`question` text,  
PRIMARY KEY (`id`) );

**CREATE TABLE** `choices` (  
`question_id` int(11),  
`number` int(11),  
`choice` varchar(32),  
PRIMARY KEY (`question_id`, `number`),  
FOREIGN KEY (`question_id`) REFERENCES `questions` (`id`) );