ENTITY PREDICTION TASK IN NOISY ENVIRONMENTS

20 Questions game with an open world assumption and noisy data

Hisham Alkaed (2671370)

A thesis submitted in fulfilment of the requirements for the VU Bachelor of Science degree in Artificial Intelligence

JUNE 24, 2022

VRIJE UNIVERSITEIT AMSTERDAM
Supervisors: Dr. Benno Kruit & Dr. Stefan Scholbach

Table of Contents

1. Abstract

This paper takes the 20 Questions game as an instance of the entity prediction task problem with incomplete knowledge about the world and inconsistencies in the data. The focus of this paper is on finding computationally simple techniques to function in a world full of contradictions, rather than solving the world itself. To do this, an engine is created where the game can be simulated. A subset of the Yago 4 knowledge graph is used. Multiple approaches are introduced, motivated and evaluated. Various kinds of graphs have been created that illustrate the performance of the given approaches. Finally, convincing statistical results suggest that the main idea behind the given approaches is sufficient to solve the game within the given noisy environment and under the open-world assumption.

2. Introduction

The amount of knowledge people have access to in today's world is increasing exponentially as you read this. Most of the knowledge is stored in online databases. Different databases contain different kinds of information. And the same information could be fetched from different sources. The complication here is that different databases have different views about the world. That is, what is known as a fact in database A is either unknown or its contradiction is known to be true in database B. Moreover, what is known as a postulate for experts in some fields might not be known to other individuals or even not exist on any dataset yet.

This phenomenon poses an interesting aspect in the 20 Questions (20Q) game. The game originated in the early 19th century as a magical machine that could read people's minds. This game is played between a "questioner" and an "answerer". The "answerer" thinks of an object that the "questioner" should guess within 20 questions. These 20 questions should be answered by the "answerer" who can only answer in a binary manner, i.e., with either 'yes' or 'no' as an answer. Since different variant of the game exists - e.g. by the inclusion of a 'probably' or 'maybe' answer -, in this paper the basic version of the game is considered. The best strategy the "questioner" can follow is asking questions that minimize the search space and maximize the information gain with each answer. Given the introduced phenomenon, the "questioner" and the "answerer" may have different views about the truth in the search space. If the "questioner" is unaware of this fact, it becomes difficult for them to guess the hidden object. Hence, it is important when playing this game to realize that the world is full of incomplete and inconsistent information. In addition, an open-world assumption should be taken into consideration. That is, the truth value of a statement may be true irrespective of whether or not it is known to be true (Baader et al., 2003).

This paper discusses the 20Q game because it naturally embodies the problem and it has numerous direct and indirect real-life applications. Some of the direct applications are systems used in the healthcare sector, where medical staff log patient symptoms to find the most likely illness that correlates with the symptoms. In this case, the contradiction in the data could originate from a) patients who are not fully acquainted with the symptoms, b) miscommunication between

the specialists and the patients, c) incorrect filling of the symptoms in the system or d) the system having incomplete knowledge of some diseases and illnesses. If the system's designers are not aware of these complications the predictions would contain a high percentage of false positives. In addition, such a system should have an efficient search strategy that utilizes in real-time each symptom filled in. While it is easy to see the correlation between the 20Q game and the previously mentioned problem, it also appears indirectly in the mathematical noisy 20 questions problem (Variani et al., 2015), stochastic search (Tsiligkaridis et al., 2014b), searching hypothesis spaces (Cohen and Lake, 2016), object detection (Chen et al., 2016) and many more.

The focus of this paper lies in developing strategies that could be adopted by the "questioner" playing the 20Q game to win even when dealing with noisy data. The "questioner" in this case is the computer. The source of the noise is irrelevant to this matter. In addition, the questions asked should utilize the previously answered questions and minimize the search space. The focus of this paper is to utilize useful information and not to solve any inconsistencies in the data. More specifically, an attempt is made to answer the following research question:

"How to correctly and efficiently complete an entity prediction task with both an open-world assumption and inconsistencies in the data?"

To address the research question, an engine is introduced where a) the 20Q game can be simulated, b) different strategies of playing the game can be compared through global metrics and c) some structural inconsistencies could be introduced to imitate the phenomenon explained.

Previous studies show that it is possible to solve the 20Q game within the question limit. Some of them were even able to handle noisy data. A typical decision tree would seem like a perfect fit for this problem. This is due to the fact that decision trees make splits in the data based on whether or not a specific feature is true which resembles having "yes" and "no" in this game. However, in that case, two requirements should be satisfied: 1) a well-defined Knowledge Base (KB) where all features for all entries are present and 2) no misleading answers could be given by the "answerer". Both are almost impossible to have in real-life. Burgener (1990), as explained in his patent, was able to overcome the second requirement by introducing a relevance table and using ANNs that were linked together in a matrix format. This allowed him to differentiate a huge number of plausible objects within the first 20 questions and handle some wrong answers.

Moreover, Fitzgerald (2017) was inspired by Burgener and introduced a bot that utilized neural nets. This approach resulted in better performance than Burgener's in terms of the number of questions asked before finding the hidden object. However, his solution suffered from scalability issues; the bot was computationally expensive, tested on a small dataset, and incapable to handle a large number of incorrect answers. Other people were also inspired by Burgener, such as (Wu et al., 2018) who used a similar approach where they improved Burgener's relevance table performance by applying entropy-based tricks combined with the usage of neural nets and Natural

Language Processing. This system relied on a self-constructed relevance table of 38,375 samples. Both their strategy and Burgener's were dependent on an existing KB.

In (Hu et al., 2018), the authors introduced a novel policy-based Reinforcement Learning model that could both efficiently solve the 20Q game and handle noisy answers. In addition, their solution was not dependent on the Knowledge Base, i.e., it was scalable. While all the previously mentioned approaches made use of complex and computationally expensive neural nets, the paper by (Dey et al., 2019) introduced a probabilistic model that utilizes a knowledge graph (KG) and handles some incorrect answers due to its probabilistic nature. Some of the probabilities introduced in their paper were KG-specific and relied on the fact that all entities in the KG have an equal amount of metadata tags. The main differences between the previously mentioned studies and this paper is that this introduces a novel approach that 1) does not rely on a specific dataset, 2) uses KGs as the source of the knowledge, 3) does not use any complex neural nets and 4) is specifically optimized and evaluated to handle noise in the answers.

In the following sections, the used engine is introduced. Thereafter all strategies used to play the game in the basic settings, i.e. without noise, are presented. And all model variants used to handle the noise in the answers are motivated and explained. Followed by both a quantitative evaluation and statistical analysis where the variants are compared. Finally, some limitations of the current study, future work and concluding remarks are discussed.

3. Methodology

In this section, the framework that simulates the game is discussed (3.1). Next, two naive bots that could solve the game in the basic setting, i.e. without any noise in the data, are introduced (3.2). Followed by motivation and explanation of the models that could handle some misleading answers (3.3).

3.1 The engine

To start with, a framework has been built to solve the 20Q game using python. It uses a KG and makes automated SPARQL queries that fetch the required information at each step. The used KG is a subset of YAGO4 (Pellissier et al., 2020). It mainly contains the most popular searched entities during the year 2021. It holds in total 278,854 statements divided over 561 classes. The distribution of the classes is depicted in Figure1.

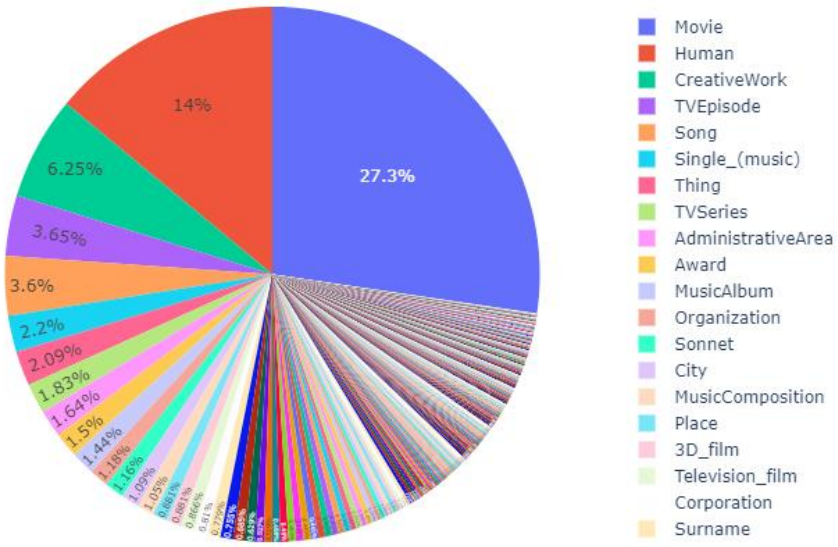


Figure 1

All components of the engine were made modular to allow for free utilization of any KG. Regarding the technical part, the game could be played in two modes, a human mode and a machine mode. As the names suggest, the human mode is played between a human (answerer) and the machine (questioner), while the other mode is played between one of the created bots and an Answerer bot. The Answerer bot picks a random entity from the KG and gathers all related information about it to answer any asked question. Furthermore, the Answerer bot has an unreliability attribute that could be set to any value between 0 and 1. The 0 here implies that the bot has perfect knowledge about the picked entity and that it answers all questions correctly. And the closer the value is to 1 the less knowledgeable the bot is about the picked entity. For instance, 0.2 implies that in each question there is a 20% chance to give a wrong answer. The inconsistencies or the wrong answers are introduced according to a random uniform distribution. The used pseudorandom number generator follows the principle of the Mersenne twister (*Numpy.Random.Randint — NumPy v1.15 Manual*, 2018). The unreliability measure is the method by which noise is introduced into the data. That is, noise originates from the "answerer" side in this setting and not from the Knowledge Base. A tournament option is also added where the game could be run for several iterations in machine mode. This has been implemented to make the automation of the game possible to allow for qualitative testing of the bots and models to be introduced in 3.2 and 3.3.

All facts are stored as triples within the engine. Questions, however, are represented as tuples, namely a predicate and an object. For instance, typical questions are countryOfOrigin America? Type Human? Award Grammys_for_best_actor? Meaning: does the entity the "answerer" thinks about have a country-of-origin America? Is the entity the "answerer" thinks about a human? Does the person the "answerer" thinks about have the Grammys award for best

actor? Both the predicate and the object in those cases are full URIs that are only parsed when playing against a human for readability. This representation has been chosen to simplify the game interaction without adding any complexity to the solution.

According to the rules of the game, the "questioner" wins when a correct guess is made. A guess is made by asking one of the following questions: Label X? sameAs X? givenName X? image X? where X is the entity the "answerer" is thinking about. It is worth mentioning that making a guess in the built framework does require the entity to have one of the previously mentioned relations. Within the picked KG, it is ensured that the "questioner" can make a guess for each entity because all entities have at least one of the previously mentioned relations used to make a guess. This is how the game is guaranteed to terminate. When playing the game in the machine mode with the unreliability attribute being non-zero the Answerer bot is not allowed to give misleading answers when a guess is given. This again ensures that the game terminates when some misleading answers are provided.

3.2 Bots

Here two naïve bots are introduced that could play the game in the basic settings, i.e. without misleading answers. From here on, two different words will be used to denote two kinds of approaches. "bot" will be used to refer to approaches that should converge in the basic settings of the game, while "model" will be used to refer to approaches that should be able to handle some misleading answers.

3.2.1 SubGraphs

The main strategy of the SubGraphs bot is asking random questions and utilizing the answer provided by the "answerer" to make a new subgraph which is used to pick the next question. A "no" answer leads to removing all entities that have the same predicate object combination (PO) as the question. While a "yes" answer makes the bot create a new subgraph that contains all entities that have the PO mentioned in the question. For instance, getting a "yes" answer for the question "type Human" and a "no" answer for the question "knowsLanguage English" makes a SPARQL query that includes all humans from the KG and filters out those who do not speak English. This smaller KG is then used to pick randomly the next question. Moreover, the asked PO is removed from the new subgraph. This is done to prevent any duplicate questions. Theoretically speaking, this bot should be able to guess any hidden entity as long as there is no limit to the number of questions that could be asked. Although practically, within the implemented environment, it is not made possible to have the number of questions unbounded.

3.2.2 Entropy

The Entropy bot builds upon the SubGraphs bot strategy. The only difference between the two is the way the next question is picked. While the SubGraphs does that randomly, the Entropy bot picks the PO that minimizes the entropy and maximizes the information gain. The entropy H is measured according to the Shannon Entropy formula (Shannon, 1948), namely:

$$H(X) = -\Big(P(x_{yes}) \log_2 \Big(P(x_{yes}) \Big) + P(x_{no}) \log_2 \big(P(x_{no}) \big) \Big) \qquad (1)$$

Where $P(x_{yes})$ is the probability of some PO. This probability is calculated by:

$$P(x_{yes}) = \frac{C_{x_{yes}}}{C_t} \qquad (2)$$

Where $C_{x_{yes}}$ denotes the count of all entities in the KG that have the PO and C_t denotes the total count of all entities in the current KG. $P(x_{no})$ is calculated according to the same concept. Note that in this setting C_t is not constant because each question creates a new subgraph.

3.3 Models

In the following section, a total of 6 models are introduced that are meant to play the 20Q game in a noisy environment. The general idea of the following models is to not neglect entities upon a "no" answer and still consider other entities that do not have the PO asked upon a "yes" answer. This is to account for the noisy answers and the open-world assumption respectively. All the following models utilize hash tables that connect some POs to some score. This data structure has been chosen for its low-time complexity lookup operation, namely O(1). All the hash tables are referred to as indices.

3.3.1 Basic

The Basic model is the simplest model and is used as a building block for the models to be introduced in the following sections (3.3.2, 3.3.3, 3.3.4). Upon initialization, this model constructs an index that connects each PO present in the KG to a score. This score is initialized to be 1. Throughout the game, the scores are updated. The update action starts with making a query that retrieves all related POs from the KG. For instance, asking the question "type Human?" leads the Basic model to retrieve all human-related POs such as: has age X, has name X, has award X, etc. Next, if the answer is "yes" all related entries in the index are reinforced by incrementing them by 1. Whereas, if the answer is "no" all related entries are punished by decrementing them by 1. Thus, getting two "yes" answers for the first two questions of the game will increase all entries related to the PO asked by two, resulting in some entries in the index having a score of 3. In all cases, the asked PO is deleted from the index to prevent any duplicate questions. The next question is

randomly picked from the POs that have the highest score. This is how the model focuses on a cluster in the data. The model wins the game when the picked PO qualifies to be a guess and is correct. In other words, when the model picks the next question to be “label Taylor Swift?” and the “answerer” is thinking of Taylor Swift as the hidden entity, the model wins. Note that this is made possible by the fact that both guesses and questions have the same format, namely Predicate Object. Theoretically speaking this model should converge and win in all cases. The reason is that no entities are deleted from the index unless they are asked. And the “answerer” is not allowed to lie upon a guess. Thus, even if some misleading answers were given, the model should eventually be able to make a correct guess. However, efficiency of finding the right entity might still vary among the different models.

3.3.2 Combined

The Combined model uses three indices. The first is called the Score index which is the same as the one used in the Basic model. The second is the Count index which connects each PO to its number of occurrences in the KG. This has been included because asking a question about the most common PO provides the most information to the "questioner". Note that the notion of most information is different from the highest information gain. Simply, asking whether the entity the “answerer” is thinking about is of type owl:Thing will not make the search space any smaller (since everything is of type owl:Thing), but it will give the “questioner” one extra hint about each entity in the KG. This is improved in (3.3.4). The third used index is the Combined index which connects each PO to its weighted average value from the Score and the Count indices. This weighted average is calculated as follows:

$$x_{combined} = \sum w_{index} vi_{index} \mid index = [Score, Count] \qquad (3)$$

Where w_{index} corresponds to the weight assigned to the index. And vi_{index} corresponds to the value of entry i of that index. The Score index weighs 0.9, while the Count index weighs 0.1. Those values have been assigned through a random search that will be discussed later in this section. The next question in this model is picked by randomly picking a PO from the highest values from the Combined index.

The Count index is static throughout the whole run, while the Score and the Combined indices change. Since the difference between the values of the Score and Count indices is significant, updating the Score index with plus and minus 1, for “yes” and “no” respectively, hardly affects the weighted average. For instance, if some PO occurs 20,000 times and its initial value in the Score index is 1, after updating it with a “no” the weighted average changes from 2000.9 to 2000. Moreover, those update parameters highly depend on the respective KG. The bigger the counts are the higher the update parameters should be. Additionally, the variance in the counts affects the optimal parameters. Hence, a random search has been applied to find the best values

for updating the Score index along with the weights that should be assigned to each index. The objective of the random search is to minimize the number of asked questions before winning the game. A detailed description of the applied random search could be found in Appendix A.a. In total, one thousand iterations have been played to get the optimal parameters. The optimal parameters are to assign 0.9 and 0.1 to the weights of Score and Count indices respectively, to increment the Score index by 39,730 upon a "yes" answer and decrement it by 2240 upon a "no" answer.

3.3.3 Normalized

The main idea of the Normalized model is to create one index that could both capture the information from the Count index mentioned above and be updated directly. This should make updating the index easier, which is computationally less complex and allows the solution to be more modular and scalable. To achieve this, the Normalized model creates a new index upon initialization. This index has the same entries as all other indices created so far and the normalized count of the Count index as values. Since all values are normalized counts, i.e. between 0 and 1, it is possible to simply update the index with plus and minus 1 for "yes" and "no" respectively. The next question is randomly picked from the POs that have the highest normalized score.

3.3.4 Entropy Combined

In this subsection, two model variants will be discussed. The goal of these two variants is to be able to handle noisy answers and ask questions that maximize the information gain simultaneously. To do this, the Shannon Entropy mentioned in formula (1) will be used. The main difference between the two following model variants is the number of indices they use.

3.3.4.1 Entropy Combined 1 (EC1)

This variant utilizes three indices similar to the working of the Combined model (3.3.2). Namely, the Score index, the Entropy index and the Combined index. There are two main differences between the Entropy and Combined model. The first is that Combined uses the Count index while this model uses the Entropy index. The Entropy index contains, instead of counts of the POs, the Shannon entropy (Shannon, 1948) of each PO. This is done upon the initialization of the model and computed only once because the number of occurrences of each PO stays the same throughout the whole run. In other words, POs are not neglected upon a "no" answer. The second difference is the manner in which the Score index is updated. In the Combined model (3.3.2) the updating values were chosen to be different from plus and minus 1 due to the substantial difference between the values of the Score and the Count index. This is not needed here because both the values of the Entropy index and the Score index are between 0 and 1. Hence the standard plus and minus 1 is used to update the Score index. As for the calculation of the Combined index a formula comparable with formula (3) has been used, namely:

$$x_{combined} = \sum w_{index} vi_{index} \mid index = [Score, Entropy] \qquad (4)$$

The weights in the previous formula should be assigned such that the model has the best performance. This is done through a random search where all possible splits have been provided and the objective function was to minimize the number of asked questions needed to win the game. The best split is 0.1 for the Entropy index and 0.9 for the Score index. A detailed description of this random search can be found in Appendix A.b.

3.3.4.2 Entropy Combined 2 (EC2)

The main idea of this variant is to use one index that can capture the information provided in the Entropy index to ask entropy-minimizing questions and be updated directly. This should give a less computationally complex solution and allow for more modularity. The index used here is the same Entropy index mentioned in the previous variant. Since the Entropy index has values all between 0 and 1, it is possible to update those directly by adding and subtracting 1 upon a "yes" and "no" answer respectively. The reason why it is possible to update the entropies directly will be explained in the next paragraph.

Both EC1 and EC2 pick the next question by taking the entry with the highest value from the Combined and Entropy indices respectively. It may sound counter-intuitive to pick the highest entropy score to minimize the entropy, but one thing that must be noted here is that picking the highest entropy will lead to using fewer bits which implies a higher information gain according to the Information Theory (Jaynes, 1957). Although the entropy values are updated, taking the highest one will still yield the highest information gain. This will be explained through an example. Assuming that the best first question is whether the entity thought of is a human. And assuming that the answer is "yes". This leads to incrementing all entropy scores of the related POs. The next question will be the one with the highest entropy score, which in this case will be the entropy minimizing human-related PO. The same reasoning could be given to updating the Score index and taking the weighted average in EC1.

All the previously mentioned approaches in 3.3 are theoretically guaranteed to converge. There are two reasons for this. One, the approaches do not neglect any entities in the KG. Two, although the asked questions get removed from the index to avoid duplicate questions, the Answerer is not allowed to lie upon a guess. Hence, eventually, if enough questions are asked the models should be able to guess the hidden entity. The only requirement that should be satisfied is that the number of questions allowed is unbounded. A summary of all the previous models is provided in the table in Appendix B.

4. Experiment

In this section, the experiment performed to evaluate the performance of the previously proposed approaches will be explained and motivated in detail. A quantitative evaluation is conducted and

a statistical analysis of the results retrieved from the evaluation is performed. The goal here is to find the model with the best performance. The performance is measured by the number of won games in total and the number of questions needed before winning the game.

The quantitative evaluation was performed by running multiple tournaments between the different approaches and the Answerer bot. Each tournament was run in different settings. The compared approaches are Combined (3.3.2), Normalized (3.3.3), EC1, EC2 (3.3.4) and the Entropy bot (3.2.2). Note that, the SubGraphs (3.2.1) bot and the Basic model (3.3.1) were not evaluated. As mentioned earlier, this decision was made due to the random nature of those two approaches. In addition, those two were only introduced as building blocks for the other approaches.

In all run tournaments, the maximum number of allowed questions was set to five hundred. That is, if in a game the "questioner" reaches the 500^{th} question without correctly guessing the hidden entity it loses. The question limit is chosen to be five hundred because guessing the correct entity when there are misleading answers would require the "questioner" to ask more questions. And the higher the amount of misleading answers, the more questions are needed to guess the right entity. In addition, having the question limit higher than five hundred would make the experiment almost infeasible. And within the five hundred questions it should be possible to note if there are any actual discrepancies between the different approaches. As mentioned before, all models should always be able to guess the hidden entity. Thus, even when a model loses, it does not mean that that model cannot guess the entity. It merely means that within the 500 set question limit it cannot. However, this is not true for the Entropy Bot because as explained earlier this bot uses the subgraphs technique in which it neglects clusters in the data and focuses on others. Thus, if some wrong answers are given, it will focus on the wrong cluster in the KG and never be able to find the hidden entity. Moreover, each tournament contains 30 games. And each approach has been evaluated on seven distinct levels of probability of wrong answers, starting from 0% up to 30% (intervals of 5%) chance to get a wrong answer in each question. Thus, each approach was put in 210 games: seven tournaments of thirty games.

Furthermore, there are three factors that introduce randomness in this setting. The first is the game itself. Each entity in the KG needs a different number of questions for it to be reached in the best-case scenario. Running two tournaments of thirty games without any chance of getting a wrong answer will still give us different results. This causes a high degree of uncertainty about which entity was picked in each game, how many questions it would need to be guessed in the best-case scenario and how much that would deviate from the average. The second factor which introduces randomness is the unreliability measure in the Answerer bot. The place where a wrong answer occurs can significantly change the course of the game. Let us assume only one wrong answer is given throughout a whole game. If this wrong answer is given at the beginning of the game, it will probably lead to the "questioner" asking more questions because it then focuses initially on the wrong cluster. While if it was given at the far end of the game, it will hardly affect the number of questions asked since the "questioner" would have already gotten enough hints about the hidden entity. The third factor is the randomness introduced by the proposed approaches,

each one of them picks randomly one PO from the highest scored POs to be the next question. To reduce the randomness in these settings, instead of picking a random entity for each game, each tournament was run for one entity only across the different approaches. In total, five entities were carefully handpicked such that they are representative of the KG. The entities and the relevant information about them can be found in the table below:

	Grey's Anatomy	*Elizabeth II*	*Robin Williams*	*Fast & Furious 6*	*10,000 Hours*
Type	Series	Queen, Human	Actor, Human	Film	Song
Number of triples present in the KG	237	79	51	9	4

While running all tournaments on the same five entities reduces the randomness of one of the three factors, namely the game itself, it does not reduce the randomness of the other two. Hence, a statistical analysis is carried out to check for the statistical difference in the results. The Null-Hypothesis states that "there is no difference in performance between the different approaches". The statistical analysis will be performed by applying an independent two samples T-test. In addition, 95% and 99% confidence intervals of the number of asked questions of each tournament will be computed if possible. According to the central limit theorem (Kwak et al., 2017) if there are at least thirty occurrences, we can assume that the sample mean is normally distributed. Hence a normal distribution is used to estimate the confidence interval in the settings when there are exactly thirty values. Otherwise, i.e. if there are less than 30 values, a T distribution is used to estimate the confidence interval. This occurs when a model in a tournament does not win all 30 games.

5. Results

In this section, the results gathered from running the tournaments will be presented. To start with, each approach was put in 35 tournaments: 7 different unreliability levels x 5 different entities. Each one of those tournaments resulted in thirty values corresponding to the thirty played games within that tournament. Those thirty values were converted into three values and two intervals: the number of won games (out of 30), the average number of asked questions in all games, the average number of asked questions in won games, the 95% and 99% confidence intervals of the number of asked questions. In some tournaments, there were not enough occurrences to estimate the confidence intervals and hence a NaN was used instead. For instance, when a model did not win any game or won a single game in a certain setting it is not possible to estimate the confidence interval for the number of asked questions in won games. In addition, the confidence intervals might go below zero. This is mathematically correct, however, within the game environment, this is not possible since the "questioner" cannot guess the hidden entity with a negative number of

questions. Hence those values were written in red colour. (check Appendix D) Lastly, naturally the fewer values there are in some setting the larger the confidence intervals would be, and vice versa.

Next, the values discussed in the previous section were used to generate plots. For each setting, three line plots were generated. Namely, the number of won games, the average number of asked questions overall and the average number of asked questions in won games. For the latter two, the 95% confidence intervals are visualized as shaded regions. In addition, total average plots across the different entities were generated. In that case, two extra boxplots were generated: the number of asked questions in all games and in won games. Those were added because the shaded regions in the line plots were overlapping a lot. While all plots of the five different approaches were plotted across the seven different intervals of the probability of wrong answers, the entropy bot is omitted in the plots of the average number of asked questions in won games. This is because

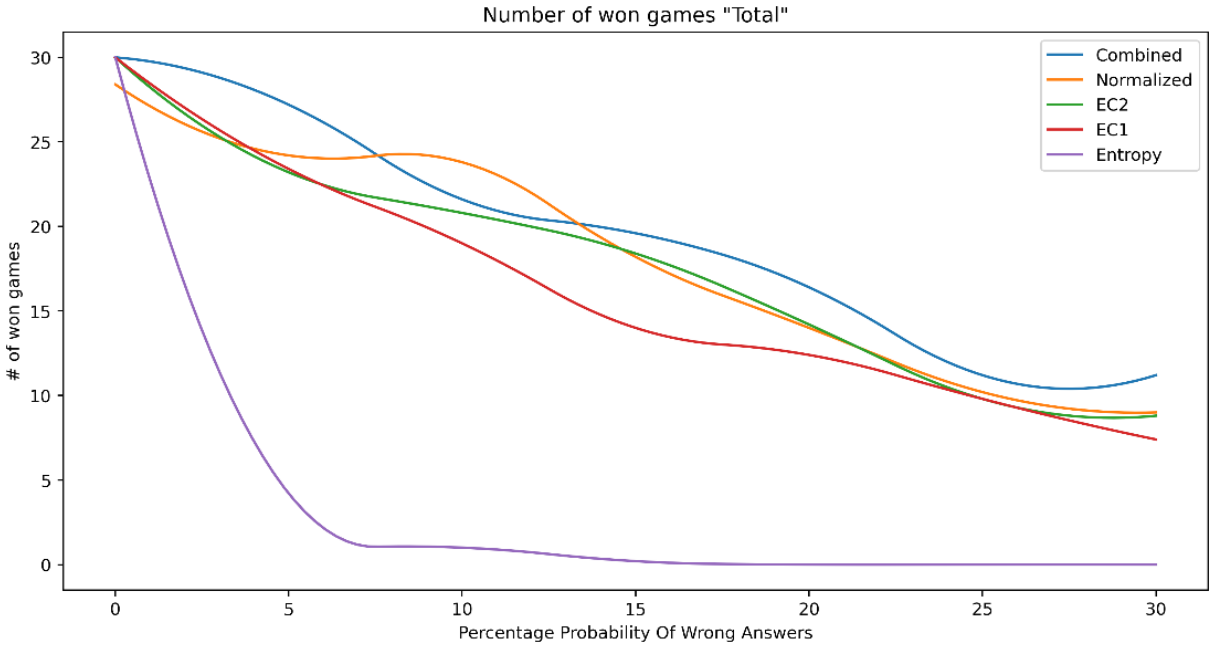


Figure 2

in most of the runs that bot did not win any games or won a single game. This means the confidence intervals, in that case, could not be computed or were inaccurate. Figure 2 depicts the total average number of won games throughout the whole experiment. While the whisker plot in Figure 3 represents the total average number of asked questions when there is no chance of receiving misleading answers by the Answerer bot.

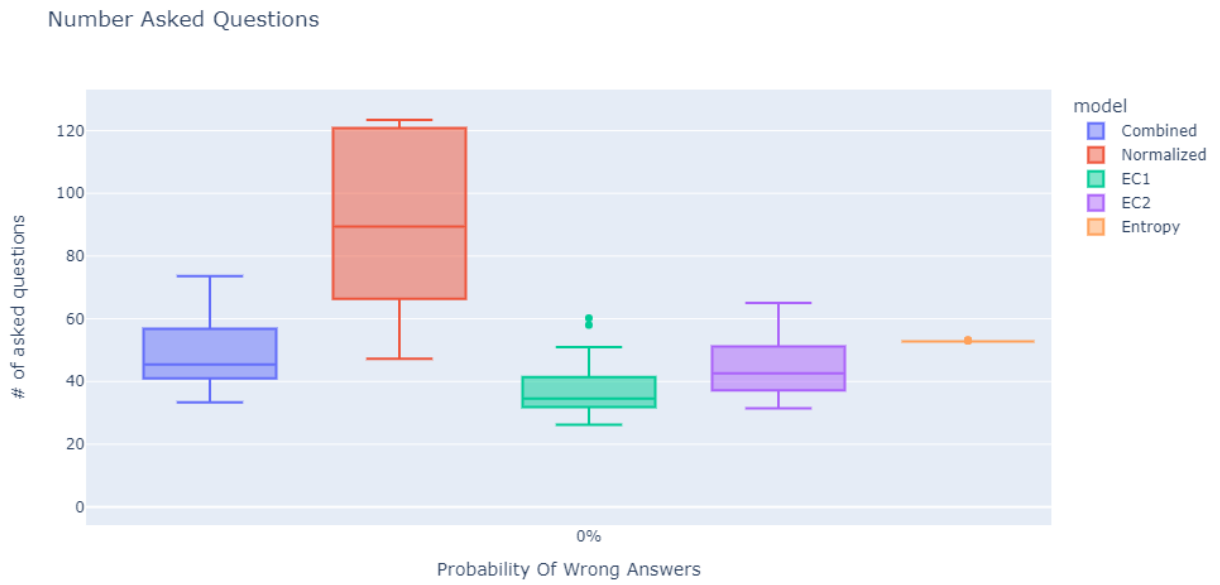


Figure 3

As for the total average number of asked questions in all games and in won games Figure 4 and 5 have been provided respectively. Figure 4 is a line plot with the 95% confidence intervals shown

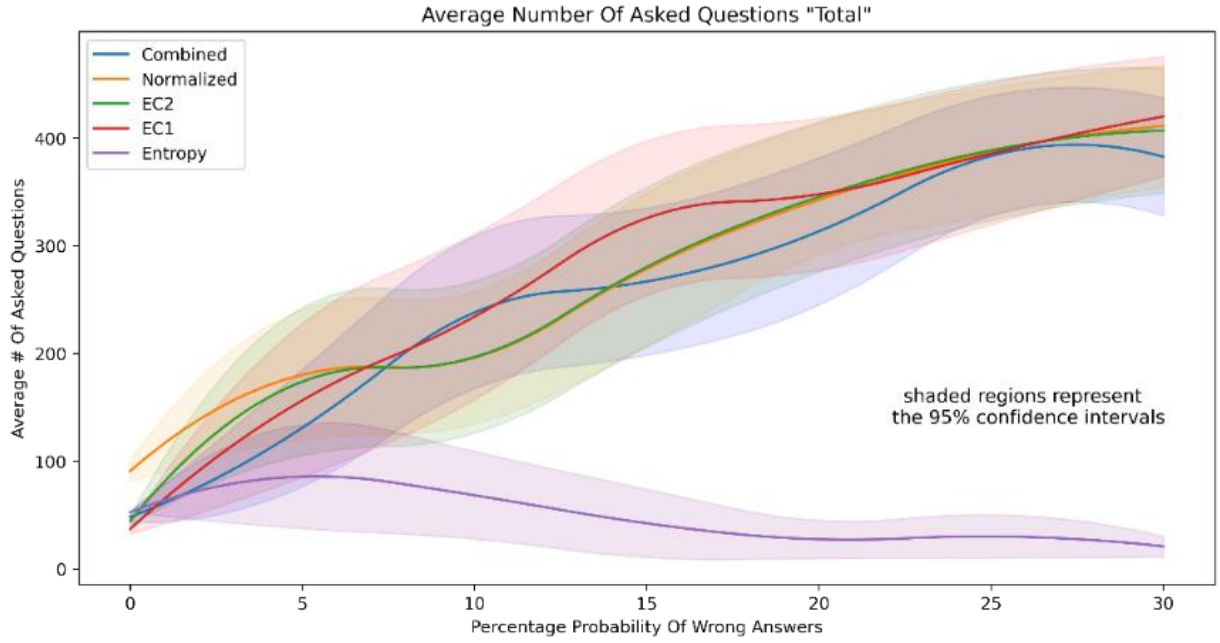


Figure 4

as shaded areas while Figure 5 is a whisker plot. Each individual box in all box plots stands for 150 games. All provided whisker plots can be clicked upon to navigate to an interactive version of the plots. In Appendix F.f, all plots representing the total average performance throughout the

whole experiment can be found in full size. Each plot of the total performance plots represents 5250 games.

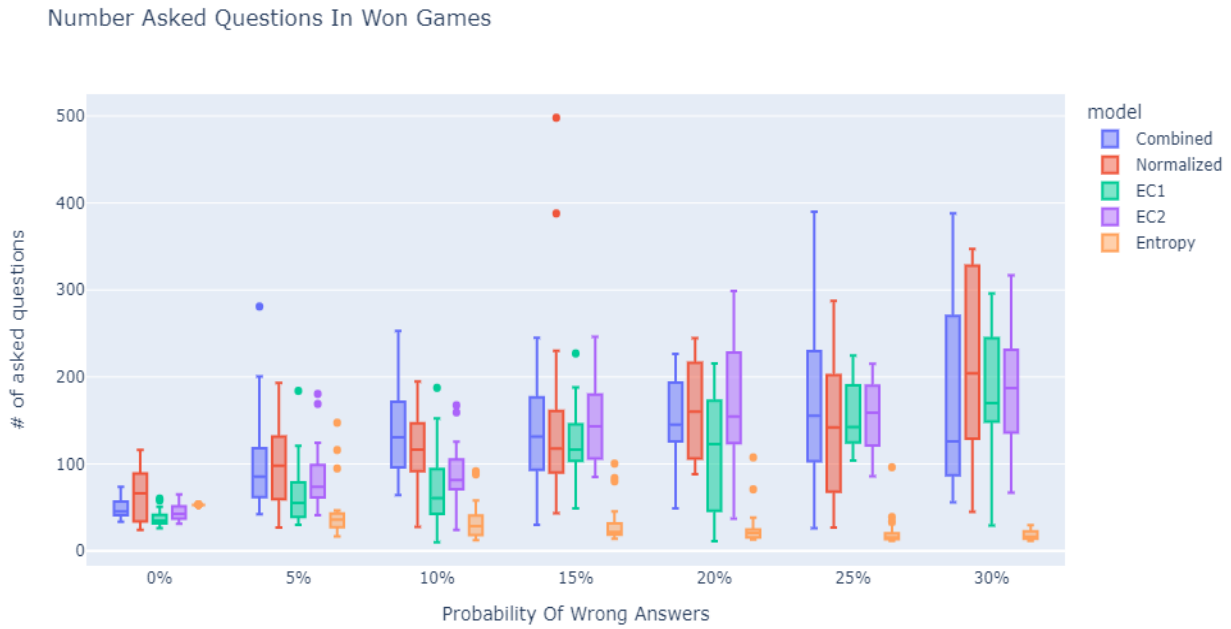


Figure 5

To make the graphs more visually appealing all line plots have been smoothed. That is, although seven points on the x-axes have actual values, for each one of the models a polynomial interpolation has been applied to create thirty points with values on the x-axes. This resulted in a smoother line that does not have edges. Lastly, the T-test results were computed. Three tables were constructed for each metric. Namely, the number of won games, the number of asked questions in all games and in won games. The p-values less than 0.05 were written in red. Table 2 is a compact version of the three tables in Appendix G. The left side represents the number of won games, the middle the number of asked questions in all games and the right the number of asked questions in won games only.

	Combined	*Normalized*	*EC2*	*EC1*	*Combined*	*Normalized*	*EC2*	*EC1*	*Combined*	*Normalized*	*EC2*
Normalized	0.7404	1			0.7925	1			0.9645	1	
EC2	0.6745	0.9281	1		0.8825	0.9161	1		0.4432	0.3647	1
EC1	0.4749	0.6919	0.7583	1	0.7763	0.9671	0.8907	1	0.2698	0.1997	0.7127
Entropy	0.0135	0.0230	0.0267	0.0462	0.0011	0.0004	0.0011	0.0011	---	---	---

Table 2

6. Discussion

After conducting the experiment and collecting the results, in this section the results will be interpreted and put into context. To start with, it is noticeable that in all plots there is a difference

between the performance of the Entropy bot and the proposed models. In the plots representing the number of won games, it is observable that the Entropy bot goes from winning all the games when there are no wrong answers at all to losing almost all games when there is a slight increase in the chance of getting a wrong answer. This is while the other models still win a reasonable number of games throughout the whole experiment. This is reflected in Figure 2 and in the individual entities plots (Appendix F.a to F.e) about the number of won games.

As for the number of asked questions, we can also see an enormous difference between the models and the Entropy bot. Two cases will be discussed here, the case of no wrong answers and the case where there is a chance to get a wrong answer. The first case is best visualized through the whiskers plot in Figure 3. While the models have some variance throughout the runs, the Entropy bot does not. This is because the Entropy bot is deterministic. Moreover, both boxes of the EC1 and the EC2 lie lower than the one of the Entropy bot. This is interesting because even though the Entropy bot tries to maximize the information gain in each question and only makes a guess when there is no better question, it might sometimes be better to make a guess and win the game. In addition, among all the models, the Normalized has the highest variance and needs the greatest number of questions to guess the hidden entity. Although the strategy of Normalized was analogous to the one of Combined, the only significant difference was the manner in which the values are updated, which should have been the reason why the performance is different.

The second case is when there is a chance to get a wrong answer. From the line plot about the average number of asked questions in all games depicted in Figure 4, it is observable that the Entropy model asks the least number of questions. This does not imply that the bot performs better than the other models in these tournaments. Because when the models reach the question limit, they lose. While the Entropy bot most probably loses after a couple of questions when there are wrong answers given. This is simply because the subgraph it focuses on gets empty due to the contradicting information and hence it will not be able to ask any questions. In addition, there is no significant difference between the models to be observed. Both the lines and the confidence intervals intersect. All individual plots about the number of asked questions in Appendix F.a to F.f depict the same. As for the plots of only won games, the distance between the lines is even smaller and confidence intervals intersect even more (Appendix F.f). Furthermore, one trivial note can be made about the only won games plots. Namely, the higher the chance is of wrong answers the more questions would be needed to guess the hidden entity. We can also observe in the whisker plot in Figure 5 that the EC1 model performs slightly better than all other models in the sense that it is more robust (low variance) and ask the lowest number of questions in most intervals.

As for the T-test results, the difference between the models and the Entropy bot in terms of the number of won games and the number of asked questions overall is significant with 95% and 99% confidence level respectively according to table 2. All p-values in the last rows are less than 0.05 for the number of won games and less than 0.01 for the number of asked questions in all games. This implies that we can reject the Null-Hypothesis, i.e., there is a statistical difference between each model and the Entropy bot in those two metrics. However, there is no significant

difference among the proposed models neither in the number of asked questions nor in the number of won games. In the same table, all p-values retrieved from running the T-test among the models are greater than 0.05. Hence, we cannot reject the Null-Hypothesis in that case. One reason for this similarity in performance among the models could be due to the KG used. Within this KG, the entropy minimizing PO and the highest count PO is in most of the cases the same. Hence, within the given settings, the chosen KG and the taken sample size the difference is not significant.

With this being said, although there is no significant difference between the models, if compared to the Entropy bot the proposed models have the advantage that they can handle misleading answers. The higher the amount of the wrong answers the higher the number of questions needed to predict the right entity. Thus, it is possible to play the 20Q game with both the assumption of an open world and noise in the data. Although most models in most settings were not able to guess the correct entity within the 20 questions, when this problem is reflected in real-life it means that it is possible to predict a hidden entity even when the data is noisy and the knowledge about the world is incomplete. Should the number of asked questions not be bounded (or not matter) all models are guaranteed to guess the hidden entity eventually. It is worth mentioning that all tested models used an efficient scoring technique while some random models have also been introduced. Those random models would have required much more questions to find the entity. Thus, having efficient scoring is sufficient to predict an entity in those settings.

7. Limitations and Future Research:

Several significant and interesting points arose throughout the course of this study, which could be considered as limitations of this paper and areas for investigation in future work. First, three factors introduced randomness in the environment. A workaround has been found for one of them while the other two were kept. It might be possible to account for the other factors in future studies, e.g., adding other parameters to the engine such that it could introduce misleading answers only at the beginning or the end of the game. This will lead to more robust results. Second, the solution in this paper was meant to be as modular and scalable as possible to use any KG, but some dependencies on the KG were found such as the parameters of the Combined and EC1 models that needed a random search. In future approaches, providing some kind of general parameters might prove to be fruitful when such application is used in a different field, such that it does not need any pre-tuning when used on a new KG. This brings us to the third interesting point: testing the proposed models or any similar scoring architectures on KGs from different -maybe specific- fields. Finally, the entries, the questions and all information in the engine were represented as either predicate-object or regular triples from the KG. Making syntactically and grammatically human-readable questions and sentences from that information would make the usage of KGs much more intuitive in many fields of study.

8. Conclusion

In this paper, the 20Q game has been simulated with wrong answers as an instance of the entity prediction task problem with an open-world assumption and noise in the data. The source of the noise is irrelevant to this matter and has been introduced by producing incorrect answers. This study aimed to find an approach that finds the hidden entity in an environment full of noise and assumes that our knowledge about the world is incomplete. Several papers have already investigated this problem and were able to find quite powerful techniques to solve it. However, all of them used cumbersome and computationally heavy solutions. And none of them evaluated it purely on noisy data. In this work, multiple naïve simple approaches were introduced that have as the main idea a scoring index that keeps track of both negatively and positively answered questions without neglecting any entities in the KG. The most efficient approaches were evaluated through running tournaments with multiple levels of chance of getting a wrong answer. The results were converted into visual graphs and used to perform a statistical analysis to find whether there is any statistically significant difference among the proposed approaches. Although there was not, there was compelling evidence that the proposed approaches, that handle noise in the data and act according to the open-world assumption, had superior performance on the regular entropy approach when there is some noise in the data. Thus, having any efficient scoring architecture that keeps track of both negatively and positively answered questions without neglecting any entities in the KG proves to be a sufficient technique that solves the problem.

9. Acknowledgement:

A special thanks for the feedback and guidance that helped in shaping this study provided by Dr. B.B. Kruit and Dr. K.S. Schlobach. The game engine and the basic bots were a collaborative work of myself, D. Thijsse, H. Shahoud, M. Abdullfattah and S.S. Doorn. The full code to the engine can be found on the GitHub repository[1]. The code relevant to this study can be found in a separate branch[2].

[1] https://github.com/hasan-sh/20-questions
[2] https://github.com/hasan-sh/20-questions/tree/Entropy_unreliability

References

1) Variani E., Lahouez K., Bar-Hen A., and Jedynak B., "Non-adaptive policies for 20 questions target localization," in Proc. IEEE International Symposium on Information Theory Proceedings (ISIT). IEEE, 2015, pp. 775 – 778. Retrieved from https://arxiv.org/pdf/1606.09233.pdf
2) Cohen A., and Lake B. M. (2016). Searching large hypothesis spaces by asking questions. In CogSci. Retrieved from https://cogsci.mindmodeling.org/2016/papers/0122/paper0122.pdf
3) Chen X. S., He H., and Davis L. S. (2016, March). Object detection in 20 questions. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1-9). IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/7477562/
4) Tsiligkaridis T., Sadler B. M., and Hero A. O. (2014). Collaborative 20 Questions for Target Localization. IEEE Transactions on Information Theory, 60(4), 2233–2252. Retrieved from https://doi.org/10.1109/tit.2014.2304455
5) numpy.random.randint — NumPy v1.15 Manual. (2018, November 4). Numpy. Retrieved April 23, 2022. Retrieved from https://numpy.org/doc/1.15/reference/generated/numpy.random.randint.html
6) WebCMS3, and Fitzgerald J. (2017). Topic 15: Learning to play "20 Questions." Retrieved from https://github.com/earthtojake/20q/blob/master/report.pdf
7) Burgener R. (1990). ARTIFICIAL NEURAL NETWORK GUESSING METHOD AND GAME (US 2006/0230008 A1). Nishimura. Retrieved from https://patentimages.storage.googleapis.com/41/59/17/cfd8c51aa658a2/US20060230008A1.pdf
8) Wu X., Hu H., Klyen M., Tomita K., and Chen Z., (2018). Q20: Rinna riddles your mind by asking 20 questions. In Japan NLP. Retrieved from https://anlp.jp/proceedings/annual_meeting/2018/pdf_dir/E7-4.pdf
9) Hu H., Wu X., Luo B., Tao C., Xu C., Wu W., and Chen Z. (2018). Playing 20 question game with policy-based reinforcement learning. arXiv preprint arXiv:1808.07645
10) Dey A., Jain H. K., Pandey V. K., and Chakraborty T. (2019). All It Takes is 20 Questions!: A Knowledge Graph Based Approach. arXiv preprint arXiv:1911.05161. Retrieved from https://arxiv.org/pdf/1911.05161v1.pdf
11) Pellissier T. T., Weikum G., and Suchanek F. (2020, May). Yago 4: A reason-able knowledge base. In European Semantic Web Conference (pp. 583-596). Springer, Cham. Retrieved from https://suchanek.name/work/publications/eswc-2020-yago.pdf
12) Shannon, C. E. (1948). A Mathematical Theory of Communication. Bell System Technical Journal, 27(3), 379–423. Retrieved from https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

13) Jaynes E. T. (1957). Information Theory and Statistical Mechanics. Physical Review, 106(4), 620–630. Retrieved from https://doi.org/10.1103/physrev.106.620

14) Kwak, Gyu S., and Kim J. H.. 2017. "Central Limit Theorem: The Cornerstone of Modern Statistics." Korean Journal of Anesthesiology 70 (2): 144–56. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5370305/#:~:text=Therefore%2C%20the%20central%20limit%20theorem,regardless%20of%20the%20population%20distribution.

15) Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press.

Appendix

A. Random Search:

a. Combined: settings:

- Machine mode (Answerer Bot vs Combined Bot)
- Unreliability level = 0
- To minimize randomness the same entity has been picked throughout all runs. Entity picked: Taylor Swift

Values to choose from:

* Weights [Score index weight, Count index weight]:
[0.1,0.9], [0.2,0.8], [0.3,0.7], [0.4,0.6], [0.5,0.5], [0.6,0.4], [0.7,0.3], [0.8,0.2], [0.9,0.1]

* Increment factor upon a "yes": 10 random values between 100 and 100000

* Decrement factor upon a "no": 10 random values between 100 and 100000

b. EC1: settings:

- Machine mode (Answerer Bot vs EC1 Bot)
- Unreliability level = 0
- To minimize randomness the same entity has been picked throughout all runs. Entity picked: Taylor Swift

Values to choose from:

* Weights [Score index weight, Entropy index weight]:
[0.1,0.9], [0.2,0.8], [0.3,0.7], [0.4,0.6], [0.5,0.5], [0.6,0.4], [0.7,0.3], [0.8,0.2], [0.9,0.1]

B. Models comparison:

	Basic	***Combined***	***Normalized***	***EC1***	***EC2***
Number of indices	1	3	1	3	1
Index	Score	Score, Count, Combined	Normalized	Score, Entropy, Combined	Entropy
initialization	Score ➔ 1's	Score ➔ 1's Count ➔ Counts of PO's Combined ➔ 0.9Score + 0.1Count	Normalized ➔ normalized counts of PO's	Score ➔ 1's Entropy ➔ entropies of PO's Combined➔ 0.9Score + 0.1Entropy	Entropy ➔ entropies of PO's
update	Score +1 -1	Score +39730 -2240	Normalized +1 -1	Score +1 -1	Entropy +1 -1
Index used to pick next question	Score	Combined	Normalized	Combined	Entropy

C. Raw results:

a. Grey's anatomy:

i. Number of won games (out of 30):

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	30	30	21	16	11	8	3
Normalized	22	19	20	12	13	3	5
EC2	30	27	24	23	14	11	7
EC1	30	25	18	15	9	10	5
Entropy	30	0	0	0	0	0	0

ii. Average number of asked questions in ALL games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	133	186	348	388	440	448	486
Normalized	346	374	375	403	401	477	473
EC2	117	206	255	302	395	392	457
EC1	91	200	297	340	420	397	460
Entropy	117	68	27	43	19	19	18

iii. Average number of asked questions in WON games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	133	186	283	291	336	303	360
Normalized	290	300	313	258	271	266	337
EC2	117	173	193	242	276	205	315
EC1	91	140	161	179	233	191	261
Entropy	117	NaN	NaN	NaN	NaN	NaN	NaN

b. Elizabeth II:

i. Number of won games (out of 30):

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	30	25	21	21	18	10	12
Normalized	30	23	26	24	15	11	11
EC2	30	23	24	18	19	11	6
EC1	30	22	19	11	10	10	6
Entropy	30	8	2	0	0	0	0

ii. Average number of asked questions in ALL games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	38	156	221	236	274	401	361
Normalized	40	180	142	219	339	388	378
EC2	37	168	165	275	310	367	421
EC1	37	176	240	377	384	389	419
Entropy	26	81	101	35	37	58	18

iii. Average number of asked questions in WON games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	38	87	101	123	123	203	152
Normalized	40	82	87	148	178	194	167
EC2	37	66	81	125	201	136	106
EC1	37	58	89	164	152	168	94
Entropy	26	26	26	NaN	NaN	NaN	NaN

c. Robin Williams:

i. Number of won games (out of 30):

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	30	26	22	21	18	12	12
Normalized	30	25	26	19	15	10	10
EC2	30	21	18	17	13	11	12
EC1	30	24	19	17	12	8	10
Entropy	30	10	3	1	0	0	0

ii. Average number of asked questions in ALL games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	27	120	250	230	311	382	387
Normalized	26	135	147	255	328	401	411
EC2	26	184	130	270	317	363	366
EC1	28	129	245	292	356	397	410
Entropy	24	106	90	37	21	34	37

iii. Average number of asked questions in WON games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	27	61	160	114	186	204	217
Normalized	26	62	93	112	156	204	234
EC2	26	49	57	94	77	126	165
EC1	28	36	98	132	140	113	230
Entropy	24	24	24	24	NaN	NaN	NaN

d. Fast & Furious 6:

i. Number of won games (out of 30):

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	30	29	25	24	19	22	23
Normalized	30	27	26	18	14	10	11
EC2	30	26	20	18	16	10	12
EC1	30	24	24	19	18	13	10
Entropy	30	2	0	0	0	0	0

ii. Average number of asked questions in ALL games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25	98	155	162	248	222	256
Normalized	25	104	140	267	340	400	384
EC2	25	116	217	269	321	386	372
EC1	11	123	122	222	245	348	377
Entropy	43	34	31	27	29	22	16

iii. Average number of asked questions in WON games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25	84	86	77	102	122	182
Normalized	25	60	84	111	158	200	184
EC2	25	57	75	115	164	159	180
EC1	11	28	28	61	75	149	130
Entropy	43	43	NaN	NaN	NaN	NaN	NaN

e. 10,000 Hours:

i. Number of won games (out of 30):

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	30	26	19	16	16	4	6
Normalized	30	27	21	18	13	17	8
EC2	30	19	18	16	9	6	7
EC1	30	22	15	8	13	8	6
Entropy	30	1	0	0	0	0	0

ii. Average number of asked questions in ALL games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	19	96	217	319	295	465	424
Normalized	18	109	177	249	306	271	411
EC2	18	196	217	286	385	436	420
EC1	19	155	268	398	335	398	435
Entropy	54	140	93	70	32	18	16

iii. Average number of asked questions in WON games:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	19	34	54	161	116	241	120
Normalized	18	65	39	81	52	96	166
EC2	18	20	28	99	118	181	158
EC1	19	29	35	117	119	117	177
Entropy	54	55	NaN	NaN	NaN	NaN	NaN

D. Confidence Intervals number of asked questions in ALL games:

a. Grey's anatomy:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	114, 152	149, 223	288, 409	338, 438	397, 482	409, 486	468, 504
Normalized	300, 393	324,423	329,421	350,456	349,453	446,507	449,497
EC2	102, 132	162, 250	202, 307	250, 354	345, 446	334, 450	423, 490
EC1	74, 108	143, 256	231, 363	275, 405	368, 472	339, 454	425, 495
Entropy	117, 118	27, 108	20, 34	9, 77	11, 26	13, 25	15, 22

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	107. 159.	136. 236.	267. 430.	321. 456.	383. 497.	395. 500.	462. 510.
Normalized	283,409	307,440	313,437	332,475	331,471	436,517	441,505
EC2	96, 138	146, 266	184, 325	231, 373	327, 463	313, 470	412, 502
EC1	69, 114	124, 276	208, 386	252, 427	350, 490	319, 474	413, 507
Entropy	116, 119	13, 122	18, 36	-3, 89	9, 29	10, 27	14, 23

b. Elizabeth II:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	35, 41	91, 220	146, 296	160, 312	200, 348	339, 463	293, 429
Normalized	37, 43	106, 254	85, 199	152, 285	369, 409	322, 454	309, 447
EC2	33, 40	96, 240	96, 234	198, 352	242, 378	296, 437	361, 482
EC1	34, 41	100, 252	161, 318	310, 444	317, 451	325, 454	356, 481
Entropy	NaN	27, 135	37,164	1, 68	4, 71	8, 109	13, 22

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	34, 42	68, 243	120, 322	133, 338	174, 374	318, 484	269, 453
Normalized	35, 44	80, 279	65, 219	129, 308	244, 434	299, 477	285, 471
EC2	32, 41	70, 265	73, 257	171, 379	219, 402	272, 461	339, 503
EC1	33, 42	73, 278	134, 345	287, 467	294, 474	302, 476	335, 503
Entropy	NaN	8, 154	16, 186	-10, 79	-7, 82	-10, 127	12, 24

c. Robin Williams:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 29	60, 179	182, 319	158, 302	242, 381	316, 448	324, 449
Normalized	24, 27	68, 203	83, 212	178, 331	253, 403	343, 460	354, 469
EC2	24, 28	104, 265	64, 197	188, 351	235, 398	290, 435	296, 436
EC1	25, 31	58, 201	166, 325	217, 367	283, 429	328, 465	355, 465
Entropy	24,25	39, 173	29, 152	4, 70	16, 25	1, 67	4, 70

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	24, 30	39, 200	158, 343	133, 327	218, 405	293, 471	303, 471
Normalized	23, 28	44, 226	60, 234	151, 358	227, 429	322, 481	334, 489
EC2	24, 29	77, 292	41, 220	160, 380	207, 427	265, 460	271, 461
EC1	24, 32	33, 225	138, 353	191, 393	258, 454	304, 489	336, 485
Entropy	24,25	16, 196	7, 173	-7, 81	14, 27	-11, 79	-7, 82

d. Fast & Furious 6:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 25	51, 146	89, 220	91, 232	171, 325	152, 293	194, 319
Normalized	25, 25	52, 156	83, 197	191, 343	269, 412	340, 461	316, 452
EC2	25, 25	57, 174	140, 294	193, 345	246, 395	319, 453	307, 437
EC1	11, 12	51, 195	50, 194	140, 304	163, 327	278, 418	308, 446
Entropy	NaN	26, 42	20, 42	19, 36	10, 48	14, 30	13, 18

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 25	34, 162	67, 243	67, 257	144, 352	128, 317	172, 341
Normalized	25, 26	34, 174	63, 216	164, 369	244, 437	318, 482	293, 476
EC2	25, 26	37, 195	113, 320	167, 371	220, 421	296, 477	284, 460
EC1	11, 12	25, 220	25, 220	112, 332	135, 355	253, 443	283, 470
Entropy	NaN	23, 45	16, 46	16, 39	3, 55	11, 33	13, 19

e. 10,000 Hours:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	19, 20	33, 159	133, 302	246, 392	217, 373	424, 507	362, 486
Normalized	17, 18	49, 169	95, 260	166, 331	220, 392	191, 352	347, 475
EC2	17, 18	108, 284	129, 305	202, 370	311, 459	378, 494	357, 483
EC1	19, 20	75, 235	179, 356	329, 467	256, 414	329, 467	380, 491
Entropy	NaN	69, 212	33, 154	21, 119	8, 56	15, 21	11, 20

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	18, 20	12, 181	104, 331	220, 418	189, 401	410, 521	341, 507
Normalized	17, 18	28, 190	66, 289	137, 360	190, 422	163, 379	325, 497
EC2	17, 18	77, 314	98, 335	173, 399	286, 485	358, 514	335, 505
EC1	19, 20	47, 263	148, 387	305, 490	228, 442	305, 490	360, 511
Entropy	NaN	44, 237	12, 175	4, 136	0, 64	13, 22	10, 22

E. Confidence Intervals number of asked questions in WON games:

a. Grey's anatomy:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	114. 152.	149. 223.	214. 353.	231. 351.	244. 428.	212. 395.	161. 560.
Normalized	248,332	246,355	264,361	180,336	199,344	-98,629	285,389
EC2	102, 132	146, 200	163, 224	199, 284	212, 339	138, 273	231, 399
EC1	74, 108	110, 169	126, 197	131, 228	144, 323	138, 244	180, 342
Entropy	117, 118	NaN	NaN	NaN	NaN	NaN	NaN

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	107. 159.	136. 236.	188. 379	208. 374.	205. 467.	168. 438.	-100. 821.
Normalized	232,348	226,375	246,379	148,368	170,373	-573,1105	251,423
EC2	96, 138	137, 209	152, 234	184, 299	187, 364	109, 301	188, 442
EC1	69, 114	100, 180	113, 210	112, 247	103, 363	114, 267	126, 395
Entropy	116, 118	NaN	NaN	NaN	NaN	NaN	NaN

b. Elizabeth II:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	35, 41	53, 120	60, 143	67, 179	81, 165	95, 311	93, 210
Normalized	37, 43	41, 124	63, 111	98, 198	104, 252	86, 302	68, 266
EC2	33, 40	42, 91	49, 114	68, 183	134, 267	66, 206	41, 170
EC1	34, 41	36, 80	52, 126	85, 244	69, 236	81, 255	30, 157
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	34, 42	41, 132	44, 158	46, 199	65, 181	48, 358	69, 234
Normalized	35, 44	26, 139	54, 120	81, 216	75, 281	40, 348	26, 308
EC2	32, 41	33, 99	37, 125	47, 204	110, 291	37, 235	4, 207
EC1	33, 42	27, 88	38, 139	51, 278	32, 272	43, 294	-6, 193
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

c. Robin Williams:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 29	39, 84	107, 212	77, 151	121, 250	103, 306	125, 308
Normalized	24, 27	29, 95	49, 137	64, 161	78, 234	117, 292	124, 344
EC2	24, 28	26, 73	32, 82	41, 147	32, 122	53, 198	75, 255
EC1	25, 31	21, 52	49, 148	79, 185	60, 220	14, 212	138, 322
Entropy	24,25	NaN	NaN	NaN	NaN	NaN	NaN

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	24, 30	31, 91	88, 231	63, 165	97, 274	61, 348	87, 346
Normalized	23, 28	18, 106	34, 153	46, 179	48, 264	79, 330	76, 392
EC2	24, 29	17, 81	23, 90	21, 167	14, 140	23, 229	38, 291
EC1	24, 32	16, 57	30, 166	59, 205	27, 253	-34, 260	98, 363
Entropy	24,25	NaN	NaN	NaN	NaN	NaN	NaN

d. Fast & Furious 6:

i. 95% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 25	45, 124	50, 122	40, 114	60, 145	79, 164	134, 231
Normalized	25, 25	44, 76	63, 106	71, 151	88, 229	103, 298	75, 294
EC2	25, 25	42, 71	56, 94	71, 159	87, 240	64, 254	106, 255
EC1	11, 12	17, 39	17, 39	38, 84	38, 113	89, 210	62, 198
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

ii. 99% CI:

	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	25, 25	31, 138	38, 134	27, 127	44, 161	64, 179	116, 248
Normalized	25, 26	39, 81	55, 114	56, 166	60, 257	61, 340	29, 340
EC2	25, 26	37, 76	49, 101	54, 176	58, 270	23, 295	76, 285
EC1	11, 12	13, 43	13, 42	29, 93	24, 127	65, 234	32, 228
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

e. 10,000 Hours:

i. 95% CI:

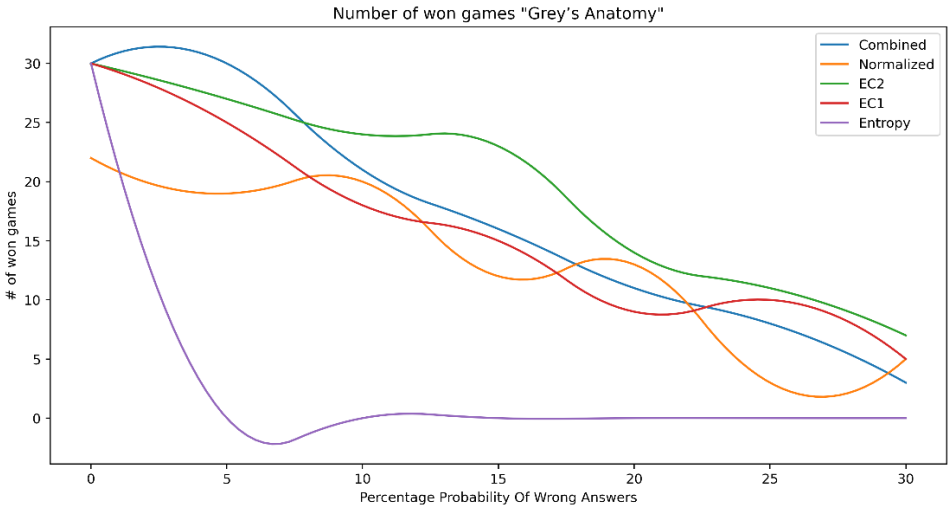
	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	19, 20	13, 55	19, 88	91, 230	58, 174	-80, 562	-32, 272
Normalized	17, 18	27, 103	9, 69	34, 129	11, 93	47, 145	24, 308
EC2	17, 18	18, 21	20, 35	28, 169	-10, 246	-34, 395	-8, 324
EC1	19, 20	10, 49	20, 51	7, 227	35, 203	8, 226	-3, 358
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

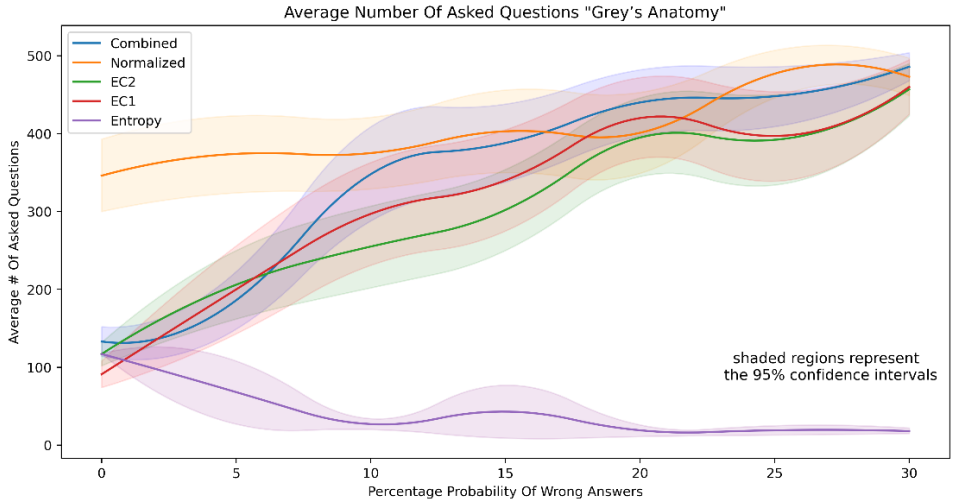
ii. 99% CI:

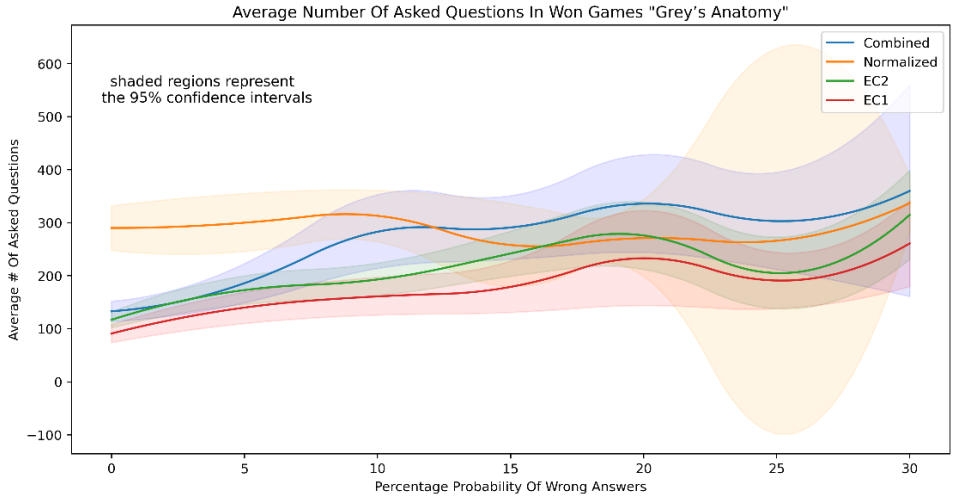
	0%	*5%*	*10%*	*15%*	*20%*	*25%*	*30%*
Combined	18, 20	6, 63	7, 101	64, 257	36, 196	-348, 830	-118, 359
Normalized	17, 18	14, 117	-2, 80	16, 146	-6, 110	29, 163	-44, 376
EC2	17, 18	17, 22	18, 38	2, 196	-69, 305	-156, 517	-94, 409
EC1	19, 20	3, 56	14, 57	-46, 280	1, 237	-45, 279	-106, 460
Entropy	NaN	NaN	NaN	NaN	NaN	NaN	NaN

F. Plots:

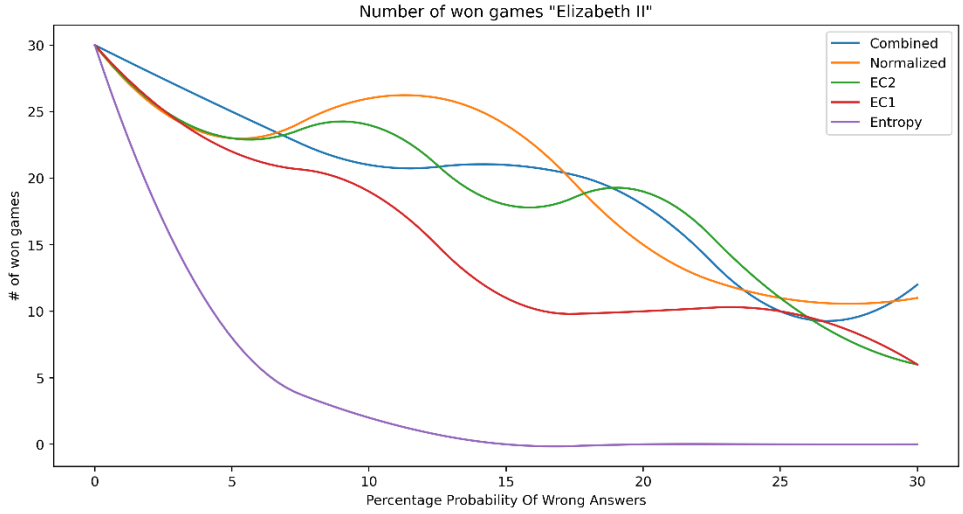
a. Grey's anatomy:

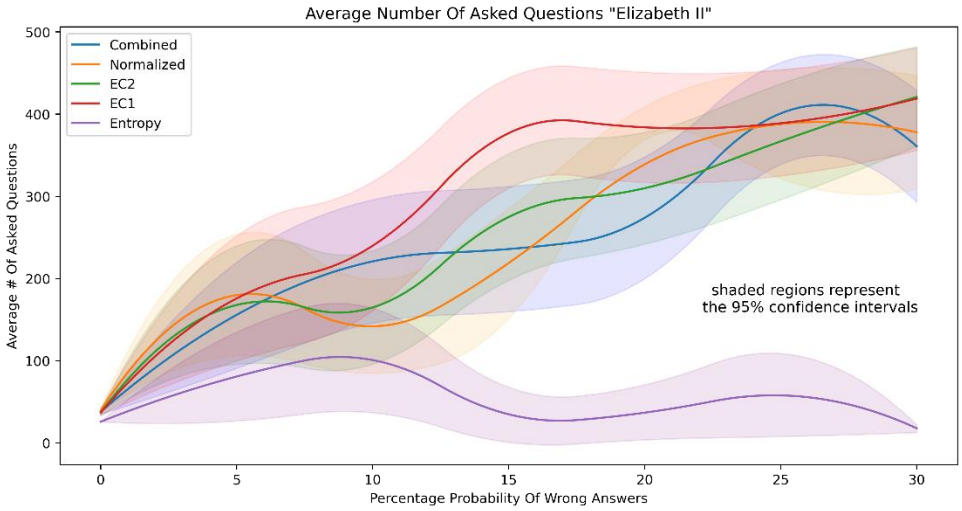


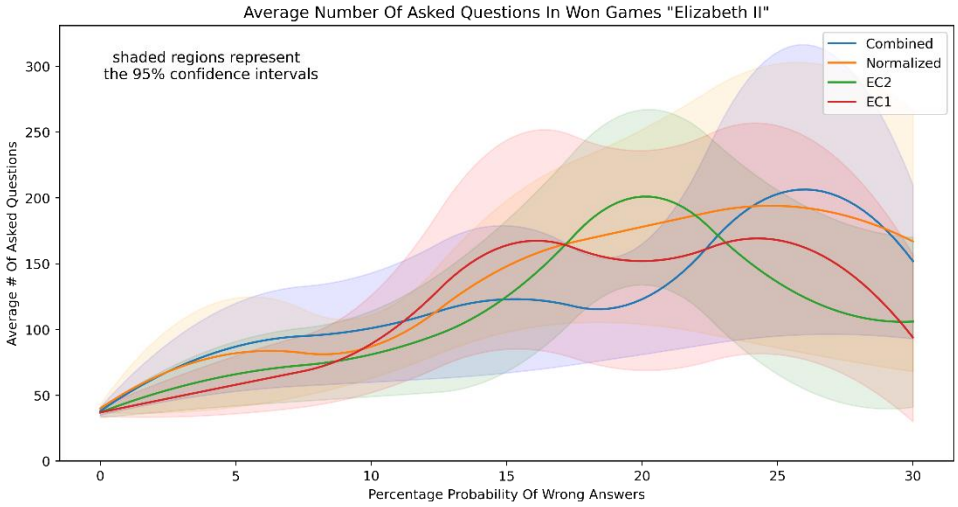




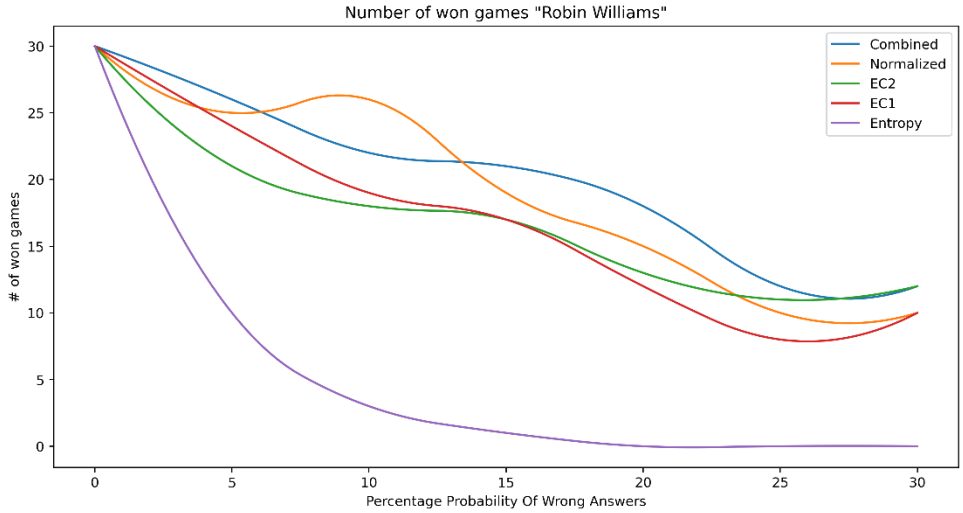
b. Elizabeth II:

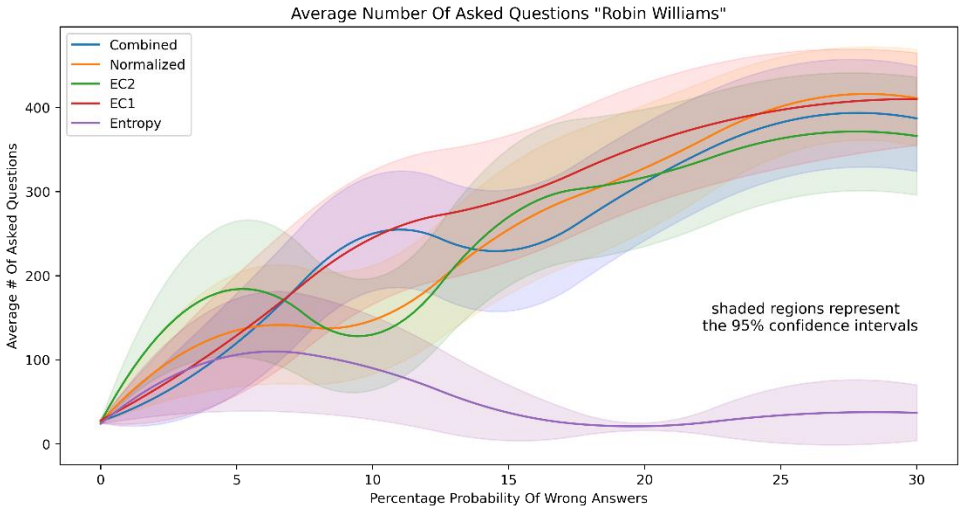


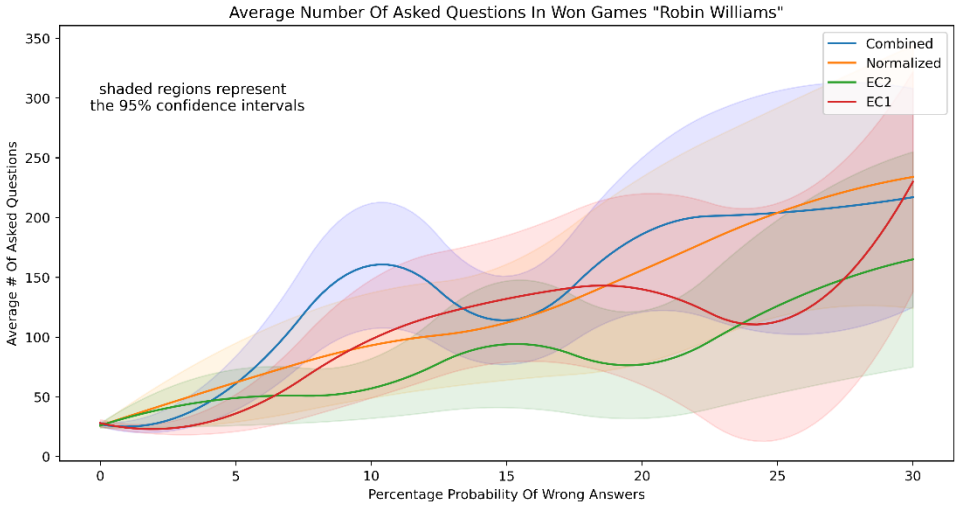




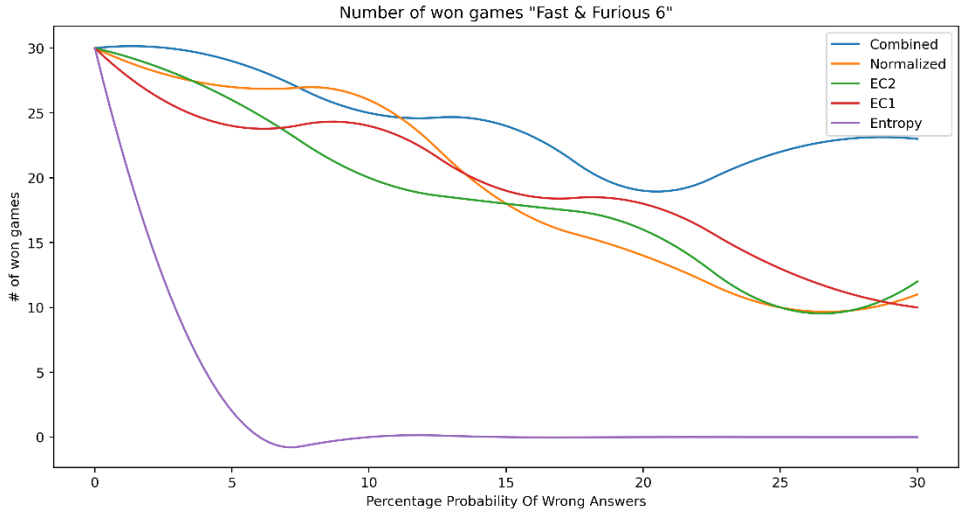
c. Robin Williams:

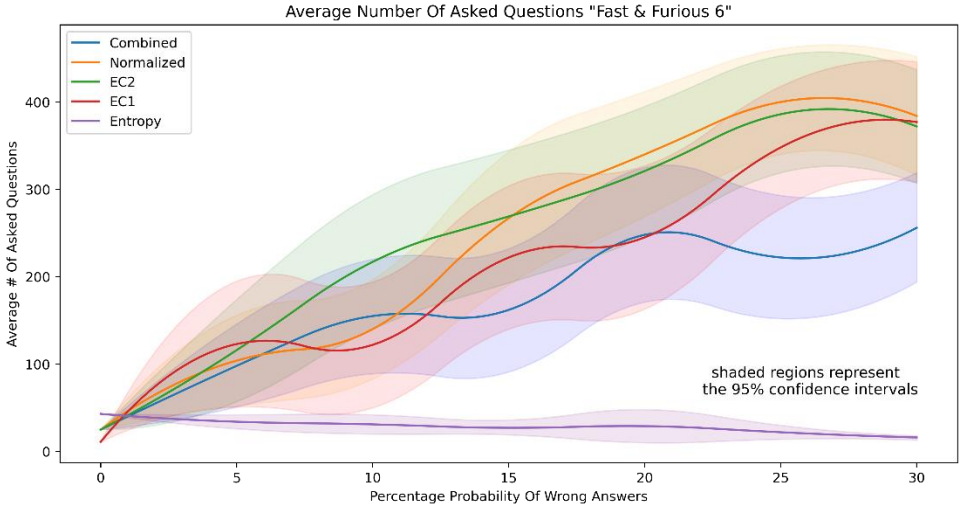


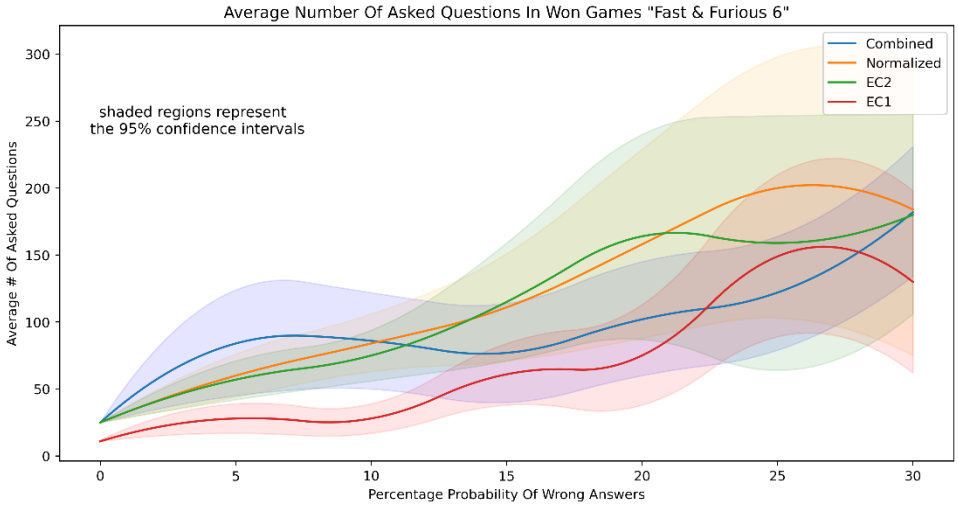




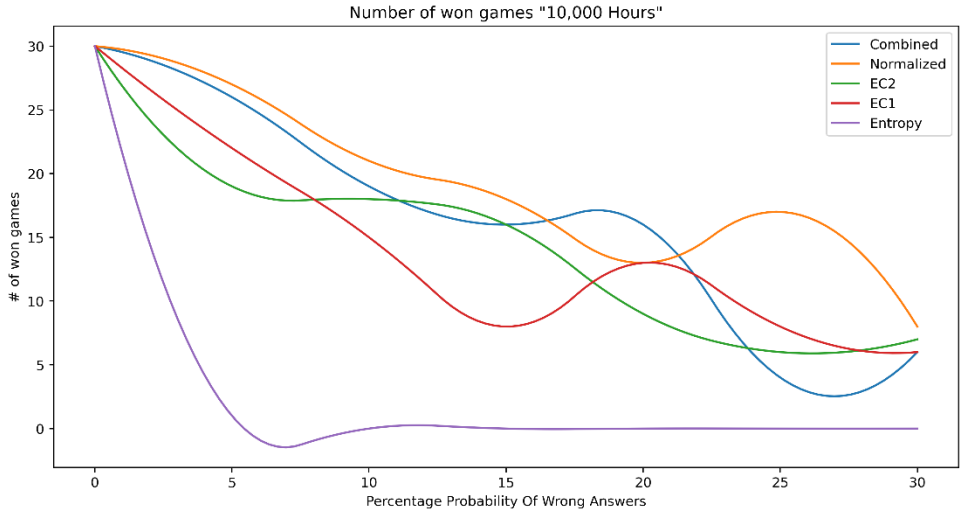
d. Fast & Furious 6:

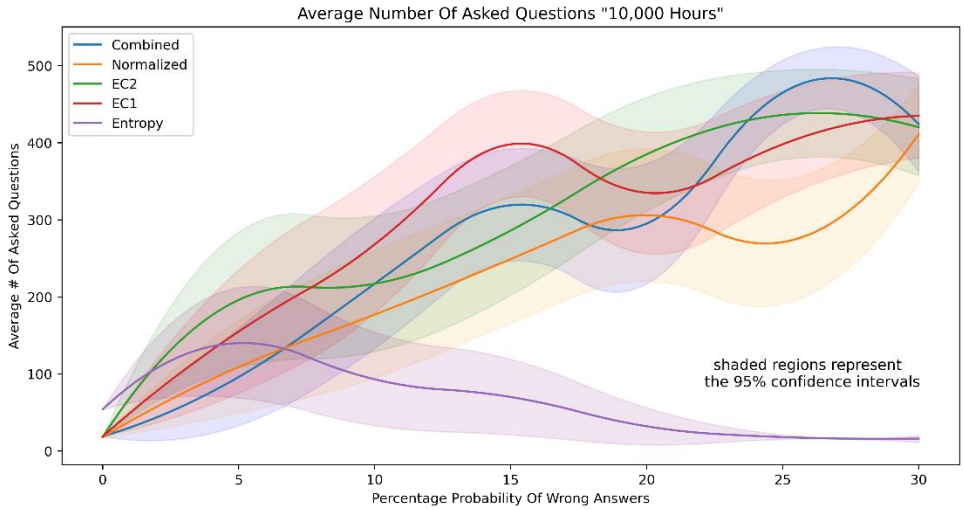


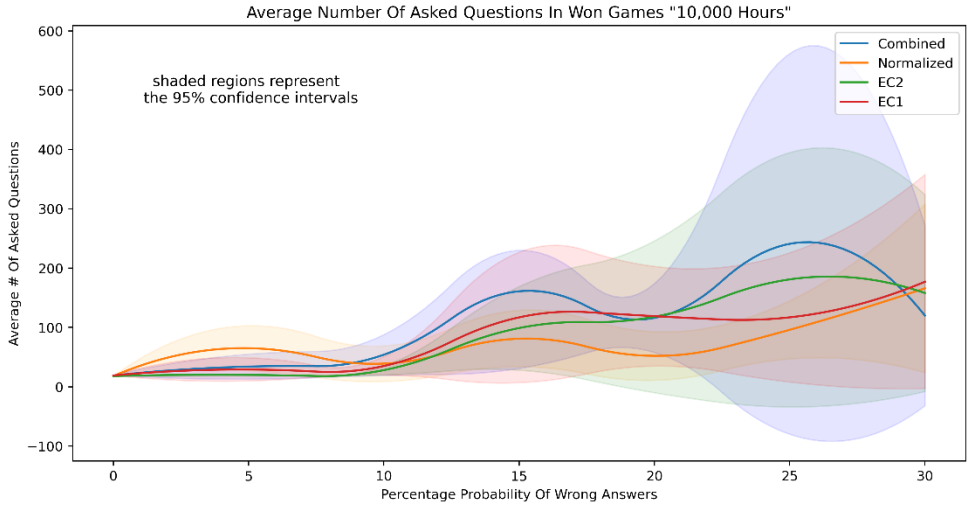




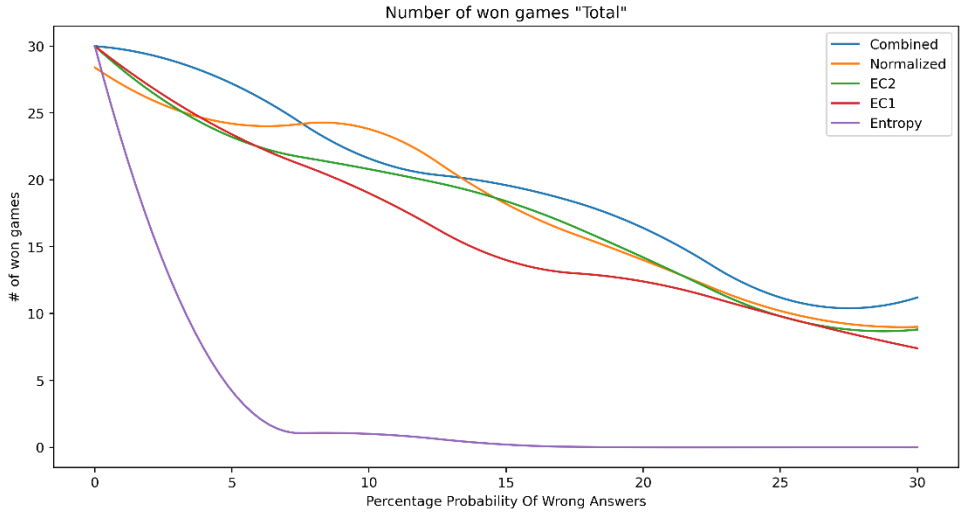
e. 10,000 Hours:

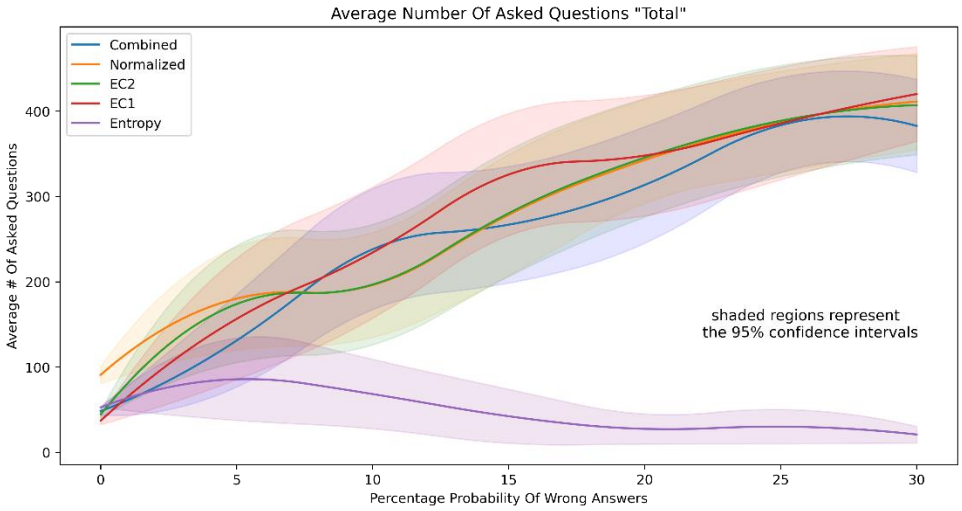


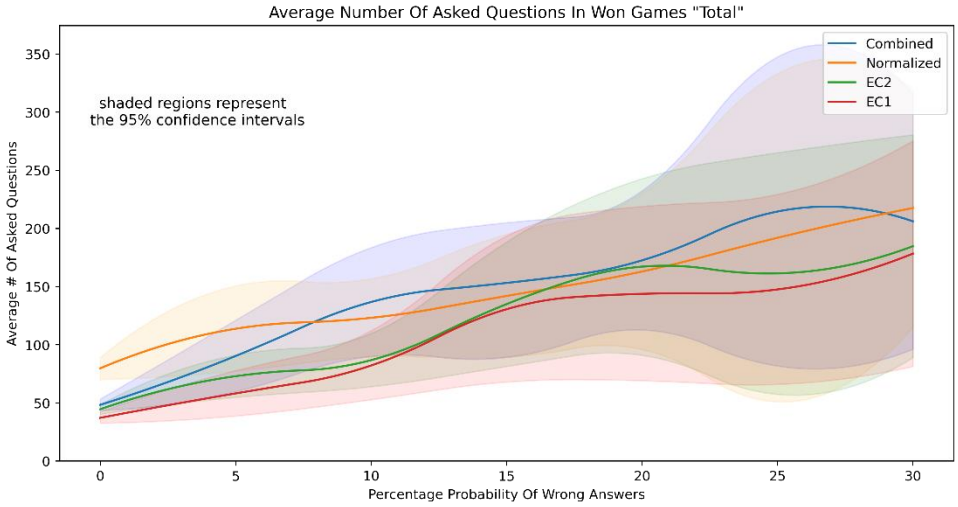

Average Number Of Asked Questions "10,000 Hours"
Combined
Normalized
EC2
EC1
Entropy
Average # Of Asked Questions
Percentage Probability Of Wrong Answers
shaded regions represent
the 95% confidence intervals


Average Number Of Asked Questions In Won Games "10,000 Hours"
shaded regions represent
the 95% confidence intervals
Combined
Normalized
EC2
EC1
Average # Of Asked Questions
Percentage Probability Of Wrong Answers

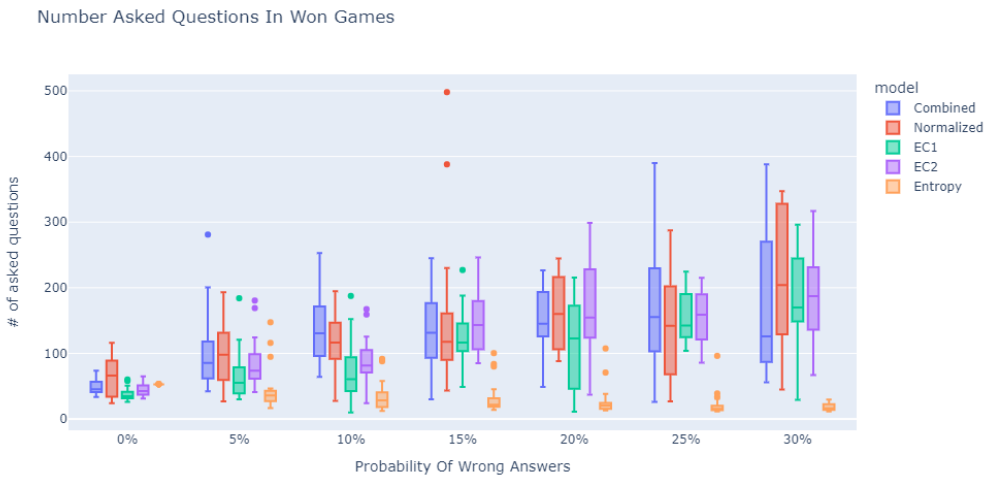
f. Total:


Number of won games "Total"
Combined
Normalized
EC2
EC1
Entropy
# of won games
Percentage Probability Of Wrong Answers





Link for the interactive plot: https://chart-studio.plotly.com/~HishamAlkaed/1/#plot



Link for the interactive plot: https://chart-studio.plotly.com/~HishamAlkaed/5/#plot

g. Additional Plots:

G. T-test results:

a. Number of won games: P-value

	Combined	*Normalized*	*EC2*	*EC1*	*Entropy*
Combined	1				
Normalized	0.74045	1			
EC2	0.67459	0.92813	1		
EC1	0.47494	0.69198	0.75832	1	
Entropy	0.01349	0.02296	0.02671	0.04616	1

b. The number of asked questions in ALL games:

	Combined	*Normalized*	*EC2*	*EC1*	*Entropy*
Combined	1				
Normalized	0.79258	1			
EC2	0.88257	0.91617	1		
EC1	0.77631	0.96717	0.89079	1	
Entropy	0.00113	0.000387	0.001089	0.00107	1

c. The number of asked questions in WON games:

	Combined	*Normalized*	*EC2*	*EC1*
Combined	1			
Normalized	0.96457	1		
EC2	0.44326	0.36478	1	
EC1	0.26986	0.19970	0.71277	1