

# Learning Consensus Opinion: Mining Data from a Labeling Game\*

Paul N. Bennett  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052  
pauben@microsoft.com

David Maxwell  
Chickering  
Microsoft Live Labs  
One Microsoft Way  
Redmond, WA 98052  
dmax@microsoft.com

Anton Mityagin  
Microsoft Live Labs  
One Microsoft Way  
Redmond, WA 98052  
mityagin@microsoft.com

## ABSTRACT

We consider the problem of identifying the consensus ranking for the results of a query, given preferences among those results from a set of individual users. Once consensus rankings are identified for a set of queries, these rankings can serve for both evaluation and training of retrieval and learning systems. We present a novel approach to collecting the individual user preferences over image-search results: we use a collaborative game in which players are rewarded for agreeing on which image result is best for a query. Our approach is distinct from other labeling games because we are able to elicit directly the preferences of interest with respect to image queries extracted from query logs. As a source of relevance judgments, this data provides a useful complement to click data. Furthermore, the data is free of positional biases and is collected by the game without the risk of frustrating users with non-relevant results; this risk is prevalent in standard mechanisms for debiasing clicks. We describe data collected over 34 days from a deployed version of this game that amounts to about 18 million expressed preferences between pairs. Finally, we present several approaches to modeling this data in order to extract the consensus rankings from the preferences and better sort the search results for targeted queries.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*selection process*

## General Terms

Algorithms, Performance

## Keywords

preference judgments, learning preferences

## 1. INTRODUCTION

In many information retrieval applications, we desire to rank items in relation to an information need or task. The

\*This is a preprint of an Article accepted for publication in *The 18th International World Wide Web Conference* ©2009 International World Wide Web Conference Committee.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

WWW 2009, April 20–24, 2009, Madrid, Spain.  
ACM 978-1-60558-487-4/09/04.

purpose may be to display directly the ranked items to a user of the system or to use the ranking as an intermediate step in another algorithm. Applications of ranking prediction include image search, ad hoc web search, recommender systems, ordering e-mail and a host of other problems. Often in these tasks, little attention is paid to the challenge of how to reliably identify what ranking should serve as ground truth for training and evaluation purposes. That is, what ranking of the retrieved set would best satisfy all users?

While the standard approach to ranking has been to rank items according to their relevance [14], and in particular *topical* relevance [4], an item may be preferred by a user based on a variety of other characteristics including quality, authoritativeness, and readability; these characteristics may be seen as defining a broader notion of relevance that incorporates the context of a task. For image search—which is the primary ranking task considered in this paper—a user may prefer an image because of its focus, composition, artistry, or a variety of other dimensions.

Ranking systems are typically constructed with data consisting of explicit relevance judgments for a set of training items. One approach to getting this training data is to elicit relevance judgments from a small number of “expert” judges or editors in a controlled setting. A difficulty of this approach is that, due to query ambiguity and personal preferences, it may be difficult for any single person except the query issuer to accurately judge the relevance of results.

An alternative approach to data collection is to sample judgments from a large population resembling the user base. This has motivated much of the research on mining click data. While click data has undeniable value, most notably because it comes from the issuer of the query, it also has potential weaknesses. In particular, items that have not been displayed cannot be clicked, and the lack of a click is often not informative because the search page itself may satisfy the information need. This latter problem is even more of a concern in image search because the search result page typically consists of actual images (potentially scaled down). Furthermore, if the designers of a ranking system experiment with the live system by (*e.g.*) swapping items or placing potentially non-relevant results in the top of the list [12, 13], there is a risk of frustrating the user and prompting him to switch search engines.

In this work, we attempt to solve the data-acquisition problem in the domain of image search with a social labeling game [19]. The game, which we describe in detail in Section 3.1, pairs two participants on the internet who are shown a

sequence of queries and corresponding sets of image results; they are both asked to choose the best image for each query, and every time they agree they are awarded credits. After collecting enough credits, they may turn the credits in for prizes. Assuming participants are primarily seeking credits or prizes, the incentive mechanism encourages users to give their actual opinion of the best image because it is a good way to achieve agreement with high likelihood.

We study several probabilistic models that can be used to convert the preference data that results from game play into a set of relevance scores for the items; these relevance scores can be used for evaluating an existing system, training a new ranking algorithm, or simply improving the retrieval set for the particular queries in the data.

In the remainder of the paper, we first give an overview of related work, including a more in-depth discussion of why current social labeling games fall short, and a survey of preference-learning models of particular relevance to this work. We then present a detailed description of our labeling game. Next, we present several methods for mining the resulting preference data to learn rankings. Then, we present our results and discuss their implications. Finally, we conclude with a discussion of interesting future directions.

## 2. RELATED WORK

There have been a number of human-computation games developed for the purpose of collecting data. Arguably the most successful such game is the ESP Game [19], where as a result of game play, participants produce descriptions of images. Others include Peekaboom [20], where the players produce data about where objects are located in an image, and TagATune [10] where players produce descriptions of sounds and music.

The image-description data produced by the ESP game could in principle be used to help infer ranking quality; for example, if players annotate one image with “dog” more often or sooner than a second image, we might reason that the first image is a better result for the query “dog”. One problem with this approach is that the descriptions that are associated with an image are determined by the players and not the game designer. A second problem is that, because players can earn more points in the ESP game if they match quickly, there is incentive to provide very short and obvious descriptions that may not be very informative (*e.g.*, prevalent colors in the images). That is, it is not clear that the task of trying to get a match with a partner on tags (which leads one to give common tags) maps closely to asking for the tags for which this image is one of the most relevant results. Instead, the ESP game is most suited for addressing the indexing aspect of search (identifying a large set of relevant items) while our approach targets the ranking aspect (how to only present a small subset of very relevant items out of a large number of matching items).

While obtaining *absolute* relevance judgments over a particular image/query pair may seem like a better way to obtain judgments, Carterette et al. [2] demonstrate that absolute judgments are less reliable in terms of agreement with other assessors than *pairwise preferences*, *i.e.*, relative comparisons between two items. In Section 4.2, we discuss several methods that are based on pairwise preferences.

In terms of preference learning, Carterette et al. [2] address how to infer a ranking given only a linear number of preferences from a single assessor but do not address how

to use preferences across a population of users. Fürnkranz and Hüllermeier [7] present a study on breaking label preference ranking (*e.g.*, predicting order of preferred classes in a multiclass classification problem) into a series of binary classification tasks [7]. The primary issue that concerns us here, however, is one of instance-preference ranking. Chu and Ghahramani [3] propose a Gaussian-process model applicable to both instance and label preference learning. The model is most similar to the Go model we present but does not have some of the flexibility to model player abilities.

If we consider the challenge proposed here as learning to predict the preferences of all users given a small sample of users, one natural solution is to consider averaging the preferences of the users in the training set and then extracting the optimal ranking for the averaged preferences to predict on the testing set. Cohen, Schapire, and Singer [5] demonstrated that the problem of extracting the optimal ranking is NP-complete, and whereas they provide useful approximation algorithms for this, we are more focused on the second question of how well this generalizes to the full population. Similarly, as alternate models to those presented here, we could consider this a task of rank aggregation [6] and apply rank aggregation methods to extract the consensus ranking. Whereas a number of these approaches would require rankings from users instead of preferences, these rankings could easily be extracted from the preferences first (*e.g.* run quick-sort for each user). We note that in our setting, we do not ask directly for rankings because the increased complexity in the task both increases noise in response and interferes with the fast-paced excitement of the game. Finally, Joachims [9] proposes ranking SVMs for preference learning and applies them to preference learning using click data. Our work is most similar in flavor to this, but we deviate in both the source of our data and the scale.

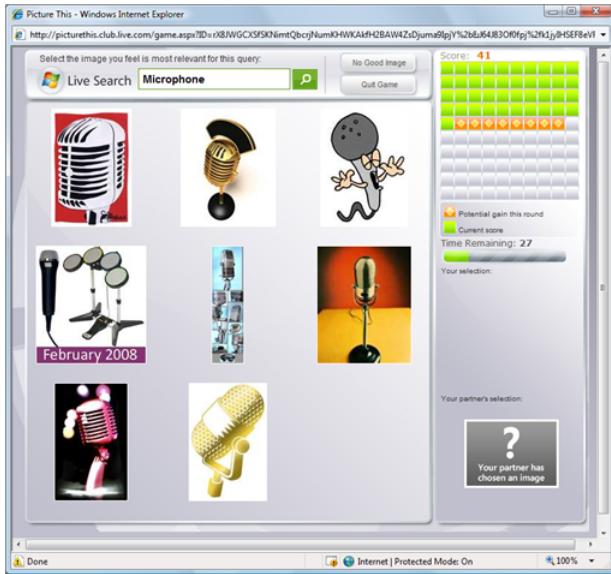
In contrast to previous work on social labeling games, our work is the first to use queries drawn from search logs to drive image rankings. As a result, the language distribution actually seen on the search engine drives the labeling. Furthermore, ours is the first attempt to study the suitability of large-scale preference data as a signal to augment other sources of relevance information such as click data and assessor judgments. In particular, our approach is unique in its ability to obtain information like click data but with positional biases removed and with no risk of frustrating users when given non-relevant results. Finally, from a learning perspective, we contribute a systematic study of a number of preference models.

## 3. PREFERENCE COLLECTION AS GAME

*Picture This* is a collaborative game where pairs of players are rewarded for agreeing on the best result for image-search queries. In this section, we describe the game in detail. Currently the game can be accessed online via [http://club.live.com/Pages/Games/GameList.aspx?game=Picture\\_This](http://club.live.com/Pages/Games/GameList.aspx?game=Picture_This).

### 3.1 Game Play

When a user starts the *Picture This* game he is first paired with a random partner. If no human player is available within a few seconds of the player starting the game, the player is assigned a robot as a partner; we describe robots in detail in Section 3.3. The two players then proceed through a set of rounds, working together for either two minutes or until they reach 100 points, whichever comes first. The



**Figure 1:** The *Picture This* game interface. The query shown at the top is “Microphone” with eight candidate images. The green boxes in the grid at the top right show how many of the 100 possible points the player has earned; the orange boxes with diamonds show how many points will be earned on agreement (*i.e.*,  $k$ ). In this case, the partner has made his selection (indicated in lower right corner) and is waiting for this player to choose.

players are synchronized such that the next round starts only when both players have completed the current round.

In each round, the two players are shown a query and two different random permutations of the same set of  $k$  images. The images are permuted both to eliminate positional bias in the preference data and, as we discuss in Section 3.4, to mitigate against fraud.  $k$  is initially set to three and changes throughout the game. Each player selects the image that, in his opinion, best matches the query. If the two players agree on the best image, they are both awarded  $k$  points and  $k$  is incremented by one if  $k < 9$ . If the players disagree on the best image, they are awarded zero points and  $k$  is decremented by one if  $k > 2$ . Thus the number of images displayed varies from 2 to 9 depending on player actions: when users agree, the number of images displayed (and the difficulty of the game) increases, and when users disagree, the number of images displayed decreases. Adapting the game difficulty to a player’s performance allows us to both (1) take advantage of discerning players by effectively getting more preference judgments per click and (2) make the game more entertaining.

Players have the option to choose “no good image” to indicate that none of the images are a good result for the query. If either player chooses this option, the number of images  $k$  is incremented or decremented as usual depending on agreement, but no points are awarded for agreement. We discuss this decision further in Section 3.4. Players can also flag individual images as being “bad”. If two players flag the same image as “bad”, then they are awarded a time bonus of five seconds if they also agree on the best image. Flag matches are not rewarded if the players agree on “no good image”.

In Figure 1 we show what the game interface looks like when a player is choosing an image. Note that the interface indicates in the lower right corner that the partner has already chosen. After the player makes his selection, his partner’s choice will be revealed.

After two minutes has passed or the pair of players has attained 100 points, the game ends. For players signed into the game site, these points are converted into a currency that can be spent on various items including t-shirts, computer hardware, computer software, and music. Players then have three options: they can choose to play again with the same partner, they can choose to play again with a new partner, or they can quit the game. If one player requests to play again with the same partner, that partner is given a choice to either accept or reject the invitation.

### 3.2 Query and Image Selection

The queries shown for each round in a game of *Picture This* are chosen randomly from a specific list. This list is updated over time, but the data that we analyze in this paper results from game play when the list contained a specific set of 427 queries that were both of interest to the image-search team and were appropriate for a public game (*e.g.*, they were not likely to return offensive images).

The candidate images for the query in each round are the top 50 results extracted from Live Image Search. One could, of course, use different values of  $n$  as the number of top images from which to select. We were partially guided by the differences in image search from web search. In a variety of commercial search engines, the default is to display 21-35 images as the first page of results, organized into a matrix-like format rather than a single list. Thus, we chose an  $n$  large enough to find relevant images which should be in the first page but are not currently, as well as keeping  $n$  small enough to easily get ranking feedback as to the desired full ordering of the images.

With each image, we store the fraction of rounds in which that image was chosen best, and we use this fraction as a simple estimate of image quality; we discuss this model further in Section 4.2.1. On each round, we either choose the  $k$  images uniformly at random from the entire set of 50 results, or we choose a random “base” image and select uniformly at random from the  $2k$  images that have the smallest difference in quality estimate from the base image. Rounds in which images are chosen to be close to a base image are typically both more difficult for the user and more informative to the system. We choose between the “hard” method and the “easy” method randomly, where the “easy” method is currently applied in roughly a third of the rounds. We did this to balance making fine-grained distinctions between images that are close to the same level of relevance with learning the overall ordering. Note that this is a simple form of active learning and depending on what the goals of the system are, a different active learning mechanism can be employed.

We emphasize that both the query and image pool can be constructed according to a variety of choices. For example, if no base search engine was available, one might choose to display a large number of images by default to quickly find some relevant images, and then compare winners of those rounds to get a refined ranking. Likewise, if some images are not currently in the top set, one can inject them into the result set to find out if they are in fact highly relevant. One might choose to adopt this strategy if the game is being

used to explore new features for ranking and the injected images could be images that would be highly ranked if that feature is given increased weight. In addition to our data-collection needs, the choice of the image-selection algorithm was informed both by architectural constraints in the system and by the enjoyment of game play.

### 3.3 Partner Robots

For players who could not be matched with a human partner within a small amount of time after starting the game, we implemented partner robots. These robots choose images based on preference estimates in the data, essentially using the method of Section 4.2.1. Unlike the ESP game [19], human behavior is not easily mimicked by simply replaying the choices (and timing) from previous game transcripts because the number of choices that are available for a query will vary from game to game. For example, the best image chosen by a player when two images are shown may be an obviously bad choice when there are additional more relevant images.

### 3.4 Incentive Structure and Fraud Mitigation

Because game points have real-world value, there is incentive for game players to cheat. In fact, there are entire discussion boards dedicated to the topic of cheating games on the Microsoft Live Search Club site. In this section, we describe a number of design decisions we made in order to mitigate against this threat.

Because points are awarded based on agreement, most cheating methods require coordination between two players. Our first line of defense against these methods is the random assignment of partners, although a fraudster can attack the game with several bots that start at the same time in order to increase the probability of getting a coordinated pair of players.

As described above, the two players are shown the candidate images in different orders and therefore strategies such as always choosing the first image will be no better than random selection. Because we do not award points for agreement on the “no good image” selection, the strategy of always making this choice does not pay off.

We were particularly concerned about defending against a random-selection strategy. Recall from Section 3.1 that the number of points earned for agreement is equal to the number of images shown; this arithmetic progression of the rewards helps to some degree because—relative to normal play—the expected value per round is low for random play. But if the system is attacked by two random-voting (and quickly voting) bots that are paired together, the total payoff could be significant. We have three major lines of defense against random-voting bots. First, all players are occasionally challenged with a puzzle (a CAPTCHA) that is difficult for an automated algorithm to pass. Second, the game site monitors all user-interaction events of the games and applies a bot-detection algorithm to this data. Finally, we have implemented a number of “honeypot” queries in the game for which a human has verified that there is an obvious best image; if a player chooses the wrong image for these queries, he is flagged as a potential cheater.

## 4. PREFERENCE MODELS

### 4.1 Data Representation

As described in Section 3.1, players can choose “no good image” instead of a best image, and they can also flag individual images as being bad. We model these choices in our preference data by including a virtual “neutral” image for every query in the data. In particular, whenever a player selects “no good image”, this is recorded as a preference for the neutral image over all the candidate images. Similarly, whenever a player flags an image as bad, we record this as a preference for the neutral image to that bad image. Finally, every time a player chooses an image instead of choosing “no good image” or flagging that image as bad, this is interpreted as a preference for the chosen image over the neutral image. Thus, the neutral image is implicitly a part of every comparison set. From a modeling perspective, this has the advantage of making the neutral image serve as an anchor and ensures that every image that is chosen at least once or part of a comparison when “no good image” is chosen is connected in a preference graph.

Now we abstract the representation of a collection of data to a more general learning setting. For a given query, we assume there is a set of items  $\mathcal{I}$  with  $I = |\mathcal{I}|$  elements; in our domain, these are all images displayed during the use of the query in game play plus the virtual “neutral” image described above. We will typically refer to the  $i$ th element of  $\mathcal{I}$  as item  $i$  or image  $i$ . We are given a set of labeled data pertaining to the query  $\mathcal{D} = \{o_1 = \langle r_1, \langle c_{1,1}, \dots, c_{1,\mathbb{S}(1)} \rangle \rangle, \dots, o_N = \langle r_N, \langle c_{N,1}, \dots, c_{N,\mathbb{S}(N)} \rangle \rangle\}$  consisting of  $N = |\mathcal{D}|$  observation records,  $o_k$ , where  $\mathbb{S}(k) = \text{length}(\langle c_{k,1}, \dots, c_{k,\mathbb{S}(k)} \rangle)$ ,  $r_k, c_{k,l} \in \{1, \dots, I\}$ , and  $r_k \in \{c_{k,1}, \dots, c_{k,\mathbb{S}(k)}\}$ . Each observation corresponds to the action of one player in a round. The  $c_{k,l}$  are the items being compared in the  $k$ th observation while  $r_k$  is the item selected from those compared in the  $k$ th observation. For shorthand, we will refer to the tuple of comparison items in an observation as  $C_k = \langle c_{k,1}, \dots, c_{k,\mathbb{S}(k)} \rangle$ .

### 4.2 Methods

In this section, we describe three different methods for modeling the preference data. We chose these models because of their progression from the most naïve probabilistic approach one might take with the grossest independence assumptions to increasingly tailored approaches with fewer independence assumptions. In particular, the frequency model simply models the global win probability of the image when it is displayed and does not account for any interaction between items. The pairwise probability model accounts for pairwise interactions, but not interactions among the entire comparison group. Finally, the Go model accounts for interactions among the comparison set by conditioning on the current example when learning and predicting. In all cases, the final set of model parameters are linear in the number of images. We restrict our attention to probabilistic models because the probabilities are useful for future improvement of our active-learning and image-selection components.

#### 4.2.1 Frequency Model

Our simplest model for the relevance score  $s_i$  for each image  $i$  is to use the fraction of times image  $i$  wins out of the number of times it appears. We refer to this as the *frequency* model. In practice, smoothing is necessary and we use a Bayesian m-estimate with a uniform prior over chosen or not chosen and two virtual observations. This simply

reduces to:

$$P(i \text{ chosen} \mid i \text{ shown}) = \frac{|\{C_k \mid i \in C_k, i = r_k\}| + 1}{|\{C_k \mid i \in C_k\}| + 2} \quad (1)$$

To use this model to predict the probability that  $i$  wins for some particular comparison set, you can compute:<sup>1</sup>

$$P(i = r_k \mid C_k) = \frac{P(i \text{ chosen} \mid i \text{ shown})}{\sum_{j \in C_k} P(j \text{ chosen} \mid j \text{ shown})} \quad (2)$$

This results in a model where the predicted winner depends on how high above the competition the relevant item is predicted to be. Given a large enough number of random comparisons, we expect this model to perform well because, in the long run, any particular image will have been compared against all other images a large number of times. Given non-random samples (*e.g.*, always comparing against very relevant images would yield a disproportionate number of losses) or a small number of samples relative to the  $n^2$  pairs, the model does not have the structure to distribute relevance scores in a transitive manner. Thus, it may not prove the most ideal choice when the goal is to quickly learn the ranking. Without empirical study, it is unclear what number of comparisons is needed for it to perform well and whether more complicated models are justified.

#### 4.2.2 A Pairwise Probability Model

Because we are measuring responses by humans, it is natural to look to psychometric theory for insights into this area. Thurstone [16, 17, 18] formulated a general law of comparative judgment that hypothesizes a relationship of the human ability (which has been repeatedly born out in empirical observations) to make pairwise comparisons between items based on a number of factors such as the actual difference between the quantities of the two items being judged (*e.g.*, actual difference in weight when trying to determine which item is heavier), the variance of the distribution, and the magnitude of each quantity involved. Under certain assumptions, and using the logistic function as the underlying distribution, this can be used to derive the Bradley-Terry-Luce (BTL) pairwise comparison model [1, 11].

In the BTL model, assuming a set of location parameters  $s_i$  (in our case, “relevance score” parameters), the probability of selecting one item as being greater (“more relevant”) than another is modeled as:

$$P(i \succ j) = \frac{\exp\{s_i - s_j\}}{1 + \exp\{s_i - s_j\}} \quad (3)$$

To interpret the data as pairwise preferences, we assume an observation  $\langle r_k, \langle c_{k,1}, \dots, c_{k,\mathbb{S}(k)} \rangle \rangle$  amounts to  $\mathbb{S}(k) - 1$  preferences expressing the response item is preferred to each remaining item,  $\{r_k \succ j \mid j \in C_k, r_k \neq j\}$ . For a pairwise probability model, modeling item  $i$  is preferred to item  $j$  (denoted  $i \succ j$ ), we assume no mass is reserved for a tie and that  $P(i \succ j) = 1 - P(j \succ i)$ . Given a pairwise probability model,  $P(i \succ j)$ , we assume an observation’s probability is the product of all expressed pairwise preferences. That is, the  $P(o_k) = \prod_{j \in C_k, r_k \neq j} P(r_k \succ j)$ . To set the parameters

<sup>1</sup>The other obvious use of these terms such as  $P(i = r_k \mid C_k) = P(i \text{ chosen} \mid i \text{ shown}) \prod_{j \in C_k, j \neq i} [1 - P(j \text{ chosen} \mid j \text{ shown})]$  would yield a deficient probability model in that it allocates non-zero probability to zero probability events because exactly one item must be chosen; we could consider this model where equality is replaced with proportionality.

by maximum likelihood, we must maximize the following function (see Appendix A):

$$\sum_{i=1}^I \sum_{j=i+1}^I [B_{i,j}^+(s_i - s_j) - n_{i,j} \log[1 + \exp\{s_i - s_j\}]] \quad (4)$$

where  $n_{i,j}$  is the number of times  $i$  and  $j$  were compared and  $i$  or  $j$  won and  $B_{i,j}^+$  is the number of times  $i$  and  $j$  were compared and  $i$  won.

Equation 3 is simply a linear model with a “1” on the winner and a “-1” on the loser, and it turns out that we can use a standard logistic-regression algorithm to optimize the  $s_i$  values given training data. In particular, it can be shown (see Appendix A for a full derivation) that the model is identical to a two-class ( $c \in \{0, 1\}$ ) logistic regression model that does not have a bias term.<sup>2</sup> To represent the pairwise model as a logistic regression problem, each pair of preferences derived from a comparison set is encoded as a feature vector with the same number of features as images. All feature values are 0 except for the two features corresponding to the preference being expressed. The feature with the lower *index* is set to 1 and the higher to  $-1$ . Finally, if the image corresponding to the feature with the lower index was preferred, the class is set to 1 and otherwise to 0.

Because of the easy availability of tested code implementing logistic regression, we chose to implement the optimization in this way. In both the original pairwise formulation and in the reduction to logistic regression, it may be advantageous to introduce a regularizer or prior to deal with sparse data. We use a L2 regularizer (Gaussian prior) which will pull the relevance scores of the images to zero unless the data dictate otherwise. The weight ( $\lambda$ ) of the regularizer is 1. Early experiments in which we investigated optimizing the regularizer weight over validation data indicated the results were not particularly sensitive to changes in the regularizer weight.

Note that Equation 3 can be rewritten as:

$$P(i \succ C_k) = \frac{\exp(s_i)}{\sum_{j \in C_k} \exp(s_j)} \quad (5)$$

where  $i \in C_k$  for those comparison sets containing two images. We can also use this equation directly to predict for comparison sets of *more* than two images. Assuming the resulting prediction model when training instead of Equation 3 would require the use of a conditional model (conditioning on the current comparison set). Since Section 4.2.3 presents an alternative conditional model, we present this to gain an understanding of whether the conditional model is needed to gain the majority of the predictive power.

#### 4.2.3 Go Model

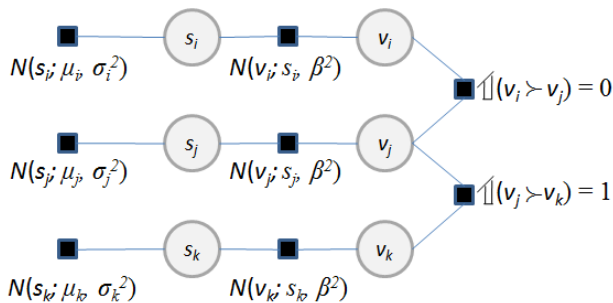
The final model was first applied to learning to predict moves in the board game Go [15] and is related in some aspects to the TrueSkill<sup>TM</sup> model used to rank and match players in online gaming [8]. The Go model is a Bayesian system with conditional online updates that can be understood as a general preference learning algorithm which learns to predict when the winning item (originally which Go move is selected) is preferred from a set of items (all Go moves available given the current board).

<sup>2</sup>A non-zero bias yields probabilities such that  $P(i \succ j) \neq 1 - P(j \succ i)$ .

There are several interesting properties of the Go model from a preference modeling point of view. First, the Go model does not “tie” the parameters of non-winning items. In contrast, many preference models implicitly or explicitly treat each preference vote as evidence that the losing items have the same relevance. The Go model is a *conditional* model; the model performs updates for the relevance of an image conditional on the current set of images being compared. In our progression of independence assumptions, the Go model is the most general because it can model group effects when comparing and predicting for a group of images. Finally, although we do not take advantage of this property in this work, the model has parameters that can be used to reflect different players’ knowledge of a query or skill in making relevance judgments.

Adapted to our application, the Go model works as follows. A Gaussian belief  $N(s_i; \mu_i, \sigma_i^2)$  is maintained about the relevance score  $s_i$  for each image  $i$ . Within each round of play and for each player, each image is assumed to have an unobserved relevance value  $v_i$  (dependent on player interpretation of query, player skill at making relevance judgments, knowledge of query *etc.*) which is Gaussian distributed around the relevance score with fixed variance,  $N(v_i; s_i, \beta^2)$ . Here  $\beta^2$  is a system input parameter. The observed variable of which image was chosen by the player is assumed by the model to be computed by choosing the images with the maximum latent relevance value  $v_i$ .

The factor graph for an example with three images ( $i, j, k$ )—where  $j$  is chosen as the winner—is shown in Figure 2. For each observed preference, a factor graph is built and approximate inference is performed to update the Gaussian belief distribution over relevance scores. This updated belief is then used as the prior belief distribution in the next update. After all training data is processed, the final system parameters are the belief distribution parameters  $\mu_i, \sigma_i^2$  and can be used to perform prediction inference. For use in our experiments, we use the default initial prior of a standard Gaussian and  $\beta^2 = 0.25$ . The update and inference algorithms are quite efficient and have already been shown to be applicable in a large-scale online system [8]. More details of the inference are available in [15].



**Figure 2:** A factor graph representing how the comparison of three items ( $i, j, k$ ) where item  $j$  was chosen as the winner depends on the factors (black boxes) in the model. The values of the indicator functions are represented as variables whose observed values are used to update the model parameters by probabilistic inference over the unobserved variables (gray circles).

For the predictions that concern us, inference is quite simple. In order to rank the images, one need only sort the images by the means of their relevance belief distribution,  $\mu_i$ . The pairwise probability that  $i \succ j$  is

$$P(i \succ j) = \Phi \left( \frac{\mu_i - \mu_j}{\sqrt{\sigma_i^2 + \sigma_j^2 + \beta^2}} \right) \quad (6)$$

where  $\Phi$  is the cumulative density of a zero-mean unit-variance Gaussian. To perform inference for comparison sets of more than two items, expectation propagation can be performed. A second method of inference that works for all comparison set sizes and may be more convenient at times is to use Monte Carlo simulation and simply sample  $n$  times from the Gaussian belief distributions for those items being compared, followed by sampling from the latent variable distribution, and, finally, count the proportion of times the corresponding image had the highest score.

## 5. EMPIRICAL ANALYSIS

We remind the reader that our primary objective is to identify a consensus ranking for a query. Thus, while we could consider alternative models, we focus on the simple question of whether a ranking extracted from the preferences of one set of users can predict the preferences of a second (disjoint) set of users. Thus any model’s success validates our primary concern. As secondary questions, we consider other properties of the consensus ranking, *e.g.* which model best predicts preferences, which model is better given limited data, *etc.*

In the deployment of the game, there are two primary servers that have independent state controlling the active learning algorithm that handles image and query selection. For any player with a user ID (*i.e.*, those playing non-anonymously), the player is assigned permanently to one server and only plays on that server. Because credits can only be collected for non-anonymous play, this constitutes the vast majority of use. Splitting users by server allows us to create a training and a test set that are essentially independent of any single user’s action. In addition, no state is shared between the servers. Thus, although an active learning component is used for image selection, the model that is updated on the test server is independent of that on the training server except for those effects dependent on aggregate assessment of relevance. Because each particular player’s actions are wholly in the test set or the training set and no state is shared, good performance indicates we are learning a general consensus view of relevance for the images and not just some particular player’s view or the view of those on the training server.

In this study, our goal is to predict—using the data from the training server—the choice behavior on the test server restricted to those choices in which the two players agree. We restrict the test set to agreement-only cases because we assume that agreement signifies a trustworthy signal about the relevance of the images. In future studies, we intend to identify trustworthy players explicitly and include all of their data.

### 5.1 Data

For each round of the game, we record in a database the query, the candidate images, the selections for each player,

and the timing information for each player. We also store a unique identifier for each round so that we can check for agreement.

To prepare the data for experimentation, we first removed all robot preference judgments, resulting in the *raw* versions of the training set and the test set. Next, we created *agreement* versions of both the training set and the test set that only included preference judgments corresponding to partners who agreed. For these agreement versions of the data sets, we retained a single preference representing each pair of judgments that agreed, and we removed all other judgments. Note that agreements with robots were not included in the agreement data due to the robot judgments being removed initially.

To test our models, we used only the agreement version of the test data. For training, we experimented both with the agreement version and the raw version of the training set. We use the raw version of the training set to test whether we can boost performance on the high-quality judgments by including noisy judgments.

As mentioned in Section 3.2, there are 427 queries for which we collected preferences over 34 days.<sup>3</sup> There are an average of 95 images per query (94 + virtual “neutral image”).<sup>4</sup> Tables 1 and 2 summarize basic data characteristics.

There was a 49% agreement rate in the training set<sup>5</sup> among the human-human rounds. Because on average 3.53 actions were available per round, random clicking would be expected to achieve a 28% agreement rate. Likewise, there was 50% agreement in the human-human test set rounds with random clicking expected to achieve a 28% agreement rate. Thus, humans agree far more often than can be expected from random clicking.

	Games	Rounds	Human-Human Rounds
Train	154,060	1,491,206	1,144,409
Test	155,322	1,522,375	1,159,570

**Table 1: Data properties before preprocessing. Counted rounds have at least one response.**

	Preferences	Pairwise Pref.	No Good Image
Raw Train	2,599,531	8,860,418	25,002
Raw Test	2,645,984	9,267,890	21,682
Agree Train	2 × 537,651	2 × 1,731,317	2 × 1,800
Agree Test	2 × 555,202	2 × 1,838,987	2 × 1447

**Table 2: Properties after removing robot preferences (*raw*) and retaining only human-human agreements.**

## 5.2 Performance Measure

The performance measure we examine here is a measure of each method’s ability to correctly predict the preference choice that was made during a game. This performance measure has the advantage that if multiple images are considered good matches for the query, the measure will be implicitly

<sup>3</sup>A version of the data suitable for preference learning is available at <http://go.microsoft.com/?linkid=9648573> for use in research experiments.

<sup>4</sup>Although the game selects from the top 50 images, the results returned by the search engine change over time.

<sup>5</sup>Note only a single preference is retained per agreement and human-robot pairs were discarded.

weighted by the proportion of the population’s preferences for one image over the other.

We present error in predicting preference choice relative to our baseline, Live Image Search. Relative error is computed as:

$$\frac{\text{error}_{\text{method}}}{\text{error}_{\text{baseline}}} \quad (7)$$

If the baseline is outperformed then relative error is between 0 and 1 with closer to 0 signifying increasing performance over the baseline. For example, a relative error of 0.33 would indicate that the method commits three times fewer prediction errors than the baseline.

To compute the baseline, we use the ranking of the images after game data collection (thus the baseline is optimistic) and predict the item ranked highest in the search results will be chosen. If images have dropped out of the index, we assume any other image in the index is preferred to them.

Note that we do not compute other search metrics like DCG because those rely on having a gold-standard ranking or graded relevance judgments to begin with. Furthermore, the preference error we give here is the natural extension of Kendall’s tau (percentage of pairs correctly ordered over total pairs considered) which is appropriate for measuring the distance between two rankings for rank aggregation [6].

## 5.3 Results

Table 3 presents the performance of models trained on the agreement training set on the left and the results of models trained on the raw training set on the right. The best result in each column is in bold. Again, we emphasize that in all cases the testing set is the agreement test set.

Because the raw training set yielded the best results for all methods, we sought to understand how much data was necessary to achieve good performance. We broke the raw training set into 5 chunks. Each chunk, numbered 1-5, is a superset of the previous chunk’s data with approximately 7 days more worth of data. Although the amount of data added is fairly uniform across each chunk, it can vary somewhat due to variance in the number of players playing and games played in a day. Therefore, we summarize the actual number of preferences and effective pairwise preferences for each chunk in Table 4. The performance for each model is presented in a learning curve in Figure 3.

## 5.4 Discussion

At first glance, it may seem surprising that the Frequency model shows such good performance when using the agreement training set, but as we discussed in Section 4.2.1, given a large enough number of random comparisons, there is reason to expect it to converge to a good model. In all cases, all of the models trained on the agreement training set commit twice fewer errors than the baseline search method with the Pairwise and Go models committing nearly three times fewer errors. Thus, all methods answer our primary question of can a consensus ranking from one set of users be used to model another. Furthermore, they all improve markedly over the baseline provided by Live Search.

Next, when considering whether all of the data can be used, by leveraging potentially noisier judgments in the raw training set, we see that all models improve using the unfiltered set. Relative to their performance trained on the agreement training set, we see the Pairwise and Go models improve less – perhaps because all models are beginning to

asymptote with the Pairwise and Go model having less room to improve further. Since all models improve using the unfiltered training set, this gives very promising evidence that, even when there is disagreement, the preferences contain useful information and offer possibilities for data mining.

Moving on to the learning curve, we gain a variety of interesting insights. First, even with 20% of the data, all methods perform relatively well compared to the baseline with the Pairwise and Go models already committing 2.5 times fewer errors than the baseline.

	Agreement Training	Raw Training
Frequency	0.3713	0.3504
Pairwise	0.3451	0.3335
Go	<b>0.3408</b>	<b>0.3325</b>

**Table 3: For each method, error relative to the baseline of using the search engine’s ranking to predict user preference on the agreement test set. Using the potentially noisy data in the raw set gains in all cases although more for the Frequency model which has more room to improve. The best result in a column is in bold.**

Next, we see the frequency model increasingly underperforms the other two models with less and less training data. This likely results from lacking the model structure to implicitly infer relationships like transitivity that are helpful when fewer pairs have been compared to each other. For this reason, we would expect this gap to be even larger with less data. Additionally, because an active learning component can exacerbate sparsity (by never comparing items believed to already be far apart), it may have undesirable interactions with the particular active learning method used. As is well-known in search, variance in both evaluation and training is largely due to queries. This is typically handled by building labeled sets over as large of number of queries as possible. Therefore a natural goal when using this approach to build a labeled data set is to try to get the minimal amount of information per query and cover as many queries as possible; thus, the frequency model is unlikely to be useful in practice since it requires more judgments per query.

Additionally, the Pairwise and Go model seem to track each other well. Thus, in terms of prediction accuracy, the more general Go model does not seem to offer significant benefits over the Pairwise model. In terms of probability prediction, however, it may prove more useful. In addition, its structure allows for interesting future directions discussed below. Thus, we believe both the Pairwise and Go model are good candidates for representing preferences in this domain.

Finally, because the Pairwise model is a simple model that performs well, we examined how well the rankings it learned correspond to “expert” editorial judgments. For this, we use an existing set of 5-point scale editorial judgments that were previously collected to assess performance over 60 of the 427 queries. Because the editorial judgments can often generate ties, we use the standard approach using Kendall’s tau of only considering those pairs for which we had judgments and for which the editorial judgments assigned different relevance levels to the two images (27299 pairs). Across these, the Pairwise model had an overall Kendall’s tau of 0.6742 with a median by query of 0.6881. Thus, the quality of the aggregated consensus game data appears to be quite in line

with that of editorial judges. Additionally, we benefit because the rankings learned in this manner have a much finer level of granularity. Finally, in the case of disagreements between the two, in the authors’ opinion the model learned by game data often seems superior (mistakes often appear to be because of lack of editor knowledge) – this is early anecdotal analysis, however, and we intend to conduct a much more formal analysis of the nature of disagreements.

## 6. FUTURE WORK

There are a variety of interesting avenues for future work related to this paper. One of the most obvious is to continue to apply the social labeling game framework to other search scenarios. For example, we can try to construct a game to be used to improve document ranking for web search. One challenge the designer faces is that players have an incentive for quick decisions in any situation where time is valuable. For image search, we feel that issue has less dissonance with the underlying task because most image-search tasks do not require detailed analysis to come to a conclusion of relative relevance; a document, on the other hand, may take more careful examination to come to such a conclusion. The quick-decision incentive may be advantageous in some situations (*e.g.*, exploring snippet summarization), but its impact requires careful consideration for each task.

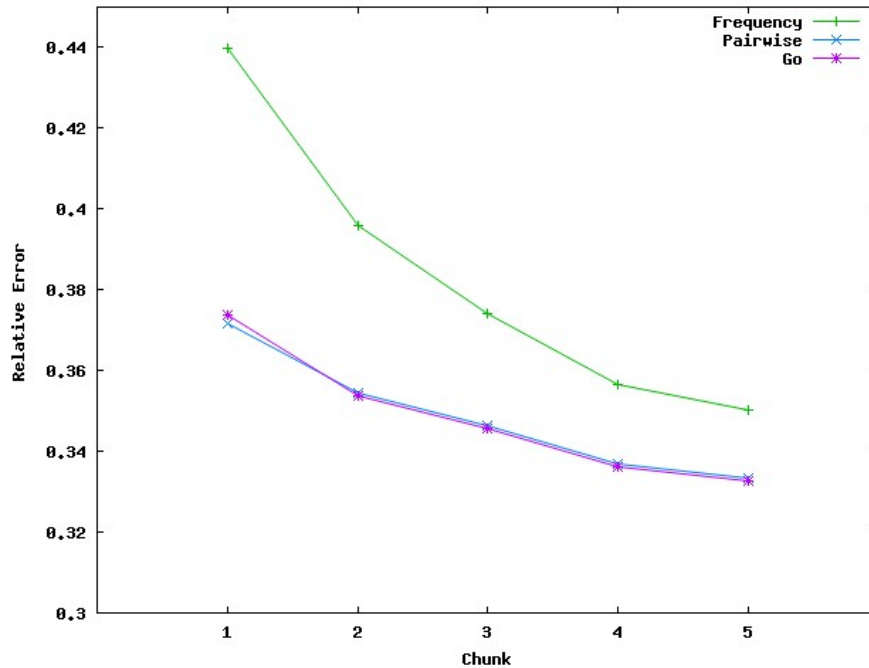
In terms of the models presented here, developing methods to analyze player performance is quite promising. The most simple would be to simply identify how accurate a player is and integrate that into the models. For example, the Go model supports setting  $\beta$  smaller for players who are likely to make consistently accurate relevance decisions. This parameter could also be used to allow the player to self-assess knowledge of query in some way before seeing the images (*e.g.*, betting more or less points) and then setting  $\beta$  as a decreasing function of the player’s self-assessed knowledge. Another interesting challenge is to address the design of the active learning component that chooses which images to compare. The system currently uses the basic heuristic described above, but there has been a variety of active learning work which may be leveraged in this scenario to estimate utility in a more principled fashion. In addition, work on preference-label ranking could be used to construct personalized models that predict the rankings of images given characteristics of the user; active-learning techniques would be relevant here in order to identify which image sets to label in order to quickly converge to a reasonable model.

## 7. CONCLUSION

In this work, we demonstrated how queries taken from search logs can be used to drive a fast-paced and enjoyable social labeling game. We described how the data resulting from game play is similar to click data but without issues of position bias, and is suitable for learning consensus rankings. We examined a number of preference learning models and demonstrated large improvements over a commercial search engine’s ranking. Finally, in an analysis that has implications for other preference learning problems, we were able to demonstrate that two of the models (Pairwise and Go) are suitable for problems with a range of training data sizes while the third (Frequency) is only suitable when training data is abundant.

Chunk	Number Pref.	Pref/Queries	Percent All Pref.	Pairwise Pref.	Pairwise/Query	Percent All Pairwise
Chunk1	657705	1540.29	0.25	2281342	5342.72	0.26
Chunk2	1152449	2698.94	0.44	4037916	9456.48	0.46
Chunk3	1529197	3581.26	0.59	5369399	12574.70	0.61
Chunk4	2201148	5154.91	0.85	7586388	17766.72	0.86
Chunk5	2599531	6087.89	1.00	8860418	20750.39	1.00

**Table 4:** Description of training chunks used for learning curve. Each chunk is a superset of the previous chunk plus approximately seven days of data. Since preference judgments where more than two images were compared can be counted in terms of effective number of pairwise judgments, we present number of pairwise preferences as well.



**Figure 3:** A learning curve demonstrating how quickly error decreases relative to the baseline for each method with increasing amounts of training data. The frequency model takes a much larger amount of training data to reach comparable levels of accuracy. Both the Pairwise and Go model cut the error rate of the baseline by more than a factor of  $2.5\times$  with only 20% (7 days) of data.

## Acknowledgments

We would like to thank Aparna Lakshmiratan, whose work designing and implementing the game was instrumental to the project’s success, as well as Ralf Herbrich and Thore Graepel for their assistance in applying the Go model.

## 8. REFERENCES

- [1] R. Bradley and M. Terry. Rank analysis of incomplete block designs, i. the method of paired comparisons. *Biometrika*, pages 324–345, 1952.
- [2] B. Carterette, P. N. Bennett, D. M. Chickering, and S. T. Dumais. Here or there: Preference judgments for relevance. In *ECIR 2008*, 2008.
- [3] W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *ICML 2005*, 2005.
- [4] C. Cleverdon. The significance of the cranfield tests on index languages. In *SIGIR '91*, pages 3–12, 1991.
- [5] W. Cohen, R. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.
- [6] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW '10*, pages 613–622, 2001.
- [7] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *ECML 2003*, 2003.
- [8] R. Herbrich and T. Graepel. Trueskill<sup>TM</sup>: A bayesian skill rating system. Technical Report MSR-TR-2006-80, Microsoft Research, June 2006. <ftp://ftp.research.microsoft.com/pub/tr/TR-2006-80.pdf>.
- [9] T. Joachims. Optimizing search engines using clickthrough data. In *KDD 2002*, 2002.
- [10] E. Law, L. von Ahn, R. Dannenberg, and M. Crawford. Tagatune: A game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval*, 2007.

- [11] R. Luce. *Individual Choice Behaviours: A Theoretical Analysis*. J. Wiley, New York, 1959.
- [12] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *AAAI 2005*, 2005.
- [13] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML 2008*, 2008.
- [14] S. E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33:294–304, 1977. Reprinted in: K. Sparck Jones and P. Willett (eds), *Readings in Information Retrieval*. Morgan Kaufmann, 1997. (pp 281-286).
- [15] D. Stern, R. Herbrich, and T. Graepel. Bayesian pattern ranking for move prediction in the game of go. In *ICML '06*, 2006.
- [16] L. Thurstone. A law of comparative judgment. *Psychological Review*, pages 273–286, 1927.
- [17] L. Thurstone. *Essays in Philosophy by Seventeen Doctors of Philosophy of the University of Chicago*, chapter The Measurement of Psychological Value. Open Court, Chicago, 1929.
- [18] L. Thurstone. *The Measurement of Values*. The University of Chicago Press, Chicago, 1959.
- [19] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *ACM CHI*, 2004.
- [20] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A game for locating objects in images. In *ACM CHI*, 2006.

## APPENDIX

### A. PAIRWISE MODEL DERIVATIONS

In this appendix, we derive results related to the pairwise model of Section 4.2.2. We assume terminology introduced throughout the paper and particularly that introduced in Sections 4.1 and 4.2.2. First, we demonstrate the derivation of Equation 4. If we desire to set the relevance score parameters by maximum likelihood, then we must maximize the following log-likelihood.

$$\begin{aligned} \Lambda(\mathcal{D}) &= \log \prod_{i=1}^N \prod_{j \in \{j | j \in \{1, \dots, \mathbb{S}(i)\}, r_i \neq c_{i,j}\}} P(r_i \succ c_{i,j}) \\ \text{Let } B_{i,j}^+ &\text{ be the number of } C_k \in \mathcal{D} \text{ s.t. } i, j \in C_k, r_k = i. \\ &= \log \prod_{i=1}^I \prod_{j=1 | j \neq i}^I P(i \succ j)^{B_{i,j}^+} \\ \text{Since } P(i \succ j) &= 1 - P(j \succ i) \\ &= \log \prod_{i=1}^I \prod_{j=i+1}^I P(i \succ j)^{B_{i,j}^+} [1 - P(i \succ j)]^{B_{j,i}^+} \\ &= \sum_{i=1}^I \sum_{j=i+1}^I [B_{i,j}^+ \log P(i \succ j) + B_{j,i}^+ \log [1 - P(i \succ j)]] \\ \text{Let } n_{i,j} &= |\{C_k \mid C_k \in \mathcal{D}, i, j \in C_k, (r_k = i \text{ or } r_k = j)\}| \\ \text{Note } B_{i,j}^+ + B_{j,i}^+ &= n_{i,j} = n_{j,i} \end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^I \sum_{j=i+1}^I [B_{i,j}^+ \log P(i \succ j) \\ &\quad + (n_{i,j} - B_{i,j}^+) \log [1 - P(i \succ j)]] \\ &= \sum_{i=1}^I \sum_{j=i+1}^I [n_{i,j} \log [1 - P(i \succ j)] \\ &\quad + B_{i,j}^+ \log \frac{P(i \succ j)}{1 - P(i \succ j)}] \\ &= \sum_{i=1}^I \sum_{j=i+1}^I [n_{i,j} \log \frac{1}{1 + \exp\{s_i - s_j\}} \\ &\quad + B_{i,j}^+ \log \exp\{s_i - s_j\}] \\ &= \sum_{i=1}^I \sum_{j=i+1}^I [B_{i,j}^+ (s_i - s_j) - n_{i,j} \log [1 + \exp\{s_i - s_j\}]] \quad (8) \end{aligned}$$

Next we demonstrate briefly that given the right encoding of a labeled example, the model is identical to a two-class ( $c \in \{0, 1\}$ ) logistic regression model that does not have a bias term. That is, a model where  $P(c = 1 \mid \vec{s}, \vec{x}) = \frac{\exp\{\vec{s} \cdot \vec{x}\}}{1 + \exp\{\vec{s} \cdot \vec{x}\}}$  and  $P(c = 0 \mid \vec{s}, \vec{x}) = \frac{1}{1 + \exp\{\vec{s} \cdot \vec{x}\}}$  where  $\vec{x}$  is the feature encoding for a pairwise preference described below.

In particular, the original comparisons should be encoded in the following way. To represent this as a logistic regression problem, each comparison set  $\mathbb{S}(i)$  gets converted to  $|\mathbb{S}(i)| - 1$  labeled examples for logistic regression. Similar to how a choice was modeled as the pairs of preferences needed to indicate the chosen item was preferred to each of the remaining items, each example in the logistic regression representation will encode one of these preferences. There is one feature in the model for each of the images. To encode a preference that  $i \succ j$ , all features are set to zero except for the  $i$ th and  $j$ th feature. Then an arbitrary choice is made to set one of these features to 1 and the other to  $-1$ . The class variable is set to “1” if the image whose feature bit was set to 1 was chosen, and is set to “0” otherwise. Without loss of generality, for convenience we will assume that the feature with the lowest feature index is always set to 1 and the one with the higher index is set to  $-1$ . Thus the class label will be “1” if the image corresponding to the lower feature index was chosen and “0” otherwise. Assume we have indices  $i, j$  where the index  $i < j$ , then to encode  $i \succ j$ , the  $i$ th bit is set to 1, the  $j$ th bit is set to  $-1$  and the class is 1. Now, we simply must show that maximizing this likelihood results in the same optimization formula as Equation 8.

$$\begin{aligned} &\sum_{d=1}^N \sum_{i=1}^I \mathbb{1}(i \in \mathbb{S}(d)) \sum_{j=i+1}^I \mathbb{1}(j \in \mathbb{S}(d)) [\mathbb{1}(i = r_d) P(1 \mid \vec{s}, \vec{x}) \\ &\quad + \mathbb{1}(j = r_d) P(0 \mid \vec{s}, \vec{x})] \\ &= \sum_{d=1}^N \sum_{i=1}^I \mathbb{1}(i \in \mathbb{S}(d)) \cdot \\ &\quad \sum_{j=i+1}^I \mathbb{1}(j \in \mathbb{S}(d)) [\mathbb{1}(i = r_d) (s_i - s_j) \\ &\quad - (\mathbb{1}(i = r_d) + \mathbb{1}(j = r_d)) \log (1 + \exp\{s_i - s_j\})] \end{aligned}$$

Grouping like terms

$$= \sum_{i=1}^I \sum_{j=i+1}^I [B_{i,j}^+ (s_i - s_j) - n_{i,j} \log [1 + \exp\{s_i - s_j\}]] \quad (9)$$