Towards an Intelligent Bidding Agent in QuiBids Penny Auctions

David Storch*
dstorch@cs.brown.edu
Brown University Department of Computer Science

May 2, 2013

Abstract

Empirical data shows that penny auctions on QuiBids are profitable for the auctioneer at the expense of the bidders. We seek to determine whether it is possible to design a bidding algorithm which outperforms actual QuiBids users, capable of consistently making a profit in QuiBids auctions over time. The ideal bidding strategy would win the item while minimizing the total number of bids placed over the course of the auction—that is, it should only bid when the auction is on the verge of ending. We use data scraped from QuiBids in order to train a hierarchical supervised learning algorithm capable of predicting the end of novel QuiBids auctions. The results show that we can predict the end of the auction in most cases within a resolution of a few hundred bids.

1 Introduction

A penny auction is a form of ascending auction in which each bidder must pay a fixed fee for each bid it places. The winner must also pay its bid in order to acquire the good up for auction, although the bid fees are often orders of magnitude larger than the good price. Furthermore, each bid causes the good price to increase by a small, fixed increment, usually one cent. Penny auctions are all-pay auctions—the bid fees must be paid by both winners and losers—and hence can be immensely profitable for the auctioneer (Krishna and Morgan, 1997 [6]). Profitability for the auctioneer is the consequence of penny auction bidders dramatically overbidding in aggregate (Augenblick, 2012 [1], Wang and Xu, 2012 [10], Hinnosaar, 2010 [5]).

The penny auction website QuiBids[1] advertises the possibility for outrageous savings on retail items. It is certainly the case that some auctions result in items being sold to QuiBids users far below their actual retail value. But is it really possible to *consistently* make a profit by participating in online penny auctions? Or is it more accurate to categorize penny auctions as a form of gambling, in which the auction house will inevitably come out ahead over time?

Our past work uses empirical data scraped from QuiBids to demonstrate that the auctioneer achieves profit margins on the order of 30%, at the expense of most bidders (Stix et al., 2013 [8]). The vast majority of bidders take at least a small loss, and very few end up profiting. The implication is that it is difficult to make a profit in QuiBids auctions with a naive strategy.

We investigate whether a machine learning-based bidding agent is capable of reliably turning a profit in QuiBids auctions. At the very least, QuiBids' advertising campaign is misleading in the sense that naive bidding behavior is unlikely to lead to large savings on retail items. We wish to prove, however, that a more sophisticated agent is in theory capable of attaining the advertised savings.

2 Defining Penny Auctions

2.1 Auction Model

At any point in time, a penny auction is characterized by three state variables: the current highest bid p, the current highest bidder w, and the time remaining on the auction clock t. The auction begins with some

*I would like to thank my advisor, Amy Greenwald, and my collaborator, Eric Sodomka, for many hours of helpful discussion. I also credit Eric Sodomka for his contributions to the section defining penny auctions. Thanks also to Jeff and Eric Stix for their help with the initial stages of this project. Finally, I would like to thank my parents for their support in all corners of life.

[1] www.quibids.com

initial bid price $p := \overline{p}$ and an initial clock time of $t := \overline{t}$. The high bidder is initially null ($w := \emptyset$). The clock time t begins decreasing, and while $t > 0$, any bidder b may place a bid at some fixed incremental amount δ above the current highest bid. In order to place its bid, however, bidder b must pay the auctioneer an immediate bid fee ϕ. The highest bidder is then set to $w := b$, and the new bid price becomes $p := p + \delta$. If the time remaining on the clock when the bid is placed is lower than the reset time t_{reset}, then the clock is set to $t := t_{reset}$. Equivalently, when a bid is placed, the clock is set to $t := \max(t, t_{reset})$ in order to ensure that other bidders have at least t_{reset} to place an additional bid. The auction ends when $t = 0$, at which point the current highest bidder w wins the item and pays the current bid price p (in addition to any bid fees paid along the way). Note that both the winning bidder and the losing bidders have to pay the bid fee for every bid they place.

2.2 QuiBids' Rules

QuiBids acts as the auctioneer for multiple penny auctions that happen both simultaneously and sequentially across the day. For each of its auctions, QuiBids follows with the above model where $\overline{p} = \$0$, $\phi = \$0.60$, and $\delta = \$0.01$. The reset time t_{reset} is a function with range {20, 15, 10} seconds whose output decreases as the auction progresses. The starting clock time $\overline{t}$ varies depending on the auction, but is on the order of hours.

QuiBids adds some variants to the standard penny auction, including *Buy-Now*, *voucher bids*, and the *BidOMatic*. We describe these these additional auction rules below for completeness, but will not analyze them in greater detail in this work (see (E. Stix et al., 2013 [8]) for a more thorough discussion). The *Buy-Now* feature allows any bidder who has lost an auction to buy a duplicate version of that good at a fixed price $\overline{m}$. If a bidder uses Buy-Now, any bid fees the bidder incurred in the auction are subtracted from $\overline{m}$. The Buy-Now price is usually marked up from the actual retail value of the good, with an average markup of approximately 21% (E. Stix et al., 2013 [8]).

Voucher bids are a special type of good auctioned in packs by QuiBids. When a bidder wins a pack of N voucher bids, it is able to place N subsequent bids in future auctions for a bid fee of \$0 instead of the usual fee ϕ. The Buy-Now price for a pack of N voucher bids is always $\overline{m} = \phi N$. Voucher bids are identical to standard bids with one exception: whereas each standard bid reduces the Buy-Now price by ϕ, voucher bids cannot be used to reduce the Buy-Now price. If a bidder has won a voucher bid pack, it can choose at any time whether to use these voucher bids or to use standard bids in subsequent auctions.

The *BidOMatic* tool allows a bidder to specify a number of bids between 3 and 25 that will be automatically submitted on the bidder's behalf, starting at a particular price p. The BidOMatic will submit each bid at a random time between the reset time t_{reset} and zero seconds. Whether or not a bid is placed with the BidOMatic is public information.

3 Data

The dataset used in this paper consists of the complete bid histories for over 6,000 QuiBids auctions.[2] The auctions took place between February 21st, 2013 and March 19th, 2013. For each bid in each of these auctions we record the following information:

1. Auction ID: a unique auction identifier assigned by QuiBids.
2. Buy-Now Price: QuiBids' listed price $\overline{m}$ for the good being auctioned.
3. Item Name: the name of the good being auctioned.
4. Highest Bid: the new highest bid price p of the good after this bid is placed.
5. Bidder: a unique identifier for the user who placed the bid.
6. Bid Type: a flag which is 1 if the bid was placed by the BidOMatic, or 0 if the bid was a single bid.
7. Recent Bidders: the number of distinct bidders who placed a bid in the previous five minutes.
8. Clock Time: the amount of time remaining on the auction clock when the bid was placed, in seconds.
9. Reset Time: the time t_{reset}, in seconds, to which the clock was reset following this bid (either 20, 15, or 10).
10. Date: the date on which the bid was placed.

[2]The dataset was collected with a scraper written in Ruby using the Watir framework for web application testing (`www.watir.com`).

11. Time: the time of day, measured to the second, at which this bid was placed.

Voucher bid auctions are excluded from the dataset. With the exception of voucher bid auctions, the complete auction histories constitute a representative sample of QuiBids auctions. There are roughly even numbers of recorded bids placed on items in each price category (\$0–\$50, \$50–\$100, and so on) and in each retail category (e.g. electronics, home and garden, gift cards). Only bids placed while $t \leq t_{reset}$ were recorded.

The raw data consists of a matrix R with the eleven columns described above. Each row in R corresponds to a particular bid. We can also think of the raw data as comprised of a set A of auctions. For an auction $a \in A$ we notate by R_a the matrix of raw data for that particular auction. The rows of each such R_a are sorted in ascending chronological order (i.e. the bids are sorted by the time at which they were placed). The final row of R_a therefore identifies the winner of the auction a, the bid price at which the good was sold, and the time at which the auction ended.

4 Designing An Agent

Our goal is to design an automated bidding agent for QuiBids penny auctions, relying on the data R scraped from QuiBids as training data. From the bidder's perspective, the ideal outcome of a penny auction is winning the auction with a single bid. Placing a bid is costly, so a rational bidder will refrain from bidding until the auction would otherwise end. This, of course, is a drastic simplification since a bidder cannot be sure about others' intentions and can never be certain whether an auction is about to end. Furthermore, the very act of bidding when the auction would otherwise end may influence the behavior of the other bidders, potentially causing the auction to continue.

The realization that the actions of an automated bidding agent alter the behavior of the other bidders poses a profound problem. The training data, needless to say, does not incorporate the effects of the automated agent's behavior. Consequently, the training data might no longer be predictive of the auction outcome as soon as the agent places a bid. Proper evaluation of an automated agent, therefore, cannot be done with training data alone, but rather would require active experiments with real human participants.

In order to sidestep this problem, we focus on designing a method which, by passive observation of an auction a', can accurately predict the number of bids $\hat{y}$ remaining in the auction. The agent should produce a revised estimate of the number of bids remaining after each successive bid is observed, so that it produces a vector of predictions $\hat{\mathbf{y}}_{a'}$. The prediction made after observing the first i bids is denoted $\hat{\mathbf{y}}_{a'}[i]$. For the purposes of evaluation, we presume that the bidding agent takes no action, and therefore has no impact on the course of the auction. However, the predictions $\hat{\mathbf{y}}_{a'}$ could in theory be basis for a profitable bidding strategy. We suggest three bidding strategies that can be implemented on top of end-of-auction prediction:

- Define a threshold ϵ. Do nothing while $\hat{\mathbf{y}}_{a'}[i] > \epsilon$. As soon as $\hat{\mathbf{y}}_{a'}[i] \leq \epsilon$, begin bidding aggressively with the BidOMatic. Stop bidding aggressively only if the agent's total expended bid fees equal the Buy-Now price $\overline{m}$ of the item, at which point Buy-Now should be invoked, or when the agent wins the auction. Aggressive bidding behavior at the end of an auction has been shown to be associated with auction winners (J. Stix, 2012 [9]). This finding suggests that aptly timed aggressive bidding behavior might be capable of winning, despite refraining from submitting bids for the majority of the auction.

- As before, define a threshold ϵ, but also define a clock time threshold t_{thresh}. Instead of bidding aggressively with the BidOMatic, place a single bid only when observing that $\hat{\mathbf{y}}_{a'}[i] \leq \epsilon$ and $t \leq t_{thresh}$. This is an extension of the "constant-time" strategy proposed by (J. Stix, 2012 [9]).

- Store the prediction $\hat{y}_{start} := \hat{\mathbf{y}}_{a'}[1]$ associated with the first bid of the auction. Whenever the auction clock drops below t_{thresh} seconds, bid with probability $\min(1, \max(0, \hat{\mathbf{y}}_{a'}[i]/\hat{y}_{start}))$.

4.1 A Hierarchical Method for Predicting the Auction End

We take a two-level hierarchical approach for predicting the auction end. Given a novel auction a', the first step is to compare a' to all training auctions $a \in A$. The aim of this step is to identify a set of auctions $S \subset A$ that are most similar to a'.

The second step is to use each of the similar auctions $s \in S$ in order to predict the number of bids

$\hat{\mathbf{y}}_{a'}$ remaining in auction a'. Consider making a prediction $\hat{\mathbf{y}}_{a'}[i]$ after a particular number of bids i has already elapsed. For each similar auction we obtain a prediction $\hat{y}_s$. The final prediction is the average of the individual predictions from the auctions in S:

$$\hat{\mathbf{y}}_{a'}[i] = \frac{1}{|S|} \sum_{s \in S} \hat{y}_s \qquad (1)$$

We develop the details of both steps of the two-level method in the following sections. See Figures 1 and 2 for a summary of the method.

4.2 Generating Feature Vectors

The first step in the pipeline involves preprocessing the raw bid histories in order to generate a training set of feature vectors. We need to generate two types of feature vectors: *per-auction* feature vectors and *bid history* feature vectors. Per-auction feature vectors are based on auction-level features such as the Buy-Now price of the good being auctioned and the retail category of the good. The per-auction feature vectors are used in order to determine the set S of similar auctions. In contrast, bid history feature vectors are computed using more granular bid-level information such as the time at which each bid was placed and the user that placed each bid. The bid history feature vectors are used to make the predictions $\hat{y}_s$. The process used to compute both per-auction feature vectors and bid history feature vectors is described below.

We wish to construct a single per-auction feature vector $\mathbf{x}_a$ for each $a \in A$. The per-auction feature vector should be a function of the very first bid recorded in the complete auction history. This means that after the bidding agent observes the first bid of a novel auction, it has enough information to map that auction into the per-auction feature space and compute the set S. Let the $R_{a,i,j}$ denote the submatrix of R_a which contains rows i through j of the full matrix (inclusive). The per-auction feature vector $\mathbf{x}_a$ for auction a is some function f of the opening bid:

$$\mathbf{x}_a = f(R_{a,1,1}) \qquad (2)$$

We make use of the remaining rows of R_a in order to compute the bid history feature vectors. For each auction in the raw data, the matrix R_a will be converted into the corresponding matrix of feature vectors X_a. Each row in X_a is a bid history feature vector. Let the ith bid history feature vector (i.e. the ith row of X_a) be denoted by $X_a[i]$.

The ith feature vector for auction a is a function of the first i bids in the recorded history. This design, again, has the bidding agent in mind—the ith feature vector can only depend on the bids that the agent will have seen so far. The remaining bids are in the future and are hence hidden. The ith feature vector can now be written as a function g of the submatrix comprised of the first i rows of raw data:

$$X_a[i] = g(R_{a,1,i}) \qquad (3)$$

Although f and g are left unspecified here, we define particular a f and a particular g in order to evaluate our method.

Each bid history feature vector $X_a[i]$ has an associated training label $\mathbf{y}_a[i]$. Equivalently, the training data X_a for auction a has an associated vector of labels $\mathbf{y}_a$. Suppose that the bid history for auction a contains n_a bids. The label corresponding to the ith feature vector is the number of succeeding bids:

$$\mathbf{y}_a[i] = n_a - i \qquad (4)$$

Each training example is thus paired with a ground truth giving the number of bids remaining in the auction. For evaluation, we compare our predictions $\hat{\mathbf{y}}_a$ against the ground truth $\mathbf{y}_a$.

4.3 Finding Similar Auctions

Our goal is, given a new auction a', to find a set $S \subset A$ of auctions which we will use to make a prediction of the number of bids remaining. In order to do so, we make use of the set of per-auction feature vectors: $\{\mathbf{x}_a : a \in A\}$. We determine S using k-nearest neighbor where k is some predetermined small integer. Namely, the problem is to find a subset of auctions S such that $|S| = k$ and for any $s \in S$:

$$||\mathbf{x}_s - \mathbf{x}_{a'}|| \leq ||\mathbf{x}_a - \mathbf{x}_{a'}|| \quad \forall a \in A \setminus S \qquad (5)$$

This S is not guaranteed to be unique; we choose a solution S uniformly at random among all possible solutions. We efficiently find a solution that satisfies Equation 5 by using the set of per-auction feature vectors $\{\mathbf{x}_a\}$ to construct a kd-tree, and then querying the kd-tree for the k vectors nearest to $\mathbf{x}_{a'}$ (see e.g. (Yianilos, 1993 [11]) and (Cormen et al., 2009 [4])). The metric is Euclidean distance.

4.4 Predicting Remaining Bids

Suppose that we are trying to compute a particular $\hat{y}_s$—we have identified an auction s which looks similar to the novel auction a' and, by comparison to s,

```
QuiBidsTrain($A^{train}$, $R$)
1  $E \leftarrow \emptyset$
2  $models \leftarrow \{\}$
3  for $a \in A^{train}$ do
4      $\mathbf{x}_a \leftarrow f(R_{a,1,1})$
5      $E \leftarrow E \cup \mathbf{x}_a$
6      $n_a \leftarrow \text{length}(R_a)$
7      $X_a \leftarrow$ new matrix with $n_a$ rows
8      $\mathbf{y}_a \leftarrow$ new vector of length $n_a$
9      for $i = 1, \ldots, n_a$ do
10         $X_a[i] \leftarrow g(R_{a,1,i})$
11         $\mathbf{y}_a[i] \leftarrow n_a - i$
12     $models[a] \leftarrow$ TrainModel($X_a, \mathbf{y}_a$)
13 output KDTreeConstruct($E$), $models$
```

Figure 1: Given the raw data for the set of training auctions A^{train}, the generic procedure shown here is used to train the hierarchical model. The procedure returns two data structures: a kd-tree constructed from the set of per-auction feature vectors, and an associative data structure that maps from an auction to the corresponding trained model. Depending on whether we are running *knn-knn*, *knn-lin*, or *knn-poi*, the TrainModel procedure might involve constructing a kd-tree, computing parameters for a linear regression model, or computing parameters for a Poisson regression model. For details on notation, see the glossary of symbols (Figure 3).

Symbol	**Description**
A^{train}	the set of auctions in the full auction bid histories
A^{test}	the set of auctions in the end auction data
a	generic particular auction
f	per-auction feature vector function
g	bid history feature vector function
n_a	the number of recorded bids in auction a
R	matrix of all raw data
R_a	matrix of raw data for auction a
$R_{a,i,j}$	submatrix of R_a consisting of rows i through j, inclusive
S	set of similar auctions found by the first step of the hierarchical method
X_a	matrix containing feature vectors
$X_a[i]$	ith feature vector for auction a
$\mathbf{x}_a$	per-auction feature vector for auction a
$\mathbf{y}_a$	vector of training labels for auction a
$\mathbf{y}_a[i]$	ith training label for auction a
$\hat{\mathbf{y}}_a$	vector of predicted bids remaining
z_a	actual number of remaining bids upon predicting the auction end
δ	bid increment
ϵ	bid threshold for computing z_a
ϕ	bid fee
θ	vector of model parameters

Figure 3: Glossary of symbols.

we wish to determine the number of bids remaining in a'. In order to accomplish this task, we harness the training data X_s and the associated labels $\mathbf{y}_s$. If the first i bids of the novel auction have elapsed, then the most up-to-date bid history feature vector is $X_{a'}[i]$. Our approach is to train a model with X_s and $\mathbf{y}_s$, and then to use $X_{a'}[i]$ as test data which yields the model prediction $\hat{y}_s$. We compare three models for solving this problem: 1) single-nearest neighbor, 2) linear regression, and 3) Poisson regression.

In the single-nearest neighbor case, we train a model for each $a \in A$ by building a kd-tree containing the feature vectors in X_a. In order to compute $\hat{y}_s$, we first lookup the kd-tree corresponding to auction s. Then we use this kd-tree to find the index j of the feature vector that is closest to $X_{a'}[i]$ according to a Euclidean distance metric. If the training data X_s consists of n_s bids:

$$j = \underset{1 \le v \le n_s}{\arg\min} ||X_s[v] - X_{a'}[i]|| \qquad (6)$$

Although evaluating Equation 6 by brute force would require $O(|S|)$ time, the time complexity of the kd-tree query is logarithmic in the length of the auction history. Once we determine index j, we set $\hat{y}_s := \mathbf{y}_s[j]$. After following this process for each s, we

```
$\textsc{QuiBidsTest}(A^{test}, R, k, kdtree, models)$
$Y \leftarrow \emptyset$
$\hat{Y} \leftarrow \emptyset$
for $a \in A^{test}$ do
    $\mathbf{x}_a \leftarrow f(R_{a,1,1})$
    $S \leftarrow \textsc{KDTreeQuery}(kdtree, k, \mathbf{x}_a)$
    $n_a \leftarrow \text{length}(R_a)$
    $X_a \leftarrow$ new matrix with $n_a$ rows
    $\mathbf{y}_a \leftarrow$ new vector of length $n_a$
    $\hat{\mathbf{y}}_a \leftarrow$ new vector of length $n_a$
    for $i = 1, \ldots, n_a$ do
        $X_a[i] \leftarrow g(R_{a,1,i})$
        $\mathbf{y}_a[i] \leftarrow n_a - i$
        $\hat{\mathbf{y}}_a[i] \leftarrow 0$
        for $s \in S$ do
            $\hat{y}_s \leftarrow \textsc{Predict}(models[s], X_a[i])$
            $\hat{\mathbf{y}}_a[i] \leftarrow \hat{\mathbf{y}}_a[i] + \hat{y}_s / k$
    $\hat{Y} \leftarrow \hat{Y} \cup \hat{\mathbf{y}}_a$
    $Y \leftarrow Y \cup \mathbf{y}_a$
output $Y, \hat{Y}$
```

Figure 2: The procedure which is used to make predictions for all test auctions. Given the kd-tree and trained models generated by the QuiBidsTrain procedure, it returns the set of all prediction vectors, $\{\hat{\mathbf{y}}_a : a \in A^{test}\}$, and the set of all training labels for the test data, $\{\mathbf{y}_a : a \in A^{test}\}$. Depending on whether we are running *knn-knn*, *knn-lin*, *knn-poi*, the Predict procedure could query a kd-tree for the single-nearest neighbor, use a trained linear regression model to make a prediction, or use a trained Poisson regression model to make a prediction. For details on notation, see the glossary of symbols (Figure 3).

compute the final estimate $\hat{\mathbf{y}}_{a'}[i]$ according to Equation 1.

A general problem with nearest neighbor techniques is that Euclidean distance loses its meaning in a high-dimensional space (Beyer et al., 1999 [2]). The single-nearest neighbor strategy thus limits us to using a small number of features (on the order of 10). This "curse of dimensionality" problem motivates using ordinary least squares linear regression as an alternative approach. Linear regression is also among the simplest of regression models, so it provides a baseline for comparison against other regression techniques.

Linear regression solves the "curse of dimensionality" problem, but suffers from a new problem: it yields negative predictions. The number of bids remaining is always a nonnegative integer. We therefore explore Poisson regression as a third technique for computing the estimates $\hat{y}_s$. The Poisson regression model is tailored for making predictions on count data. It is an instance of a generalized linear model with a logarithmic link function (Cameron and Trivedi, 1998 [3]). The model assumes that the data follows

$$P(\mathbf{y}_a[i] \mid X_a[i], \theta) = \text{Poisson}(e^{\theta^T X_a[i]}) \qquad (7)$$

where θ is a vector of the model parameters and Poisson(λ) is the probability mass function of the Poisson distribution with mean λ. Equivalently, the labels are Poisson-distributed with means $E[\mathbf{y}_a] = \exp(X_a\theta)$. Training the model requires maximum-likelihood parameter estimation—finding $\arg\max_\theta L(\theta|X_a, \mathbf{y}_a)$. There is no closed-form solution for the MLE, but good estimates can be obtained by standard gradient descent algorithms.

5 Results and Evaluation

We have described three hierarchical methods for end-of-auction prediction: k-nearest neighbor followed by single-nearest neighbor (*knn-knn*), k-nearest neighbor followed by linear regression (*knn-lin*), and k-nearest neighbor followed by Poisson regression (*knn-poi*). Our implementation uses the k-nearest neighbor and regression packages provided by Scikit-learn (Pedregosa et al., 2011 [7]) along with MATLAB's generalized linear model toolbox. Here we present and compare the results for these three methods.

The set of auctions A was split into a training set A^{train} and a test set A^{test} by randomly selecting auctions. Approximately 20% of the auctions are in A^{test}, with the remaining 80% in A^{train}. For each auction $a \in A^{test}$, we have an associated vector of labels $\mathbf{y}_a$. Our goal is to use the training set A^{train} to compute the corresponding vector of estimates $\hat{\mathbf{y}}_a$. The predictions $\hat{\mathbf{y}}_a$ are plotted against the true labels $\mathbf{y}_a$ for an example auction in Figure 4.

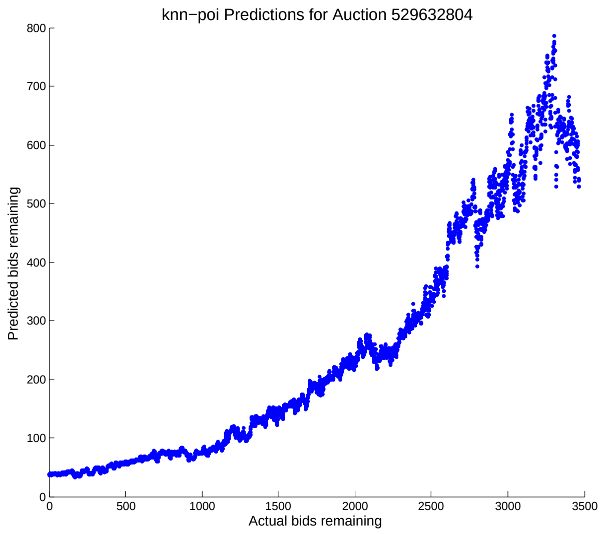

Figure 4: Predicted bids remaining $\hat{\mathbf{y}}_a$ versus actual bids remaining $\mathbf{y}_a$ for auction $a = 529632804$. The auction was for a Cuisinart FP-14DC Elite Food Processor with a Buy-Now price of \$283.99.

The results use $k = 20$ for k-nearest neighbor. The Buy-Now price is the sole per-auction feature, so that $f(R_{a,1,1})$ simply returns the Buy-Now price contained in the first row of the the raw data matrix. Furthermore, g is defined to compute the following bid-level features for each $X_a[i]$:

1. Time Elapsed: the time elapsed between the placement of the auction's opening bid and the placement of the ith bid.
2. Absolute Time: the absolute time of day at which the ith bid was placed, measured in seconds.
3. Highest Bid: the new highest bid price p of the good after this bid is placed.
4. Unique Bidders: the number of unique bidders that have bid so far in the auction.

5. Unique Bidders Last 50: the number of unique bidders in the last 50 bids.

6. Bids Placed: the total number of bids placed thus far by the current highest bidder.

7. BidOMatic Bids: the total number of BidOMatic bids placed thus far in the auction.

The first evaluation metric is the residual sum of squares (RSS):

$$RSS = \sum_{a \in A^{test}} ||\hat{\mathbf{y}}_a - \mathbf{y}_a||^2 \tag{8}$$

The smaller the RSS, the more closely $\hat{\mathbf{y}}_a$ predicts $\mathbf{y}_a$. The results show that *knn-lin* has the largest squared residuals; *knn-knn*'s RSS is 41.3% that of *knn-lin*, and *knn-poi*'s RSS is 5.9% that of *knn-lin*. By this measurement, *knn-poi* is the favored method.

The RSS indicates how well the method predicts the number of bids remaining at any point in the auction. But for the purposes of a bidding agent, it is not strictly necessary to always generate an accurate prediction of the number of remaining bids. What is more important is whether the predictions accurately indicate when the end of the auction is imminent. Figure 4, for example, shows large errors between the predicted and actual numbers of remaining bids—when there are 3,000 bids remaining, *knn-poi* suggests that there are only 500. This is nevertheless a "good" case in the sense that the graph lies close to the origin towards the end of the auction. When there are a small number of bids remaining, the prediction indeed recovers that few bid remain.

The preference for accuracy specifically when the end of the auction is impending requires an end-of-auction aware evaluation scheme. We use the following algorithm to produce an end-of-auction prediction for each $a \in A^{test}$: find the first bid in the test auction whose associated prediction $\hat{\mathbf{y}}_a[i]$ falls below some threshold ϵ. After computing the smallest j such that $\hat{\mathbf{y}}_a[j] \leq \epsilon$, the corresponding label $\mathbf{y}_a[j]$ gives the actual number of bids remaining at this point. Let $z_a := \mathbf{y}_a[j]$ so that z_a can be interpreted as the actual number of bids remaining when we predict the end of auction a. If $\hat{\mathbf{y}}_a[i] > \epsilon \ \forall i$ then the prediction is never sufficiently low. In this case, z_a does not exist and the method has failed to predict the end of a.

We use the end-of-auction predictions z_a for three evaluation metrics:

1. The *mean bids remaining*, or the mean z_a taken over the auctions $a \in A^{test}$. The auctions for which z_a does not exist are excluded from this calculation.

2. The *median bids remaining*, or the median z_a taken over $a \in A^{test}$. Together, the mean and median z_a indicate precision. The lower they are, the more precise the end-of-auction predictions.

3. The *auction recall*. This is the fraction of the auctions in A^{test} for which z_a exists.

These evaluation metrics are shown as a function of the threshold ϵ in Figure 5.

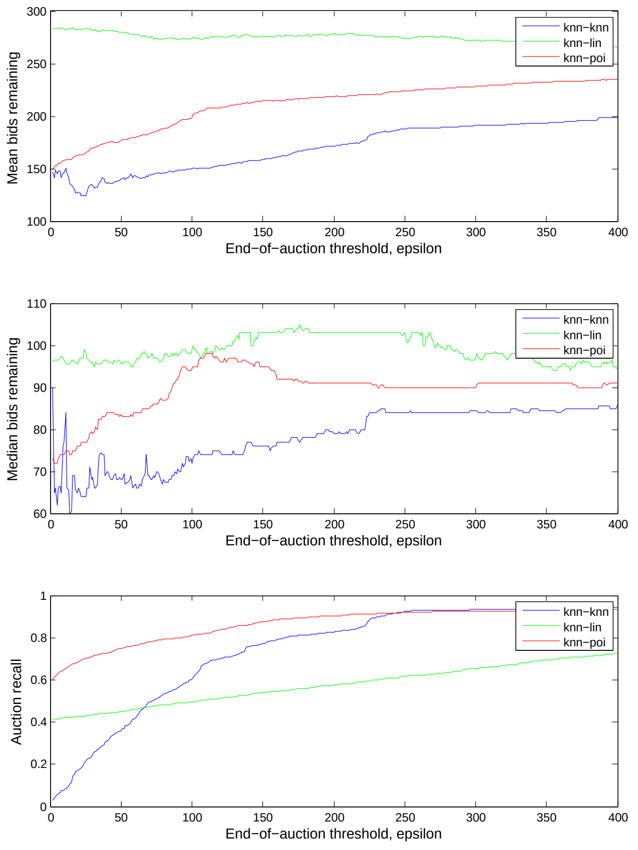

Figure 5: Evaluation metrics as a function of the threshold ϵ: the mean z_a (top), median z_a (middle), and the auction recall (bottom). Results for *knn-knn* are shown in blue, with *knn-lin* shown in green and *knn-poi* shown in red.

For nearly any choice of ϵ, *knn-knn* performs best in terms of the mean and median z_a metrics—there tend to be the fewest bids remaining in the test auction history when *knn-knn* predicts the impending end of the auction. The *knn-poi* method performs second best, with *knn-lin* performing worst (Figure 5, top and center panels). These results are consistent with the RSS in that *knn-knn* and *knn-poi* outperform *knn-lin*. Although *knn-knn* is better at identifying the auction end than *knn-poi*, the Poisson regression method has considerably better recall, especially for low values of ϵ (Figure 5, bottom panel). Poisson regression offers perhaps the best balance between precision and recall.

The *knn-knn* and *knn-poi* hierarchical methods are capable of predicting the end of the auction within a resolution of a few hundred bids. For example, setting the threshold to $\epsilon = 2$ and using *knn-poi*, the error is 149.9 bids as measured by mean z_a. The median z_a more conservatively suggests an error of 73 bids. The recall for *knn-poi* with $\epsilon = 2$ is 60.2%.

6 Conclusion

Although (J. Stix, 2012, [9]) suggests several rule-based strategies for a QuiBids bidding agent, to our knowledge no prior research has attempted to apply machine learning techniques for end-of-auction prediction in penny auctions. Given the dearth of research on automated agents for online penny auctions, we anticipate that this work will set a baseline for supervised learning-based penny auction agents.

Our results demonstrate the promise behind a hierarchical strategy which trains a model for each auction individually. The evaluation suggests that the hierarchical method is capable of obtaining end-of-auction predictions that err, on average, by less than 200 bids. There is a tradeoff between end-of-auction prediction precision and recall, but depending on the desired level of precision, *knn-knn* and *knn-poi* can achieve between 60% and 90% recall.

One direction for future work is to design a loss function which properly penalizes error at the end of a training auction more than error at the beginning of a training auction when computing the model parameters θ. This would ensure that each auction model closely fits the data at the end of the auction, and therefore makes accurate predictions when the end of the auction is imminent. A second direction for future work is to explore regression models for count data aside from Poisson regression. In particular, negative binomial regression might provide an improvement over *knn-poi*.

We believe that the discriminative models presented here for end-of-auction prediction could form the foundation for implementing a profitable QuiBids bidding agent. An alternative avenue which may be interesting to pursue is developing a generative model capable of modeling the underlying auction dynamics. A hidden Markov model might be applicable for this purpose given that the full bid histories are sequential data. Potentially of use here is a hidden semi-Markov model (HSMM) in which transition probabilities are not static, but rather depend on the time elapsed since entry into the hidden state.

References

[1] Augenblick, N. 2012. Consumer and producer behavior in the market for penny auctions: A theoretical and empirical analysis. Unpublished manuscript. Available at `http://faculty.haas.berkeley.edu/ned/Penny_Auction.pdf`.

[2] Beyer, K., Goldstein, J., Ramaskrishnan, R., and Shaft, U. 1999. When Is "Nearest Neighbor" Meaningful? 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings, 217–235: International Conference on Database Technology.

[3] Cameron, A. C. and Trivedi, P. 1998. Basic Count Regression. In *Regression Analysis of Count Data*, 61–69. New York: Cambridge University Press.

[4] Cormen, T., Leiserson, C., and Rivest, R. 2009. 3rd ed. *Introduction to Algorithms*. Cambridge, MA: MIT Press.

[5] Hinnosaar, T. 2010. Penny auctions are unpredictable. Unpublished manuscript. Available at `http://toomas.hinnosaar.net/pennyauctions.pdf`.

[6] Krishna, V. and Morgan, J. 1997. An Analysis of the War of Attrition and the All-Pay Auction. *Journal of Economic Theory* 72(2): 343–362.

[7] Pedregosa, F. et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12: 2825–2830.

[8] Stix, E., Stix, J., Storch, D., Sodomka, E., and Greenwald, A. 2013. Empirical Analysis of Auctioneer Profitability in QuiBids Penny Auctions. Association for the Advancement of Artificial Intelligence. Proceedings: AAAI-13 Workshop on Trading Agent Design and Analysis. Forthcoming.

[9] Stix, J. 2012. Designing a Bidding Algorithm for Online Penny Auctions. Available at `http://cs.brown.edu/research/pubs/theses/ugrad/2012/jstix.pdf`.

[10] Wang, Z. and Xu, M. 2012. Learning and Strategic Sophistication in Games: The Case of Penny Auctions on the Internet. Unpublished manuscript. Available at `http://www.economics.neu.edu/zwang/Penny%20auction.pdf`.

[11] Yianilos, P. 1993. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. Proceedings, 311–321: ACM-SIAM Symposium on Discrete Algorithms.