Appendix for Monopsony in Online Labor Markets

Arindrajit Dube, Jeff Jacobs, Suresh Naidu, Siddharth Suri

October 18, 2018

Appendix A Monopsony on Mechanical Turk

We assume a large number L of employers, denoted as i, who initially post N_i jobs, each worth p_i only if completed before time T_i. Jobs are completed instantaneously once accepted. Each job gets seen by λ myopic workers, whose reservation values for that are job given by $F(b)$. $F(b)$ could arise from a variety of random utility models. For example, if worker j's utility over job posted by employer i is given by $U_{ji} = \eta \ln(w_j) + \epsilon_{ji}$, where ϵ is Gumbel, then $F(w) \propto w^{\eta}$, delivering a constant elasticity labor supply curve facing the firm.[1]

While we proceed with a random utility interpretation of the idiosyncratic shock, Fosgerau et al. (2016) show a generic equivalence between rational inattention and random utility based discrete choice models (in particular the logit can be expressed as a rational inattention model with a Shannon entropy cost of information processing). While MTurk makes many work options available and easy to find, employers may have outsized market power either due to idiosyncratic tastes of workers for particular tasks (random utility) or due to costly information processing that makes it difficult to discern which task is best (rational inattention).

*Contact information: adube@econs.umass.edu, Department of Economics, 212 Crotty Hall, 411-417 North Pleasant St, University of Massachusetts Amherst, Amherst, MA 01002. jpj2122@columbia.edu, Department of Political Science, Columbia University, 422 West 118th Street, New York, NY, 10027, sn2430@columbia.edu, SIPA/Department of Economics, Columbia University, 422 West 118th Street, New York, NY, 10027. suri@microsoft.com, Microsoft Research, 641 Avenue of the Americas, 7th Floor New York, NY 10011. We thank Gary Hsieh and Panos Ipeirotis for sharing data as well as Bentley Macleod, Aaron Sojourner, and Glen Weyl for helpful comments.

[1] Gabaix et al. (2016) give conditions on the tail behavior of the distribution of ϵ such that, in a symmetric model with $p_i = p$, wages can increase towards p as L gets large (as would be intuitive). However they also given conditions under which the markdown stays constant or even increases as L gets large. Fat-tailed distributions of idiosyncratic utility imply that markdowns will remain substantial even in the presence of many firms.

Each employer posts a job and chooses a wage to maximize $\Pi(w) = \int_0^{T_i} exp(-rt)N(w,t)(p_i - w)F(w)\lambda dt$ subject to $\dot{N(w,t)} = -\lambda F(w)N(w,t)$.

When T_i is small, this profit function can be approximated (up to scale) by the static profit function:

$$\Pi(w_i) \approx (p_i - w_i)N_i T_i \lambda F(w_i) \tag{7}$$

The rate at which a batch is filled is thus $\lambda F(w)$ and the average duration of a batch is thus $d_i = min(T_i, \frac{N_i}{\lambda F(w_i)})$. The labor supply elasticity facing the firm is η. Taking a percent change approximation to the log we get:

$$\eta \approx \frac{dF(w)}{d\ln(w)} \frac{1}{\bar{F_{(w)}}} \tag{8}$$

In the general case, the first-order condition characterizing the wage is

$$\frac{p_i - w_i}{w_i} = \frac{1}{\eta - \Psi} \tag{9}$$

where $\Psi = \lambda w f(w)\big(\frac{1}{\lambda F(w)+r} - \frac{T}{exp(T(\lambda f(w)+r)-1}\big)$

Note that Ψ goes to 0 as r gets large or T gets small, and so equation 9 converges to the standard, static Lerner condition in either of these cases, both of which generate impatient requesters. If T is large and r small, the static approximation fails, and the gap between the marginal product and wage is larger than the static case. This is because the cost of paying a low wage for a requester (rejection of the offer) is attenuated by the fact that a rejection is potentially only temporary, as the job stays offered until filled or until time T.

Note that the assumption of a constant rate of offer arrival implies that the elasticity of the duration of a HIT with respect to w identifies $-\eta$. The duration of a HIT batch will be the time until an agent who will accept the offer sees the offer. Clearly if T_i is sufficiently large relative to $\frac{1}{\lambda F(w)}$ (or we have enough controls for T_i), then $\frac{d\ln(d_i)}{d\ln(w_i)} = -\eta$.

Identification of η is obtained by a) using machine-learned functions to control for batch properties (e.g. N_i as well as any other characteristics of the task that influence the distribution of worker reservation values besides the wage) in the double-ML approach and b) using the "honeypot" design described below to randomize w_i holding the batch properties constant.

We can obtain a separate estimate of the labor supply elasticity, η, using the "retention" experiments described in the text. The retention experiment involves a requester making a take it or leave it offer to a worker who has already agreed to a HIT. In principle, the worker has the same distribution of other HITs or

outside options so that the distribution of reservation values should be the same $F(b)$ as above, but we allow for the possibility that this elasticity is different from the η estimated from recruitment. The assumption of constant worker arrival rates, instantaneous job fulfillment, and no specific skills for a task suggests that these should be quite close, as both are recovering the log-curvature of F.

Appendix B Other Experiments Surveyed

We surveyed a large number of MTurk experiments, shown in 4. However, we did not include those that did not randomize the wage within the same batch. In a large number of MTurk studies, researchers will issue batches of HITs sequentially, with each batch being given a different wage[2]. The majority of these are not randomized and thus we cannot use them to recover even quasi-experimental requester's labor supply elasticities. See Table 4 for full explanations of the inclusion/exclusion criteria for each study.

[2]We examined the estimates in the following papers: Berinsky et al. (2012), Buhrmester et al. (2011), Crump et al. (2013), Doerrenberg et al. (2016), Heer and Bostock (2010), Horton and Chilton (2010), Marge et al. (2010), Mason and Watts (2009), Rogstadius et al. (2011), Sorokin and Forsyth (2008)

Study	Included	Reason
Berinsky et al. (2012)	No	HIT groups posted sequentially (not randomized)
Buhrmester et al. (2011)	No	HIT groups posted sequentially (not randomized)
Callison-Burch (2014)	No	Unable to obtain data
Chandler and Horton (2011)	No	Unable to obtain data
Crump et al. (2013)	No	HIT groups posted sequentially (not randomized)
Doerrenberg et al. (2016)	No	Piecemeal wage, non-honeypot setup
Dube et al. (2017)	Yes	Replicated
Heer and Bostock (2010)	No	HIT groups posted sequentially (not randomized)
Ho et al. (2015)	Yes	Randomized "honeypot" design
Horton and Chilton (2010)	No	Labor supply elasticity (0.34) imputed, not estimated directly
Horton et al. (2011)	Yes	Replicated
Huang et al. (2010)	No	Unable to obtain data
Hsieh and Kocielnik (2016)	Yes	Randomized "honeypot" design
Marge et al. (2010)	No	HIT groups posted sequentially (not randomized)
Mason and Watts (2009)	No	Piecemeal wage, non-honeypot setup
Rogstadius et al. (2011)	No	Common HIT pool creates non-independence of accept/reject decisions
Sorokin and Forsyth (2008)	No	HIT groups posted sequentially (not randomized)
Yin et al. (2018)	Yes	Randomized "honeypot" design

Table 4: All Experiments Surveyed

Appendix C Observational Data Appendix

A sample of the MTurk interface for workers can be found at the link `http://textlab.econ.columbia.edu/~snaidu/mturk.png`. We use two different scraping strategies. Section Appendix C.1 describes data from Ipeirotis (2010) obtained via the Mechanical Turk Tracker API[3], and goes from January 2014 through February 2016, when the account was ended by Amazon. Beginning in May 2016, we ran our own scraper, which took snapshots of all HITs available to a worker with a US address every 30 minutes, though the frequency was increased to every 10 minutes beginning in May 2017. Data from this latter scrape is described in Section Appendix C.2.

Appendix C.1 Data for January 2014 – February 2016

Ipeirotis (2010) introduces the Mechanical Turk Tracker, a web interface allowing researchers to view hourly market data (*e.g.*, number of HITs available) and demographic information (*e.g.*, proportion of workers who identify as male/female or who are from India/the United States) for the Amazon Mechanical Turk marketplace. An Application Programming Interface (API) is provided alongside the web interface, allowing for programmatic queries to be issued to the database. Using this API, we downloaded both "cross-sectional" data (*e.g.*, requester name, title, description, keywords) and "time series" data (number of HITs available in the group for each run of the scraper) for 410,284 HIT groups. Of these, 125,337 were either posted to the marketplace after February 1st, 2017 or had observations after this date, the date Amazon changed its interface and the scraper ceased working, and thus were dropped from our analysis. Of the remaining 284,947, we dropped any that had

- Zero-valued reward,
- Only one observation (since we are unable to compute durations for these groups), or
- Rewards or durations greater than the 99.5th percentile of their respective distributions (approx. 90,000 minutes for durations and $10.00 for rewards),

leaving us with 258,352 "final" observations, as seen in the leftmost panel of Table 5.

[3] `https://crowd-power.appspot.com/#/general`

	2014-2016 Scrape		2016-2017 Scrape		2017 Scrape	
	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Duration (Minutes)	3370.360	9414.101	3519.257	9721.523	2293.174	8375.199
Reward (Cents)	38.014	63.741	70.397	92.420	61.774	87.358
Log Reward ML Prediction	2.639	1.229	3.431	1.416	3.286	1.362
Log Duration ML Prediction	5.210	2.642	6.223	1.414	5.301	1.589
Log Duration ML Residuals	-0.004	0.892	-0.013	1.432	0.003	1.466
Log Reward ML Residuals	-0.001	0.679	-0.003	0.483	-0.001	0.459
Time Allotted (Minutes)	77.793	204.495	595.510	2916.676	434.435	2102.791
Max No. of HITs in Batch	83.413	1303.061	59.867	1627.825	53.539	931.335
Observations	258352		292746		93775	

Table 5: The leftmost panel presents summary statistics from scraping MTurk between Jan. 2014 and Feb. 2016. The middle panel presents analogous numbers using data obtained from May. 2016 through May 2017 (30 minute interval scrape). The last panel presents the same information for the May-August 2017 scrape at 10 minute intervals.

Appendix C.2 Data for May 2016 – August 2017

At the beginning of the project in May 2016, we set up a scraper which would log in to MTurk as a user with a US address and download all available information about each HIT group listed in the web interface. The scraper ran every 30 minutes (on the hour and on the half-hour) starting at midnight EST on May 31st 2016, though this was increased to every 10 minutes beginning at midnight EST on May 31st 2017. The scraper was finally banned by Amazon on August 21st 2017 at 7:30pm EST. The every-30-minute scrapes from May 2016 to May 2017 produced 363,181 total observations (292,746 after cleaning, as seen in the center panel of Table 5), while the every-10-minute scrapes from May 2017 to August 2017 produced 110,732 (93,775 after cleaning, as seen in the rightmost panel of Table 5).

Figure C.2 show how the distributions of log durations differ among the samples. The observed truncation is to be expected as the scraping windows for the 2016-2017 samples are different and will mechanically miss durations shorter than 30 and 10 minutes.

Appendix C.3 Heterogeneity Across Task Types

We can examine heterogeneity across task types using the classification of tasks developed by Gadiraju et al. (2014). Note that tasks are not uniquely categorized, as the same task can be in multiple categories, and many tasks fall in none of these categories. Figure C.3 shows the double-ML elasticity separately for each type of task plotted against our best proxy for the real wage, Log(Reward/Time Allotted). As would be expected if employers were using their monopsony power, higher wages would be associated with higher elasticities. Further, as is intuitive, (slightly) higher elasticities are found in HIT types with more posted

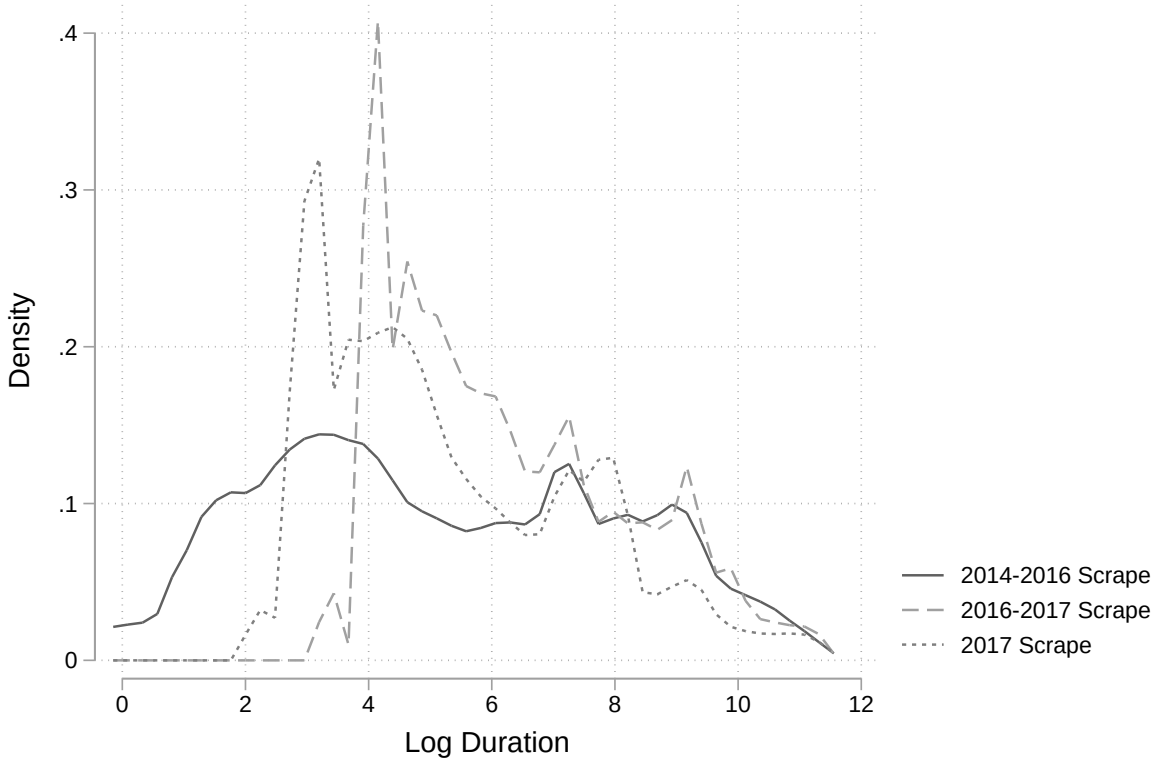

Figure C.1: Kernel density plots of log duration for the 3 different samples used in the analysis, described in Table 5.

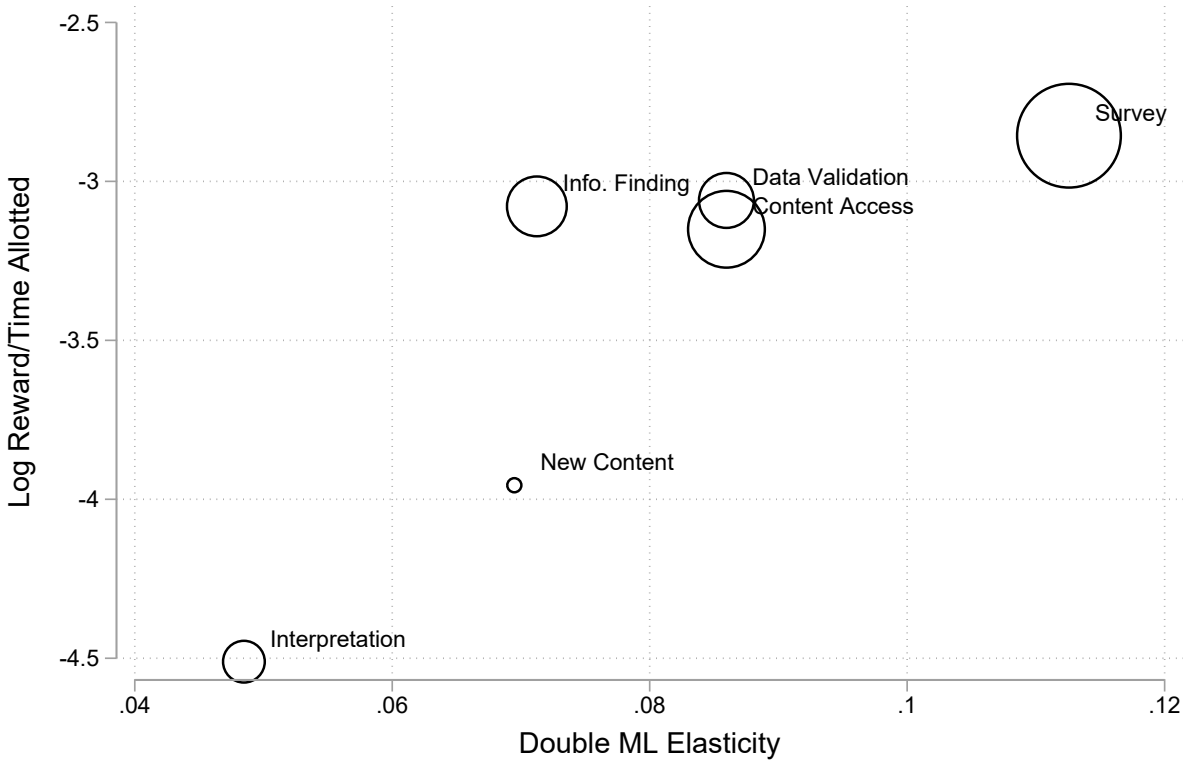

Figure C.2: Correlation Between Elasticity and Reward Per Minute Allotted. Dot size is proportional to the number of HIT batches of each type. $N = 235,940$.

batches, that is, higher volume, which could be the result of either more competition or more familiarity with workers.

Appendix D Full Double-ML Procedure

Appendix D.1 Data Loading/Merging

For each of our three datasets, the initial data processing proceeded as follows. First, a scraped panel dataset is loaded which contains, for each HIT group, the number of HITs available and the timestamp of each scrape in which the group was observed. This panel data then gets collapsed into a cross-sectional dataset consisting of several features derived from the distribution of the timestamps and HITs available – for example, into min(timestamp), max(timestamp), min(hits_available), and max(hits_available) for each HIT group. Then, a separate cross-sectional metadata file (containing, for example, the titles, descriptions, and requester names for each HIT group) is merged into the collapsed panel dataset via the unique Amazon-supplied group ID[4].

Appendix D.2 Data Cleaning

All observations with a reward greater than $5 or duration greater than 90,000 minutes (approximately two months) are dropped[5]. Then all observations with 0 reward or 0 duration values (only occurring in the 2014-2016 scrape data) are dropped, to allow transformation of the dependent variables into log space. This produces the final set of observations used in the machine learning procedure itself, which are summarized in Table 5.

Appendix D.3 Feature Selection and Test/Training Split

We transform the text scraped with each HIT batch into a large number of text features as follows:

- **N-grams**: An n-gram is an ordered sequence of n words. For example, if the full description for a HIT is "quick transcription task," this will produce three 1-grams "quick," "description" and "task"; two 2-grams "quick description" and "description task"; and a single 3-gram "quick description task." We use sliding windows of 1 to 3 words over all words within the title, HIT description and keyword list to form 1, 2 and 3-grams. The frequency of these n-grams in each HIT is then a feature used by the ML algorithm. We use the standard English stopword list in Scikit-learn to eliminate stopwords.

- **Topic Distributions**: Besides ordered sequence of words, sometimes sets of particular words ("topic") convey important information. A topic model is essentially an algorithm which searches for sets of words

[4]This final cross-sectional file contains 411,196 observations for the Jan 2014 - Feb 2016 data, 363,181 for the May 2016 - May 2017 data, and 110,732 for the May 2017 - Aug 2017 data, as described in the previous section.

[5]These values correspond approximately to the 99.5th percentiles of the original distribution.

that tend to occur together in a corpus. For example, one of our topics identifies the words "image," "text," and "transcribe" as its top words. HITs requesting transcription of text from an image will tend to have high feature values for this topic and lower values for other topics. The resulting features for each HIT is then the distribution over topics found in that HIT's title, description, and keyword list.[6] We use the NLTK English stopword corpus to drop stopwords. The top 5 words for each topic model run with $K \in \{5, 10, 15, 20\}$ are available online at `textlab.econ.columbia.edu/topicwords.pdf`.

- **Doc2Vec Embeddings**: Unlike LDA which tries to generate features by splitting *documents* into discrete human-interpretable topics, the goal of Doc2Vec is to generate a vector space in which vectors for *words* which are semantically similar are close together, and then infer a document-level vector within this same vector space via amalgamation of the learned vectors for its constituent words. For example, since "survey" and "questionnaire" are semantically similar in the sense that they are used in similar contexts ("a short [survey/quetionnaire]", "fill out this [survey/questionnaire]"), their vectors will be close together in the constructed vector space, and this will "pull" the document-level vectors for descriptions containing either word closer together. [7]

- **Hand-Engineered Features**: Finally, we use a set of custom regular-expression-based features, which are generally binary variables describing the presence or absence of certain salient keywords (*e.g.*, "survey", "transcribe"), but also real-valued variables capturing (for example) time estimates given in the titles/descriptions (*e.g.*, "5-minute survey"). The bulk of these features are derived from the explicit features described in Difallah et al. (2015), and the HIT taxonomy scheme developed in Gadiraju et al. (2014). The hand-engineered features are as follows:

  - Based on common patterns we observed in HIT titles, descriptions, and keywords, dummy variables were created indicating the presence or absence of the following regular expressions: *easy*, *transcr** (capturing, e.g., "transcription" or "transcribe"), *writ** (capturing, e.g., "written", "write", or "writing"), *audio*, *image/picture*, *video*, *bonus*, *copy*, *search*, *ident** (capturing, e.g., "identify"), *text*, *date*, *fun*, *simpl**, *summar**, *only*, *improve*, *five/5*, *?*, and *!*.
  - Based on the HIT taxonomy scheme developed in Gadiraju et al. (2014), a numerical category was assigned to each HIT group via the following regular expressions:

[6] We run a Latent Dirichlet Allocation (LDA) topic model model (Blei et al. (2003)) on all descriptions. LDA requires the choice of a parameter K which determines how many topics the algorithm should try to discover: we estimate models with $K \in \{5, 10, 15, 20\}$.

[7] We run Doc2Vec model Le and Mikolov (2014) on all titles, descriptions, and keywords in the data, producing a 50-dimensional semantic information vector for each.

    - **Information Finding (IF)**: *find*
    - **Verification and Validation (VV)**: *check*, *match*
    - **Interpretation and Analysis (IA)**: *choose*, *categor**
    - **Content Creation (CC)**: *suggest*, *translat**
    - **Surveys (S)**: *survey*
    - **Content Access (CA)**: *click*, *link*, *read*
- The following numeric features were extracted, some of which were derived from features used in Difallah et al. (2015):
    - `time_allotted`: The time a worker is given to complete a given HIT
    - `time_left`: The time remaining before the HIT group expires (expired HIT groups are removed from the marketplace)
    - `first_hits`: The number of HITs initially posted to the marketplace
    - `last_hits`: The number of HITs remaining to be completed in the group at the time it was last observed
    - `min_hits`: The minumum number of HITs available observed for the group across all scrapes
    - `max_hits`: The maximum number of HITs available observed for the group across all scrapes
    - `avg_hitrate`: The average rate (per hour) at which HITs within the group were filled by workers
    - `avg_hits_completed`: The average change in available HITs between subsequent observations of the group
    - `med_hits_completed`: The median change in available HITs between subsequent observations of the group
    - `min_hits_completed`: The minimum change in available HITs between subsequent observations of the group
    - `max_hits_completed`: The maximum change in available HITs between subsequent observations of the group
    - `num_zeros`: The number of observations for which the number of available HITs in the group was listed as 0
    - `req_mean_reward`: The average reward over all HITs posted by the requester

* `req_mean_dur`: The average duration of all HIT groups posted by the requester
* `title_len`: The length of the HIT group's title
* `desc_len`: The length of the HIT group's description
* `keywords_len`: The sum of the lengths of the HIT group's keywords
* `num_keywords`: The number of keywords given for the HIT group
* `title_words`: The number of words in the HIT group's title
* `desc_words`: The number of words in the HIT group's description
* `minutes_title`: The number of minutes if a phrase including "X minutes" appears in the title
* `minutes_desc`: The number of minutes if a phrase including "X minutes" appears in the description
* `minutes_kw`: The number of minutes if a phrase including "X minutes" appears in the keyword list
* `qual_len`: The length of the string given in the HIT group's description which lists the qualifications
* `num_quals`: The number of qualifications required for the HIT group
* `custom_not_granted`: The number of custom qualifications required for the HIT group for which our "blank" account (an account which had never accepted or completed a HIT) was not qualified
* `custom_granted`: The number of custom qualifications required for the HIT group for which our "blank" account (an account which had never accepted or completed a HIT) was qualified
* `any_loc`: A dummy variable representing whether or not the HIT group had a location restriction (e.g., US only)
* `us_only`: A dummy variable which is 1 if the HIT group is restricted to US workers, and 0 otherwise
* `appr_rate_gt`: The lower bound on approval rate required for workers to be eligible for the HIT, coded as -1 if no lower bound was enforced
* `rej_rate_lt`: The upper bound on rejection rate required for workers to be eligible for the HIT, coded as 101 if no upper bound was enforced

* `appr_num_gt`: The lower bound on number of approvals required for workers to be eligible for the HIT, coded as -1 if no lower bound was enforced
* `rej_num_lt`: The upper bound on number of rejections required for workers to be eligible for the HIT, coded as 999 if no upper bound was enforced
* `adult_content`: A dummy variable which is 1 if the HIT group indicated that it contained adult content, and 0 otherwise

To save on computation time, we utilized a "two stage" double ML procedure as outlined in Section 3.3. Given the initial split of the data into A and B sets, and the subsequent split of these sets into A_{train}, A_{val}, B_{train}, and B_{val}, a given run of our procedure ($A \to B$ or $B \to A$; here we describe the $A \to B$ run without loss of generality) proceeds as follows. First, in the "feature selection" phase, the full set of n-gram features were generated as described above, and a "preliminary" run of the learning algorithm was performed using *only* these n-gram features as the feature matrix for A_{train}, with the goal of predicting the reward and duration values in A_{val}. Upon completion of this stage, we "threw away" all but the top 100 most predictive n-gram features for reward and top 100 most predictive n-gram features for duration, and for the remainder of the run only those n-gram features were included. For illustration, the top 100 most predictive reward and duration features for the $A \to B$ run on the Jan 2014 - Feb 2016 data are available online at `textlab.econ.columbia.edu/top100_text_reward.pdf` and `textlab.econ.columbia.edu/top100_text_duration.pdf`, respectively.

Once this feature selection phase was complete, a second "full data" phase was performed, as outlined in Section 3.3. In this phase, the top 200 n-gram features from the feature selection phase are included in the feature matrix for A along with the LDA, Doc2Vec, and hand-engineered features, with the goal of predicting the reward and duration values in B.

Appendix D.4 Regression via Random Forests

Before computing predictions on the test set, the validation set was used to tune not only hyperparameters but also which learning method was chosen. Random forest regression, implemented by RandomForestRegressor in scikit-learn, greatly outperformed the other classifiers we employed: {AdaBoostRegressor, BaggingRegressor, ExtraTreesRegressor, GradientBoostingRegressor, RandomForestRegressor, SVR (Support-VectorRegressor)}, and thus a trained random forest regression was our choice for computing predictions for the test data. The random forest method constructs a series of individual decision tree estimators, where each regressor is trained on a subset of the full feature set, and then reports the mean prediction over all

	Jan 2014 - Feb 2016		May 2016 - May 2017		May 2017 - Aug 2017	
	$A \to B$	$B \to$ A	$A \to B$	$B \to A$	$A \to B$	$B \to A$
Reward	0.7716	0.7764	0.8951	0.8949	0.8982	0.8984
Duration	0.8968	0.8980	0.4379	0.4404	0.5085	0.5035

Table 6: R^2 scores for each run of the Double-ML regressions

regressors. Based on two additional cross-validation procedures, for the first-stage feature selection an random forest regression with 40 decision tree estimators was used (with the number of estimators optimized over {10, 20, ..., 100}) while for the second-stage ML the model was run with 600 estimators (optimized over {100, 200, ..., 1000}, with the increased order of magnitude made feasible due to the fact that in the second stage all but 200 of the approximately 800,000 total n-gram features are dropped).

Appendix D.5 Computing the Double-ML estimate

Once the ML algorithm has finished its runs and the predicted log duration and log reward values have been generated for each fold of the data, the estimated η value is computed straightforwardly via Equation 5 with $n = 2$.