Technical Report

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
email@domain

Abstract

1 Introduction

We examine the problem of crowdsourcing translation. Previous work has shown that non-professional translators hired on Amazon Mechanical Turk (MTurk) can achieve professional-level quality, by soliciting multiple translations of each source sentence and then choosing the best translation (Zaidan and Callison-Burch, 2011). In this paper, we explore the possibility of reducing costs by filtering out workers who produce low quality translations. Our goal is to reduce the number of independent translations that we solicit for each source sentence. We propose several ranking methods to rank workers and to to evaluate their competency. We evaluate workers' competency in several ways: first, we estimate their average quality using all of their translations, then we limit ourselves to the first k-translations that they provide. If we are able to quickly distinguish between high quality versus low quality translators, then we can reduce costs by only soliciting translations from high quality translators.

2 Experiment Setup

2.1 Data Collection

We use the data collected by Zaidan and Callison-Burch (2011) through Amazon's Mechanical Turk(MTurk). MTurk is an online platform provided to people for completing Human Intelligence Tasks(HIT) with a relatively low cost. We use their Urdu-to-English 2009 NIST Evaluation Set as our corpus. Zaidan and Callison-Burch (2011) translated the Urdu side to English through MTurk. They collected the translations in the unit of Human Intelligence Tasks(or HITs). In every HIT, they posted 10 Urdu sentences to be translated. Every sentence is translated by 4 workers, and subsequently post-edited by 10 additional workers.[1] This data set also has four corresponding professional translations for each of the Urdu sentences, collected by LDC. This makes it possible to compare the Turkers' translation quality to professionals.

2.2 Data Processing

Given all the translations and edits with the source sentence, we process the data to make it usable for model training. We tokenize the text using Penn TreeBank Tokenizer (Marcus et al., 1993). Furthermore, we use term frequency(or TF) model and term frequencyinverse document frequency(or TF-IDF) model to represent the tokenized text.

2.3 Feature Extraction

Following Zaidan and Callison-Burch (2011), we extract a number of features from the translations and workers' self-reported language skills. We use these to build feature vectors used in tuning model and choosing the best translations from the candidates. POTENTIALLY: We extend Zaidan and

[1] Zaidan and Callison-Burch (2011) collected their translations in two batches. The first batch contained 1 translation, each with 1 post-edited version. The second contained an additional 3 translations, each of which was post-edited by 3 workers.

Callison-Burch (2011)'s feature set to include additional bilingual features, which were not part of that original work.

2.3.1 Sentence-Level Features (9 Features)

This feature set contains language based features to solely implicate the quality of an English sentence without any suggestion on the bond of the meaning between the source sentence and the translation . This set of features tells good English sentences apart bad ones. The reason we use this set of features is that a good English sentence is the prerequisite of being a good English translation.

- Language model features: we assign a log probability and a per-word perplexity score for each sentence. We use SRILM toolkit to calculate perplexity score for each sentence based on 5-gram language model trained on English Gigaword corpus.

- Sentence length features: we use the ratio of the length of the Urdu source sentence to the length of the translation sentence as feature since a good translation is expected to be comparable in length with source sentence. We add two such ratio features(one is designed for unreasonably short translation and the other is for unreasonably long translation).

- Web *n*-gram log probability feature: we add the Web *n*-gram log probability feature to reflect the probability of the *n*-grams(up to length 5) exist in the Microsoft Web N-Gram Corpus. For short sentences whose length is less than 5, we use the sentence length as the order of the *n*-gram in calculation.

- Web *n*-gram geometric average features: we calculate the geometric average *n*-gram to evaluate the average matching over different *n*-grams. We add 3 features correspondent to max *n*-gram length of 3,4 and 5. Specifically, P_i denotes the log probability of *i*-gram and these 3 features are represented in $\sqrt[3]{P_1P_2P_3}$,$\sqrt[4]{P_1P_2P_3P_4}$ and $\sqrt[5]{P_1P_2P_3P_4P_5}$.

- Edit rate to other translations: In posterior methods, to minimize Bayes risk, we choose the translation that is most similar to other translations. Taking this into consideration, we add the edit rate feature to implement the similarity among all candidates translations.

2.3.2 Worker-Level Features

We take the quality of workers into consideration and add worker level features since the intuition that good workers can always high quality translations.

- Aggregate features: for each sentence level feature, we use the average values over all translations provided by the same worker as that worker's aggregate feature values.

- Language abilities: we collect worker's language ability information about whether the worker is a native Urdu speaker or native English speaker and how long they have spoken English or Urdu and add four features correspondent to the four aspects above.

- Worker Location: we add two features to indicate whether a worker is located in Pakistan or India.

2.3.3 Ranking Features

Zaidan and Callison-Burch (2011) collected 5 ranking labels for each translation and refine 3 features from these labels.

- Average Ranking: the average of the 5 ranking labels for this translation.

- Is-Best percentage: this feature shows how often a translation is ranked as the best translation among all candidates translation.

- Is-Better percentage: how often a translation is ranked as a better translation based on the pairwise comparisons.

2.3.4 Calibration Feature

We use the average BLEU score of translations provided by the same worker as the calibration feature. The BLEU score is computed against references.

2.3.5 Word Alignment Feature

We

2.4 Goal

We are trying to predict the competency of each worker and workers ranking list. The more the prediction is similar to the values against the professional references, the better the effect will be.

2.5 Gold Standard Ranking

To evaluate the correctness of the rating for a worker , we build up a gold standard metric. Given a worker *t* and all the candidates translations translated by *t* denoted as $C^t = \{c_i^t \mid c_i^t \text{ authored by t}\}$, the rating score of *t* is represented as

$$score(t) = \sum_{c_i^t \in C_i^t} (avgBLEU(c_i^t, refset(c_i^t))) \frac{1}{|C_i^t|} \tag{1}$$

where *refset(c)* denotes the references set for the candidate sentence *c*, and *avgBLEU* denotes the average BLEU score of this candidate translation computed against references. The average BLEU score is the mean of four numbers. There are four possible ways to choose 3 of 4 references and each number is the BLEU score of the candidate translation computed against 3 of 4 references. Given all candidates translations translated by worker *t*,the rating for *t* is the average score among each sentence *c* translated. We rank workers according to their scores and obtain the gold standard ranking list.

2.6 Models

We train a linear model and a decision tree model to score each translation.

2.7 Model Tuning

In our approach, we use five-fold cross validation to tune models. To be more specific, we use 80% data to compute worker aggregate features, use 10% in the 80% portion as training data and use remaining 20% data as testing data. For the worker calibration feature, we utilize 10% references commensurate with the 10% data we used in training to calculate. Each HIT consist of 10 sentences and translated by a group of 4 workers. To guarantee all workers are covered in the 10% training data, we choose the translation set (4 translations) of 1 source sentence from each HIT which has 10 source sentences totally.

2.8 Models For Machine Translation

2.8.1 Linear Model

We use Z-MERT software package which is developed by Zaidan (2009) on the basis of Minimum Error Rate Training (MERT) proposed by Och (2003) to tune the linear model. MERT is an iterative algorithm to tune the weight vector. In every iteration, it generates n-best translations and adjusts weight vector to fit these translations according to their qualities.

Feature Set	BLEU Score
Sentence features	38.55
Worker level features	37.87
Ranking features	36.74
Calibration features	38.26
All features (Calibration)	38.99

Table 1: BLEU score for Linear Model trained from different feature set

2.8.2 Decision Tree Model

We use the WEKA software package (Hall et al., 2009) to train a regression tree as our model and prune it using reduced-error pruning (with backfitting). The number of folds for reduced error pruning is 3.

Feature Set	BLEU Score
Sentence features	35.29
Worker level features	37.77
Ranking features	36.12
Calibration features	38.26
All features (Calibration)	37.70

Table 2: BLEU score for Decision Tree Model trained from different feature set

2.8.3 Linear Regression Model

We use the WEKA software package (Hall et al., 2009) to train a linear regression model as our model.

2.9 Models For Ranking Worker

As mentioned above, we train a linear model (MERT) to score each candidate translation and choose the best translation. We can also use each

Feature Set	BLEU Score
Sentence features	37.77
Worker level features	37.18
Ranking features	35.70
Calibration features	38.26
All features (Calibration)	39.43

Table 3: BLEU score for Linear Regression Model trained from different feature set

translation's score to rate the worker by averaging the total score of translations provided by that worker and rank workers based on their ratings.

To illustrate the similarity between the gold standard rating and the human predicted rating, we draw the workers' rating comparison graph to show the relationship between gold standard rating and model predicted rating. In this graph, each point represents a worker with gold standard rating (x-axis) and human predicted rating (y-axis). We set the average human predicted rating score as the threshold to select workers. The threshold divides the whole graph to 4 quadrants. The top right are correctly accepted, the top left are incorrectly accepted, the bottom left are correctly rejected and the bottom right are incorrectly rejected. More reasonably, we draw a similar graph to show the relationship between gold standard rating and model predicted rating with two thresholds: one is the mean of model predicted rating which is used to select the workers and the other is the mean of gold standard rating which is used to measure the correctness of the selection based on the first threshold. The more accurate the model is, the more closer these two thresholds.

To illustrate the similarity between the gold standard ranking and the model predicted ranking, we draw the workers' ranking comparison graph to show the model predicted ranking's fitting degree to the gold standard ranking. In this graph, each bubble represents a worker with gold standard ranking (x-axis) and model predicted ranking (y-axis). The radius of each bubble shows the number of sentences the correspondent worker provided.

Besides graphs, we also calculate the Pearson's Correlation to the show the similarity between the gold standard ranking and the model predicted ranking.

2.9.1 Linear Model

Feature Set	Prson's	Sprman's
Sentence features	0.10	0.10
Worker features	0.43	0.43
Ranking features	0.78	0.78
Calibration feature	0.98	0.98
All features(Calibration)	0.75	0.75

Table 4: Pearson's Correlation for Linear Model trained from different feature set

We train the linear model using sentence level feature to rate workers' performance and rank them. Figure 1 and Figure 2 show the relation between gold standard rating and model predicted rating. Figure 3 shows the relation between gold standard ranking and model predicted ranking. Figure 1

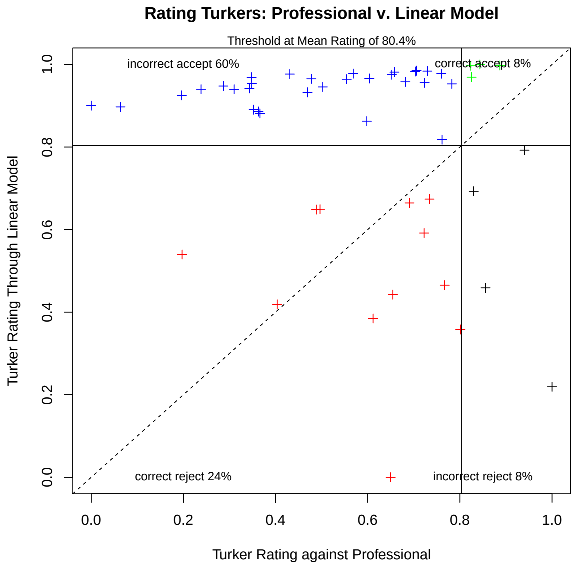

Figure 1: Relationship between model predicted rating and gold standard rating. We use sentence level feature to train a linear model. The threshold is the mean of the model predicted rating.

We train the linear model using worker level features to rate workers and rank them. Figure 4 and 5 show the relation between gold standard rating and model predicted rating. Figure 6 shows the relation between gold standard ranking and model predicted ranking.

We train the linear model using ranking features to rate workers and rank them. Figure 7 and 8

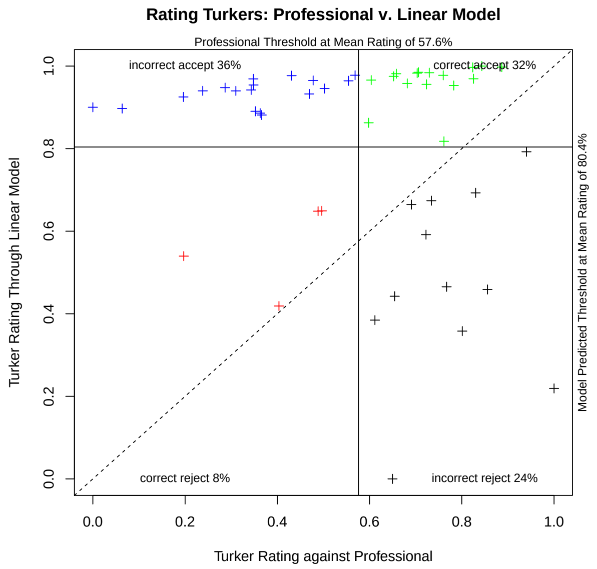

Figure 2: Relationship between model predicted rating and gold standard rating. We use sentence level feature to train a linear model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

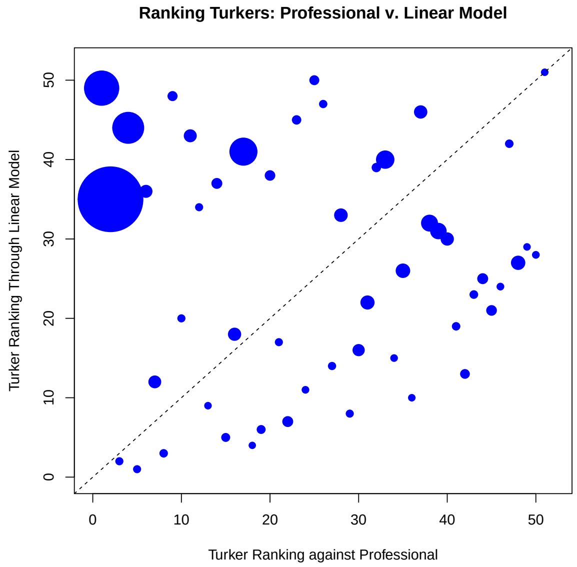

Figure 3: Relationship between gold standard raking and model predicted ranking. We use sentence level feature to train a linear model.

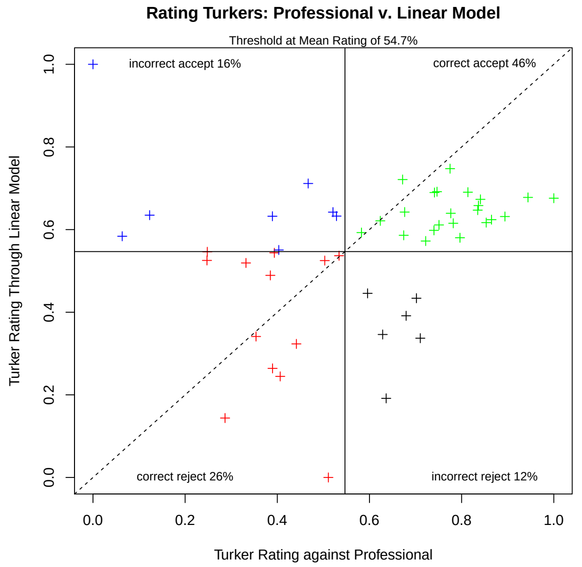

Figure 4: Relationship between model predicted rating and gold standard rating. We use worker level feature to train a linear model. The threshold is the mean of the model predicted rating.

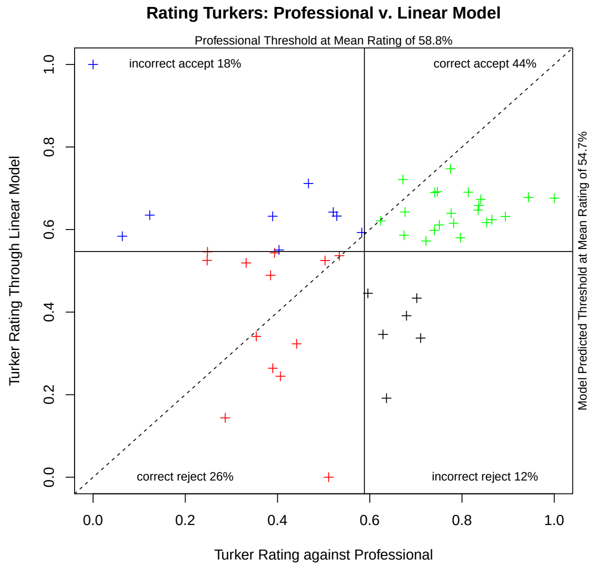

Figure 5: Relationship between model predicted rating and gold standard rating. We use worker level feature to train a linear model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

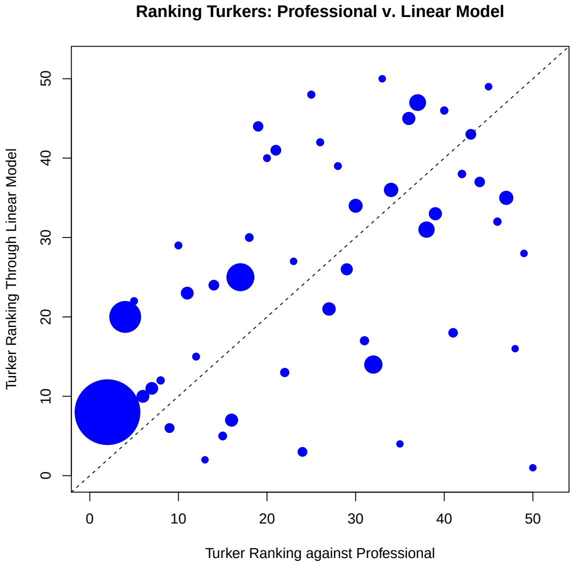

Figure 6: Relationship between gold standard ranking and model predicted ranking. We use worker level feature to train a linear model.

show the relation between gold standard rating and model predicted rating. Figure 9 shows the relation between gold standard ranking and model predicted ranking.

We train the linear model using all features with calibration to rate workers and rank them. Figure 10 and 11 show the relation between gold standard rating and model predicted rating. Figure 12 shows the relation between gold standard ranking and model predicted ranking.

2.9.2 Decision Tree Model

We also implement a decision tree model to rank workers. We train the decision tree model using sentence level features to rate workers and rank them. Figure 13 and 14 show the relation between gold

Feature Set	Prson's	Sprman's
Sentence features	0.86	0.86
Worker level feature	0.88	0.88
Ranking feature	0.84	0.84
Calibration feature	0.98	0.98
All features (Calibration)	0.95	0.95

Table 5: Pearson's Correlation for Decision Tree Model trained from different feature set

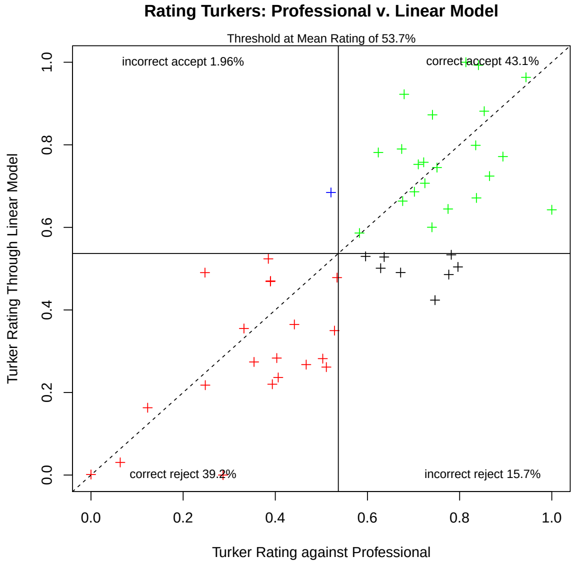

Figure 7: Relationship between model predicted rating and gold standard rating. We use ranking feature to train a linear model. The threshold is the mean of the model predicted rating.

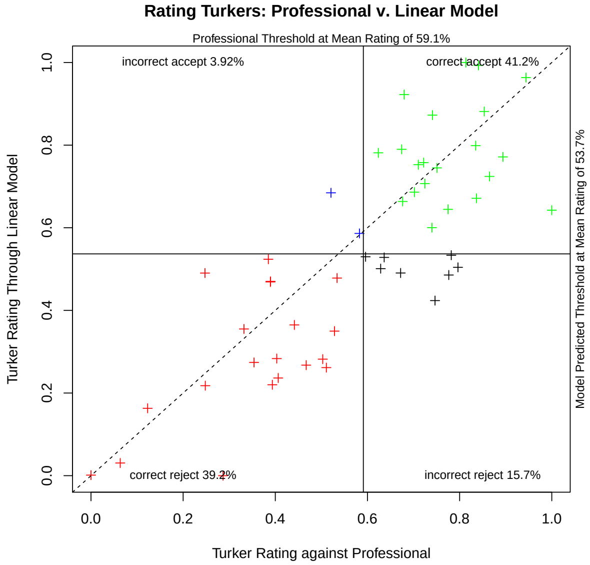

Figure 8: Relationship between model predicted rating and gold standard rating. We use ranking feature to train a linear model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

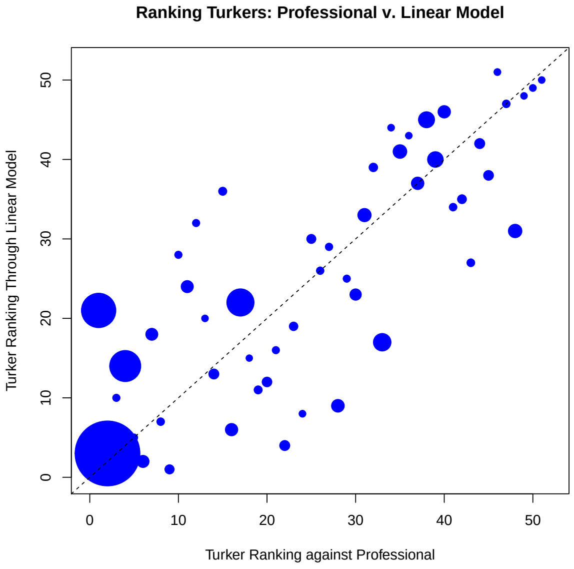

Figure 9: Relationship between gold standard ranking and model predicted ranking. We use ranking feature to train a linear model.

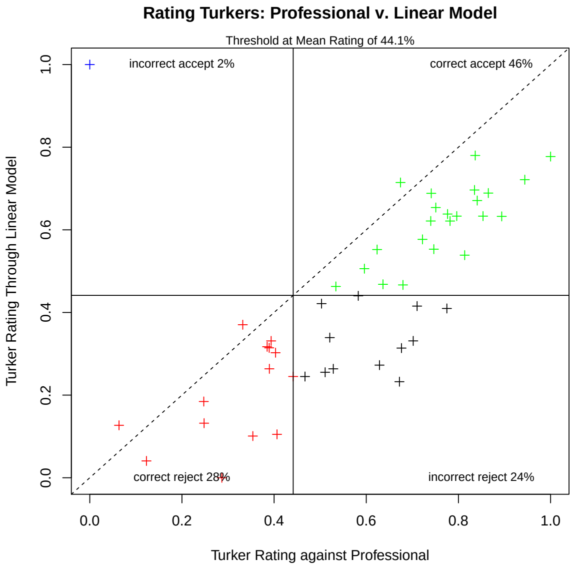

Figure 10: Relationship between model predicted rating and gold standard rating. We use all features (calibration) to train a linear model. The threshold is the mean of the model predicted rating.

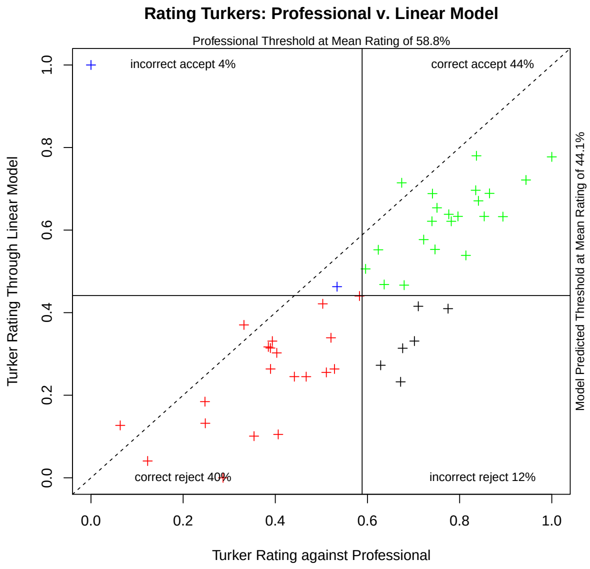

Figure 11: Relationship between model predicted rating and gold standard rating. We use all features (calibration) to train a linear model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

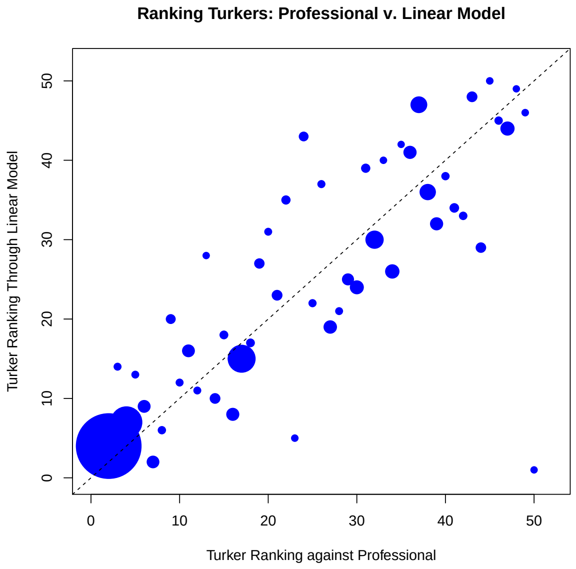

Figure 12: Relationship between gold standard ranking and model predicted ranking. We use all features (calibration) to train a linear model.

standard rating and model predicted rating. Figure 15 shows the relation between gold standard ranking and model predicted ranking.

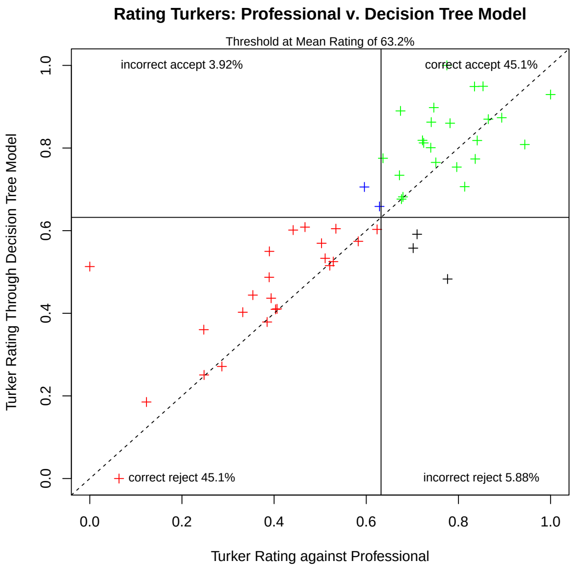

Figure 13: Relationship between model predicted rating and gold standard rating. We use sentence level feature to train a Decision Tree model. The threshold is the mean of the model predicted rating.

We train the decision tree model using worker level features to rate workers and rank them. Figure 16 and 17 show the relation between gold standard rating and model predicted rating. Figure 18 shows the relation between gold standard ranking and model predicted ranking.

We train the decision tree model using ranking features to rate workers and rank them. Figure 19 and 20 show the relation between gold standard rating and model predicted rating. Figure 21 shows the relation between gold standard ranking and model predicted ranking.

We train the decision tree model using all features with calibration to rate workers and rank them. Figure 22 and Figure 23 show the relation between gold standard rating and model predicted rating. Figure 24 shows the relation between gold standard ranking and model predicted ranking.

2.9.3 Linear Regression Model

We train Linear Regression Model to rank workers.

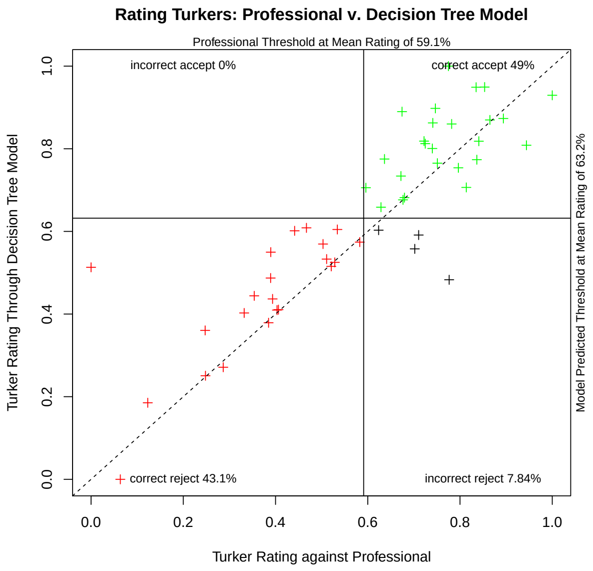

Figure 14: Relationship between model predicted rating and gold standard rating. We use sentence level feature to train a Decision Tree model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

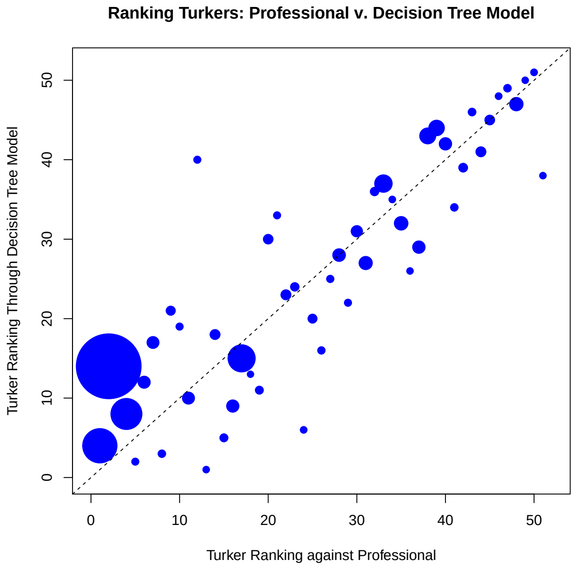

Figure 15: Relationship between gold standard raking and model predicted ranking. We use sentence level feature to train a Decision Tree model.

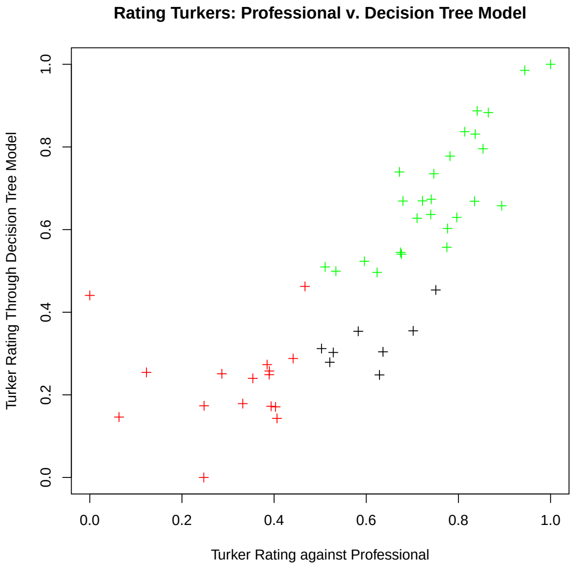

Figure 16: Relationship between model predicted rating and gold standard rating. We use worker level feature to train a Decision Tree model. The threshold is the mean of the model predicted rating.

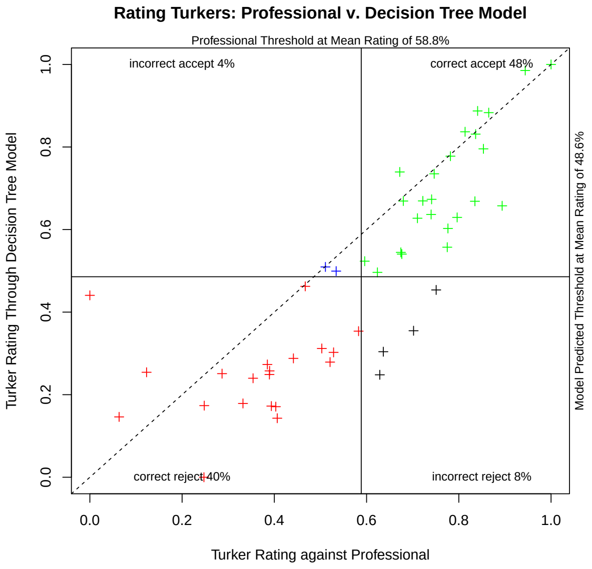

Figure 17: Relationship between model predicted rating and gold standard rating. We use worker level feature to train a Decision Tree model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

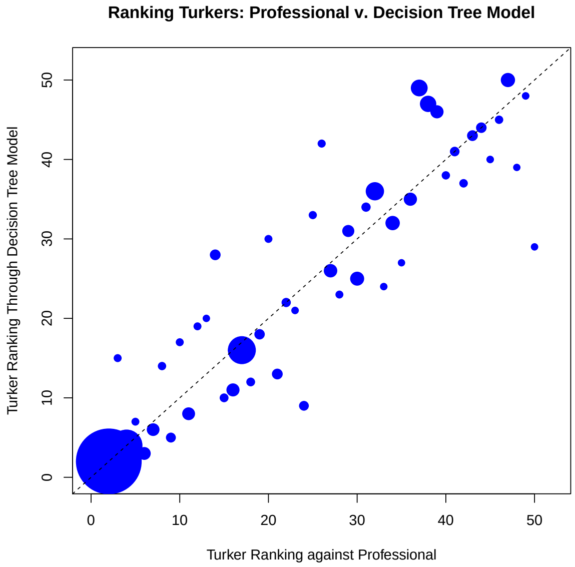

Figure 18: Relationship between gold standard raking and model predicted ranking. We use worker level feature to train a Decision Tree model.

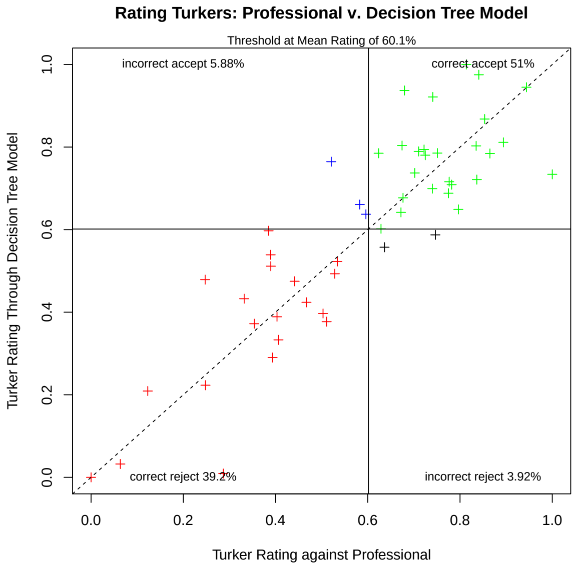

Figure 19: Relationship between model predicted rating and gold standard rating. We use ranking level feature to train a Decision Tree model. The threshold is the mean of the model predicted rating.

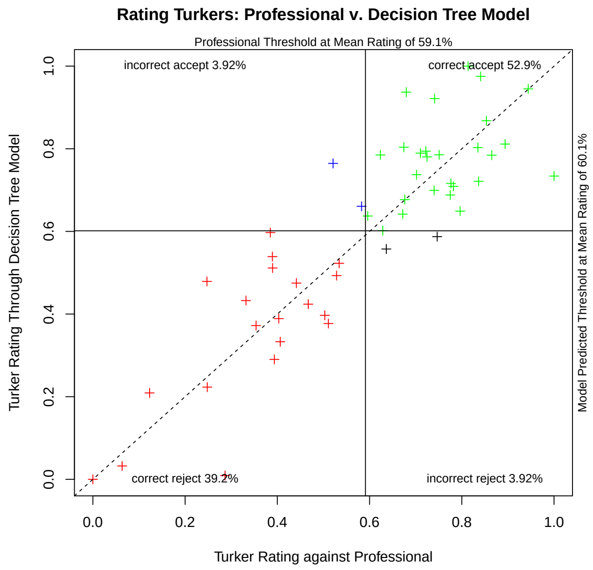

Figure 20: Relationship between model predicted rating and gold standard rating. We use ranking feature to train a Decision Tree model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

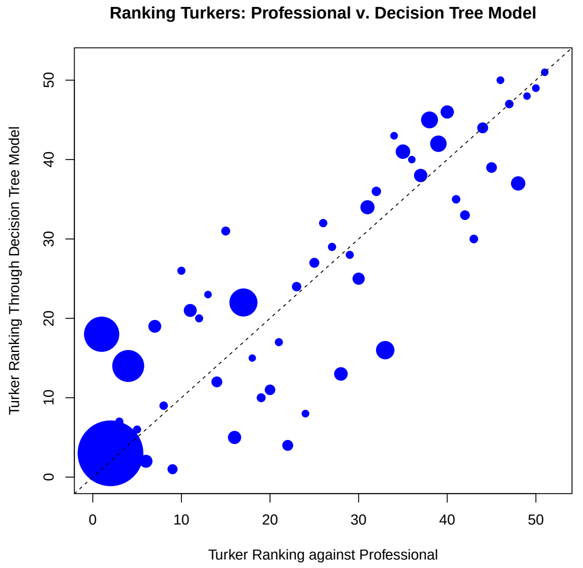

Figure 21: Relationship between gold standard raking and model predicted ranking. We use ranking level feature to train a Decision Tree model.

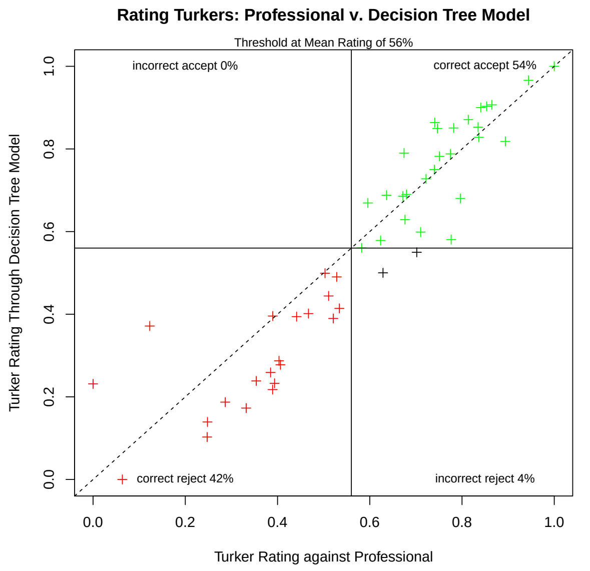

Figure 22: Relationship between model predicted rating and gold standard rating. We use all features with calibration to train a Decision Tree model. The threshold is the mean of the model predicted rating.

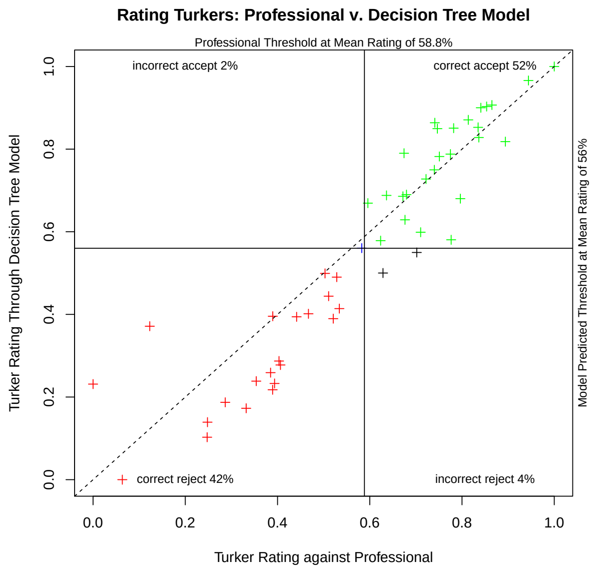

Figure 23: Relationship between model predicted rating and gold standard rating. We use all features with calibration to train a Decision Tree model. There are two thresholds: one is the mean of model predicted rating, and the other is the mean of gold standard rating.

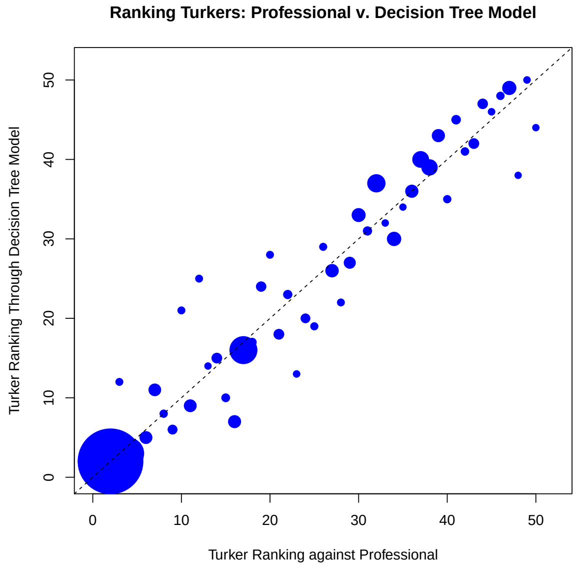

Figure 24: Relationship between gold standard raking and model predicted ranking. We use all features with calibration feature to train a Decision Tree model.

Feature Set	Prson's	Sprman's
Sentence features	na	na
Worker level feature	na	na
Ranking feature	na	na
Calibration feature	na	na
All features (Calibration)	0.97	0.97

Table 6: Pearson's Correlation for Linear Regression Model trained from different feature set

We train Linear Regression Model using all features with calibration to rate workers and rank them. Figure 25 shows the relation between gold standard ranking and model predicted ranking.

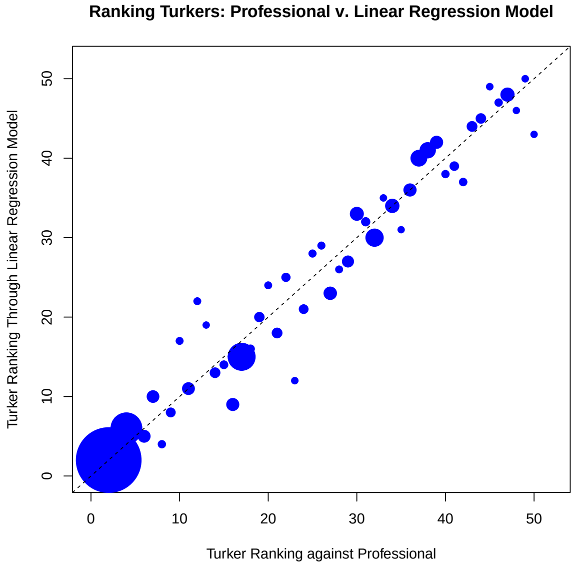

Figure 25: Relationship between gold standard raking and model predicted ranking. We use all features with calibration feature to train a Linear Regression model.

2.9.4 Data Analysis

We analyze the data considering the timing information that accompanies the translations that we collected. In this analysis, we'd like to evaluate the performance of each worker as time goes on. Each worker translates one or more HITs. Since the translations were collected in two batches, which started at different times, we assign a relative time to each assignment to simulate what would have happened if both batches were run at the same time. For each HIT, we assign it a relative time by calculating difference between the HIT's submission time and the HIT's creation time. This gives the HIT's relative submission time. For each worker, we calculated the BLEU score for each of their HITs and analyze the translation quality of the worker at different time points. We are interested in seeing whether workers produce consistently good translations over time or if their quality drops off over time, and if workers who produce bad translations submit them more quickly than workers who submit good translations.

Figure 26 illustrates workers' translation quality at across time. In this graph, each tick represent a single translation HIT, and depicts the HIT's BLEU score (color) and its size/number of sentences (thickness). The worker who provided the translation is graphed on the y axis, and the submission time is plotted on the x axis. We set the median of all HITs' BLEU scores as the threshold. If the HIT's BLEU score is higher than the median, then the HIT's color is dark. A light colored tick mark means the HIT's BLEU score is lower than the median. In Figure 26, the order of workers on y axis is based on the gold standard ranking of these workers. The top most worker ranked highest and the bottom most worker ranked lowest.

From Figure 26, we see that most workers' performance stays consistent as time passes. This is good, since it may enable us to predict workers' performance based on their early submission so that we can come to an early decision about whether to continue to hire them. If we rank the worker based solely on their first HITs' BLEU score, comparing all sentences in that HIT against the reference translation, then we get a surprisingly strong correlation with the true gold standard ranking of workers based on all of their submitted work. The Spearman Correlation is 0.86 when comparing this first-HIT ranking with gold standard ranking.

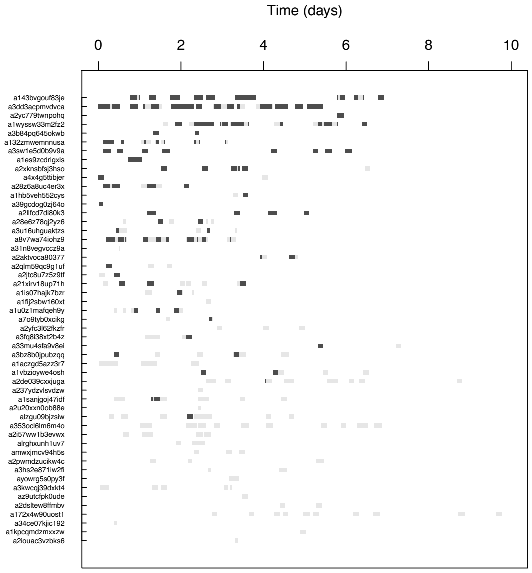

Figure 26: Workers' performance at across time.

Expanding the experiment mentioned above, we rank workers using their first k HITs and calculate Spearman Correlation comparing against the gold standard ranking list (calculated over all HITs, instead of just the first k). The value of k could range from 1 to the max number of HITs submitted by any single worker. Since some workers only translated a small number of HITs, larger values of k will be greater than the total number of HITs that they provided. For those workers who translated less than k HITs in total, we use all of the HITs they submitted to rate them. Since the correlation converges to 1 after k is larger than 10, we plot the graph to show the increase in correlation for k from 1 to 10. Figure 27 shows details. The average number HITs each worker provided is 15.06, so we also plot Figure 28 to show the percentage of HITs we use to rank workers for each value of k.

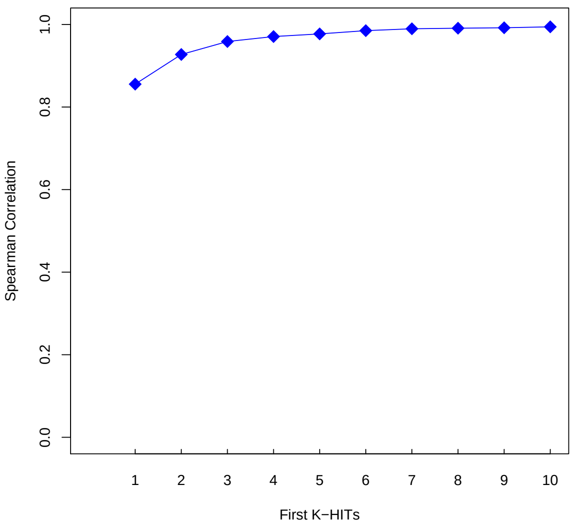

Figure 27: Spearman Correlation with the gold standard as we rank the workers based on their first k HITs

2.9.5 Cost Optimization

There are two approaches that we use to reduce costs. In the first approach, we attempt to quickly rank workers and discard low ranking workers. In the second approach, we reduce the number of translations that we buy for each foreign sentence. After receiving a translation, we decide whether its quality is sufficient or whether we ought to pay for another

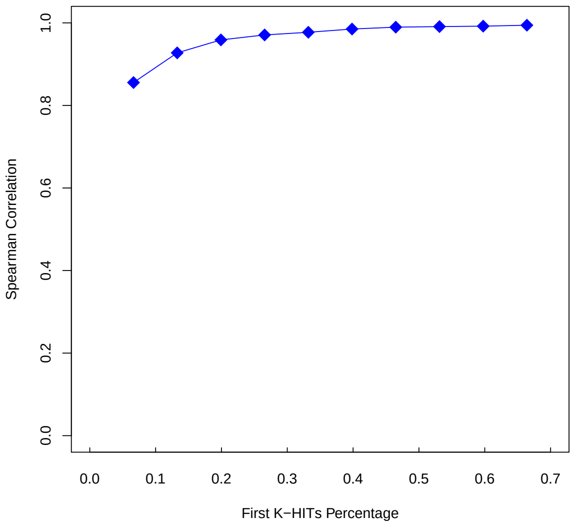

Figure 28: The percentage of worker data used to rank them as k increases

translation (with the hope that the subsequent translation will be better). In both cases we aim to reduce costs, while keeping our overall translation quality high.

For the first approach, as mentioned in section 2.9.4, we show we are able to rank workers accurately using only the translations of their first 4 HITs. The ranked list of workers using 4 HITs is almost identical to the gold standard ranking that uses all HITs. In other words, we can predict workers' performance reasonably well, as long as we have obtained a small number of HITs from them. (Assuming that we have professional translations of their initial HITs, which we can use to calculate their translation quality). Consequently, we can decide whether to continue to hire a worker in a very short time after analyzing the first HIT(s) provided by each worker.

For the second approach, we train a model to decide whether a translation is 'good enough,' in which case we don't need to pay for another redundant translation of the source sentence. To perform this experiment, we divide the data into 3 parts: 10% of the data as a training set, 10% of the data as a validation set and the remaining 80% of the data as a test set. We train the model to score each translation we've got already, to use this score to evaluate whether to get another translation. The challenge is how to set the threshold to separate acceptable translations and unacceptable ones.

In our design, we set the threshold empirically using the validation set after we have trained the model on the disjoint training set. More specifically, during the training process, we get the lower bound and upper bound of scores for translations in the training set. Then we search for the threshold through traversing from the lower bound to the upper bound by a small step size. We use each value in the process as the potential threshold. We score translations of the foreign sentences in the validation set. Since this approach assumes a temporal ordering of the translations, we compute the scores for each translation of a source sentence using the time-ordering of when Turkers submitted them. There are 2 conditions on the halt of this process for each foreign sentence: 1) the predicted BLEU score of some translation (submitted earlier than the last translation) is higher than the threshold or 2) we have scored all 4 translations.

To evaluate the performance of the model running with different thresholds, we first compute an upper bound by selecting the best translation among all 4 candidates for each foreign sentence of the validation set according to our model. We call this set S_{upper}. S_{upper} is the highest BLEU score we can get by choosing translation using the model, since it has access to all of the available translations.

After we have used the validation set to sweep various threshold values, we can pick a suitable value for the threshold by picking the lowest value that is within some delta of S_{upper}, say 90%.

Finally, we retrain our model using the union set of the training set and validation set, use the resulting model on the test set. We evaluate the model's performance by counting the average number of candidate translations that it solicits per source sentence, and by computing the loss in overall BLEU score compared to when it had access to all 4 translations. This evaluation shows how much money our model would save by eliminating unnecessary redundancy in the translation process, and how close it is to the upper bound on translation quality when using all of the translations from the original set.

3 Related Work

In the second approach mentioned above, we train models to separate acceptable and unacceptable translation candidates. Candidates with BLEU scores higher than the threshold are acceptable and vice versa. We search for the threshold on the validating set that is leading to the BLEU within some delta of S_{upper}. The threshold between acceptable and unacceptable translations is fuzzy so there exists some uncertainty in labeling each data sample and we need to sweep various values.

Repeated labeling (Sheng et al., 2008) maybe one way to solve the problem of the uncertainty in labeling and the fuzziness in the selection of the threshold. In the work of (Sheng et al., 2008), first they showed that for single-labeling examples, they showed that the labeling quality (the probability of correct labeling) is critical to the model quality. The model prediction accuracy rises as the labeling quality increases. However, in reality, we cannot always get high-quality labeled data samples with relatively low costs. To keep the model trained on noisy labeled data having a high accuracy in predicting , Sheng et al. (2008) proposed repeated-labeling framework.

In their work, Sheng et al. (2008) assume each data sample's labeling is independent of the specific data point. Each repeated-labeling data sample is assigned an integrated label by majority voting. The experimental results show that repeated labeling can improve the model's predicting accuracy even if labels quality can not be guaranteed and are noisy. As long as the integrated quality (the probability of the integrated labeling being correct) is higher than 0.5, repeated labeling makes benefits for model training. More closer the quality to 0.5, the more benefits obtained in model prediction.

If we relax the condition on the uniform labeler quality and allow lablers to have different quality, a new question arises: should we use the best individual labeler or should we combile the results from multiple labelers? According to the analysis, it depends on how much the best labeler's quality is better the average quality of all labelers.

Majority voting is an useful approach to improve the quality of corpus. However, it omits the uncertain property of data samples' labels and loses the information about uncertainty in labels. To take advantage of the uncertainty, Sheng et al. (2008) represent labeling uncertainty in probability. For each unlabeled data sample x_i, the *multiplied examples* (ME) procedure takes the existing multi-label set $L_i = \{y_{ij}\}$ as input, and for each label value y_{ij} in the multiset, make a replica of x_i which is labeled y_{ij}, and use the probability of that label value appearing in the multi-label set as the weight of that replica . We can use cost-sensitive learning method to train model on the modified data set. Experiment shows that ME strategy is better than than majority voting.

In experiments, for each data set, 30% data samples are held out as the test data, and the rest data is the "pool" from which we acquire unlabeled and labeled samples. When deciding the next data sample to be labeled, they use the generalized round robin strategy: selecting the data sample with the fewest labels. To make the selection more reasonable, they proposed the selective-repeated labeling method. For each data sample, LU is defined as the label uncertainty which measures the labels' diversity on the data sample. Similarly, MU is defined as the model uncertainty which measures the disagreement on models' prediction to the data sample. LMU is defined as the label and model uncertainty which is the geometric average of LU and MU. Instead of assigning the new label to the data sample with fewest labels, they choose the data sample with the highest LMU score and get benefits.

A very important issue in natural language processing is data annotation. Hiring professional annotators is very expensive. As an alternative, collecting several annotation for each single data sample and pick the best label is more economical. In our work, we collected several translations for each source sentence and pick the best translation. Our work shares many goals in common with Passonneau and Carpenter (2013), who created a Bayesian model of annotation, which they applied to the problem of word sense annotation. Rather than hiring professional annotators, which is very expensive, they hire non-expert annotators on Mechanical Turk. They collected 20 to 25 word sense labels for each word. To decide which label to select for each word, and to compute the quality of the annotation, they proposed the probabilistic model using Bayes's rule. They calculated the product of the prior probability (the initial probability of being the observed label) and the conditional probability (the probability of being the observed label given the true label) and pick one label with the highest score. This sort of a probability estimate provides much more information about the corpus quality than previous methods, such as calculating inter-annotation agreement through Coehn's kappa score. Kappa measures the agreement coefficient among annotators in a chance-adjusted fashion. However, the method only reports how often annotators agree, but does not provide information about the quality of the corpus and the individual data sample.

Although Passonneau and Carpenter (2013) collect word sense labels, which are a small, enumerable set, and we collect translation (which could be thought of as a kind of label, albeit a very complex one), there is a strong commonality in the goals of their word and the goals of our work. Specifically, how can we use all the labels collected in order to select of the best label. And how can we rank the annotators themselves. For selecting the best label for word senses, majority voting is a direct and easy way to solve the problem, but the task is more complex for translation.

Passonneau and Carpenter (2013) also proposed an approach to detect and avoid spam workers. They measured the performance of worker by comparing worker's labels to the current majority labels and worker with bad performance would be blocked. However, this approach suffered from 2 shortcomings: (1) Sometimes majority labels may not reflect the ground truth label. (2) They didn't figure out how much data(HITs) is needed to evaluate a worker's performance. Although they could find the spam after the fact, it was a post-hoc analysis, so they had already paid for that worker and wasted the money. We attempt to identify poor workers as quickly as possible, in order to limit the amount of work that we solicit from them.

References

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Rebecca J Passonneau and Bob Carpenter. 2013. The benefits of a model of annotation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 187–195. Citeseer.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM.

Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.