When Transliteration Met Crowdsourcing : An Empirical Study of Transliteration via Crowdsourcing using Efficient, Non-redundant and Fair Quality Control

Mitesh M. Khapra[*], **Ananthakrishnan Ramanathan**[*], **Anoop Kunchukuttan**[†],
Karthik Visweswariah[*], **Pushpak Bhattacharyya**[†]
[*]IBM Research India,
Bangalore, India.
mikhapra@in.ibm.com, anandr42@gmail.com, v-karthik@in.ibm.com

[†] Department of Computer Science and Engineering,
IIT Bombay,
Mumbai, India.
anoopk, pb@cse.iitb.ac.in

Abstract

Sufficient parallel transliteration pairs are needed for training state of the art transliteration engines. Given the cost involved, it is often infeasible to collect such data using experts. Crowdsourcing could be a cheaper alternative, provided that a good quality control (QC) mechanism can be devised for this task. Most QC mechanisms employed in crowdsourcing are aggressive (unfair to workers) and expensive (unfair to requesters). In contrast, we propose a low-cost QC mechanism which is fair to both workers and requesters. At the heart of our approach, lies a rule based Transliteration Equivalence approach which takes as input a list of vowels in the two languages and a mapping of the consonants in the two languages. We empirically show that our approach outperforms other popular QC mechanisms (*viz.*, consensus and sampling) on two vital parameters : (i) fairness to requesters (lower cost per correct transliteration) and (ii) fairness to workers (lower rate of rejecting correct answers). Further, as an extrinsic evaluation we use the standard NEWS 2010 test set and show that such quality controlled crowdsourced data compares well to expert data when used for training a transliteration engine.

Keywords: crowdsourcing, transliteration, quality control

1. Introduction

Transliteration plays an important role in CLIR and MT by handling OOV terms in queries and documents (Choi et al., 2011). Several algorithms have been proposed to address this problem. However, most state-of the art approaches for transliteration (Jiampojamarn et al., 2009; Oh et al., 2009; Jansche and Sproat, 2009) are data driven and need significant parallel words data which is not readily available for many languages. Employing experts to create such data is both time consuming and expensive, and hence not desirable. While one could argue that such data can be mined from online resources such as parallel Wikipedia articles, it should be noted that for many language pairs very few parallel resources are available to yield any meaningful amount of data (Khapra et al., 2010). This situation leads to the following question: *Can such data be collected at low cost while not compromising much on quality?*

The above question led us to crowdsourcing which is increasingly being used for creating speech and language data for several NLP tasks (Callison-Burch and Dredze, 2010), such as, paraphrasing (Denkowski et al., 2010), annotating named entities (Yetisgen-Yildiz et al., 2010), transcription (Evanini et al., 2010), Subjectivity WSD (Akkaya et al., 2010), collecting translations (Ambati and Vogel, 2010), *etc.* The success of crowdsourcing depends on having a good quality control (QC) mechanism which can automatically reject incorrect answers and be fair to both workers and requesters. Two popular QC mechanisms are consensus based QC (Ipeirotis et al., 2010) and sampling based QC (Eickhoff and de Vries, 2011; Le et al., 2010; Oleson et al., 2011). In consensus based QC, the same task is given to multiple workers and the correct answer is selected based on majority voting. In sampling based QC, a small sample of the worker's answers are evaluated against a gold standard and his entire work is accepted/rejected based on the performance on this sample. These techniques are often aggressive and unfair to workers as they end up rejecting a lot of good work submitted by them. Further, the inherent redundancy involved in these mechanisms adds to the cost which makes them unfair to requesters also. Hence, there is a need for a task-specific, fair, efficient and non-redundant QC mechanism.

In this paper, we focus on the task of collecting parallel transliteration pairs using crowdsourcing and propose a low-cost QC mechanism for this task. Our approach relies on a list of vowels in the two languages and a mapping between the consonants of the two languages. For example, given the consonant 'k' in English we are interested in the set of Hindi graphemes to which it can be transliterated. An expert bilingual (English and Hindi) speaker just took 30 minutes to prepare a full list of such consonant mappings for English and Hindi. Given such a mapping, we propose a rule based approach to identify correct transliterations submitted by the crowd. Since our algorithm requires very limited language-specific information (just a list of vowels, and a mapping between consonants) it is easy to adapt to new language pairs as we show in section 8.

We empirically compare our approach to two popular QC mechanisms described earlier and show that our approach performs better in terms of both, fairness to workers and fairness to requesters. It is fairer to workers than the other two approaches as it has a lower rate of rejecting correct answers (29% lower than the best competitor). Moreover, it is also fairer to requesters as it has a lower cost per correct transliteration (13% lower than the best competitor). This makes it more suitable for large scale data collection. Finally, a good extrinsic test of the quality of crowdsourced data would be to compare the performance of a state of the art engine trained on such data with the performance of the same engine trained on data created by experts. Our experiments involving the standard NEWS 2010 Transliteration test set suggest that the performance of an engine trained on crowdsourced data comes close to that of an engine trained on expert data. The main contributions of our work can be summarized as follows:

1. We empirically evaluate the quality of transliteration data collected using crowdsourcing.
2. We propose a low cost QC technique for this task which is fair to both workers and requesters.
3. We do an error analysis and give useful insights into the data collected by the crowd.
4. We perform extrinsic evaluations to verify the usability of crowdsourced data for building downstream transliteration engines.
5. We provide some extrinsic evaluation to prove that the proposed QC mechanism could work for language pairs other than Hindi-English also.

The remainder of this paper is organized as follows. In Section 2. we discuss related work on crowdsourcing. In Section 3. we outline our rule-based approach for filtering incorrect transliteration pairs. In Section 4. we give details of the experimental setup used for collecting data using crowdsourcing. In Section 5. we briefly discuss how different QC mechanisms were employed for our task. In Section 6. we empirically compare different QC mechanisms based on fairness to workers and fairness to requesters. Next, in Section 7. we evaluate the usefulness of crowdsourced data in training a transliteration engine and evaluate the performance of our approach on other languages in Section 8. Finally, we present concluding remarks in Section 9.

2. Related Work

In recent years, crowdsourcing has become increasingly popular. For example, (Callison-Burch and Dredze, 2010) and (Snow et al., 2008) report the use of crowdsourcing for collecting data and manual evaluations for various NLP and IR related tasks. In general, these studies show that non-expert annotations show a high degree of agreement with expert annotations and systems trained with non-expert annotations perform comparably to systems trained with expert annotations. While crowdsourcing has been tried for a variety of tasks, to the best of our knowledge, ours is the first attempt to use crowdsourcing for collecting transliterations.

Ensuring quality of crowdsourced data in the presence of factors such as malicious workers, inexperienced workers, subjectivity of the task, *etc.* is a challenging task. Prefiltering of workers on the basis of worker accuracy or geographical locations and other such criteria is one of the simplest methods adopted to assure worker quality (Eickhoff and de Vries, 2011). However, worker accuracies can be artificially boosted up and there is no incentive for the worker to be honest once the pre-filtering has been passed successfully. Asking multiple workers to do a task and deciding the correct result based on consensus or majority voting (Ipeirotis et al., 2010) is a popular approach, though this approach can often be aggressive and has inherent redundancy. Use of gold standard data to evaluate a sample of the work is also a popular method to identify spam workers (Eickhoff and de Vries, 2011; Le et al., 2010; Oleson et al., 2011). The use of redundancy (for consensus) and gold judgments (for sampling) add to the cost of the task which is unfair to requesters. Sometimes, an additional validation/ranking stage (Zaidan and Callison-Burch, 2011) is also used but there is no guarantee that these validations are themselves honest and correct (and such validations add to the cost). In contrast, we propose an efficient QC mechanism for the task of collecting transliteration pairs which has no additional cost and is fair to workers.

3. A Rule Based Transliteration Equivalence algorithm

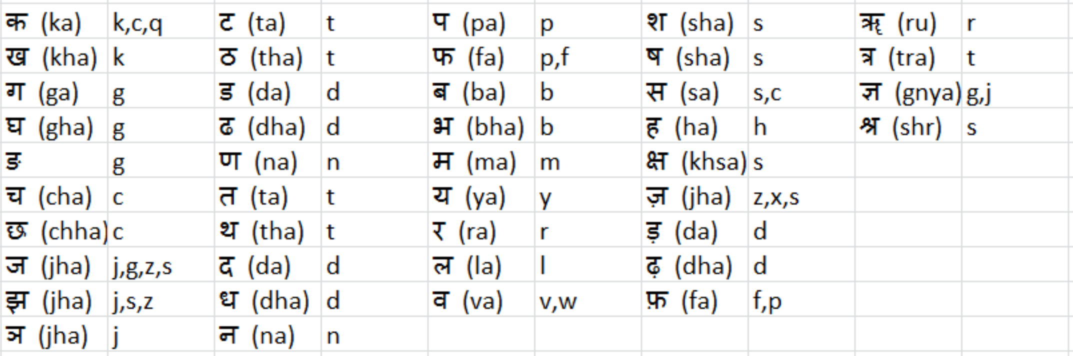
Let $\hat{v}$ be the transliteration submitted by a worker for an input word $\hat{u}$. A good QC mechanism should be able to reject this task if $\hat{v}$ is not a correct transliteration of $\hat{u}$ and accept it otherwise. Further, since we do not have any training data to begin with, such an algorithm should be minimally supervised. We propose one such minimally supervised rule based algorithm which solves this problem of recognizing transliteration equivalence.

The proposed algorithm only needs two pieces of information: (i) a list of vowels in the two languages (which we pick up from the Wikipedia page for the script[1]) and (ii) a list of consonant mappings between the two languages. As mentioned, such a list of consonant mappings can be created in a very short time (30 minutes) by a bilingual speaker. For example, Figure 1 shows the complete list of English Hindi consonant mappings created by a native bilingual speaker. The first entry in this table suggests that according to a native bilingual speaker's intuition, when a Hindi word containing the consonant ka (which represents the sound ka) is translated to English then this consonant will get transliterated as k (as in **k**ing) or c (as in **c**andid) or q (as in **q**ueen). Given such a list of vowels and consonant mappings, we use a task-aware matching step that uses these mappings to filter out incorrect transliteration pairs from the collected data. We describe this algorithm in the next subsection.

3.1. Identify Correct Transliterations (Match)

Given a word pair $\hat{u} \leftrightarrow \hat{v}$ such that $\hat{u} \in language\ L_1$ and $\hat{v} \in language\ L_2$, a list of vowels in L_1 and L_2 and a set

[1] for example, Hindi vowels are from http://en.wikipedia.org/wiki/Devanagari



क (ka)	k,c,q	ट (ta)	t	प (pa)	p	श (sha)	s	ऋ (ru)	r
ख (kha)	k	ठ (tha)	t	फ (fa)	p,f	ष (sha)	s	त्र (tra)	t
ग (ga)	g	ड (da)	d	ब (ba)	b	स (sa)	s,c	ज्ञ (gnya)	g,j
घ (gha)	g	ढ (dha)	d	भ (bha)	b	ह (ha)	h	श्र (shr)	s
ङ	g	ण (na)	n	म (ma)	m	क्ष (khsa)	s		
च (cha)	c	त (ta)	t	य (ya)	y	ज़ (jha)	z,x,s		
छ (chha)	c	थ (tha)	t	र (ra)	r	ड़ (da)	d		
ज (jha)	j,g,z,s	द (da)	d	ल (la)	l	ढ़ (dha)	d		
झ (jha)	j,s,z	ध (dha)	d	व (va)	v,w	फ़ (fa)	f,p		
ञ (jha)	j	न (na)	n						

Figure 1: A mapping between the consonants of Hindi and English

of mappings M between the consonants in L_1 and L_2, the matching process proceeds as follows:

[1.] Check boundary vowels: If $\hat{u}$ begins/ends with a vowel and $\hat{v}$ does not begin/end with a vowel, or vice-versa, return Failure ('No match'). Note that, for this check we allow for a stop list of vowels (*e.g.*, we do not insist that an "e" at the end of a word in English match with a vowel in the other language).

[2.] Remove vowels: Remove all vowels from both $\hat{u}$ and $\hat{v}$ to obtain the letter sequences u_1^n and v_1^m respectively.

[3.] Search:

- Initialize the hypothesis space: For i in 1 to n, $S_i = \emptyset$
- For-each letter u_i in u_1^n
    - For-each rule $u_i \leftrightarrow v_i$ in R, append v_i to each string in S_{i-1} and add it to S_i
- If v_1^m in S_n, return Success ('Match')

[4.] Handle letter repetitions: Sequences of identical adjacent letters are compressed into single occurrences in both v_1^m and in all strings in S_n. Now if $v_1^m \in S_n$, return Success ('Match').

Example: Let "luck" be the source word and "लक (lak)" be its transliteration as submitted by a worker. Further, let's assume that the following mapping rules are available : l $\leftrightarrow$ ल (la), c $\leftrightarrow$ क (ka), c $\leftrightarrow$ स (sa), k $\leftrightarrow$ क (ka). We proceed as follows:

- Check boundary vowels: Both words begin and end with consonants, so the match proceeds to the next step (if the Hindi word had been लकी ($lakii$), this step would have failed).
- Remove vowels: We now have lck $\leftrightarrow$ लक (lk).
- Search: The hypothesis space at the final ply is $S_4 =$ लसक (lsk), लकक (lkk)
- Handle letter repetitions: Now, $S_4 =$ लसक (lsk), लकक (lkk), लक (lk).

Since लक (lk) is in S_4, the match succeeds.

4. Data collection

We now discuss the experimental setup that we used for collecting transliterations via crowdsourcing. Following Amazon Mechanical Turk (AMT) terminology, in the remainder of this paper we use the terms *answer* and *transliteration* interchangeably.

Platform: We used AMT since it is one of the most preferred crowdsourcing platforms.

Data source: To ensure easy replicability we used the standard NEWS 2010 dataset (Li et al., 2010). Specifically, we asked the crowd to provide English transliterations for the Hindi source words in the training portion of this dataset. The reason we chose Hindi → English as the direction of transliteration is that we felt it would be easier for workers to type in English than in Devanagari.

HITs: We floated the tasks in batches of 10. In AMT parlance this means that each HIT (Human Intelligence Tasks) contained 10 tasks where each task comprised of providing transliterations for one Hindi word. We floated 1500 such HITs and collected transliterations for 15K words.

Price: The workers were paid 0.002 dollars per word, whereas the cost of getting the same work done by an expert is ten times greater at 0.02 dollars per word.

Instructions: To avoid confusion, the workers were given examples of Hindi words and their English transliterations. In order to get natural transliterations, we refrained from providing any character mappings (such as itrans) to the workers and simply asked then to transliterate the source words using their knowledge of the two languages.

Worker quality: Only those workers who had a previous record of more than 5000 approved HITs with an approval rate of >95% were allowed to preview/accept our HITs.

Time limits: Once a worker accepts a HIT he/she was given 24 hours to complete it (which is much more than sufficient since each HIT consists of 10 words only). Workers typically took less than 3 minutes to complete one

HIT.

Batches: With a hope of getting some variety in the workers, we floated the 1500 HITs in 3 batches with a gap of 3 days between each batch.

Redundancy: Each HIT was assigned only to one worker (*i.e.*, there was no redundancy). However, later to compare different QC methods, we collected 100 HITs with a redundancy of 3, *i.e.*, the same HIT was assigned to 3 workers.

5. Different QC mechanisms

As mentioned earlier, we collected transliterations for 15000 words by floating 1500 HITs on AMT. We ran simulations on this data to compare the following QC mechanisms :

Match: Here, we employ the matching algorithm discussed in Section 3. to determine whether an answer submitted by a worker is correct or not. Essentially, only those transliterations which are marked as correct by our algorithm are accepted (we reiterate that we ran an offline simulation on the data collected using crowdsourcing).

Sampling: Here, we sampled n (<10) answers (*i.e.*, transliterations) from each HIT submitted by a worker. We then manually evaluated these n answers and if c (<=n) out of these answers were correct then we accepted the HIT, else we rejected it. In practice, this manual verification can be minimized by first asking an expert to give all possible transliterations for a small set of m words (say, m=100). Each HIT can then contain n out of these m words and the answers provided by the crowd can be directly compared with the gold transliterations collected earlier. We used n=3 because typically an odd number (>1) is preferred and $n > 3$ would have been highly redundant.

Consensus: Here, the same HIT is assigned to n workers. For a given word $\hat{u}$, if k out of n workers provide $\hat{v}$ as the transliteration then $\hat{v}$ is accepted as a correct transliteration else it is considered incorrect. For the purpose of evaluation, only 100 out of the 1500 HITs were assigned to 3 workers each (the remaining HITs were assigned to only one worker).

Baseline: Here, all the answers submitted by the crowd are accepted as correct answers.

6. Intrinsic evaluation

Before comparing the above methods, we ask the following question: *In a crowdsourcing environment what does a worker/requester really wish for?* If the price and nature of the task are fixed, then an honest worker wishes that *none of his/her correct answers are rejected.* Similarly, a requester wishes that *the cost he/she pays per correct answer should be minimum.* The cost per correct answer in turn depends on the accuracy of the QC mechanism and the redundancy involved. Thus, to be fair to a requester the QC mechanism should have minimum redundancy and a high precision (*i.e.* a very high fraction of the answers accepted by it should be actually correct). In this context, we now compare different QC methods based on two parameters : (i) fairness to requesters and (ii) fairness to workers.

QC		**P**	**C**
Baseline		71.7	1.4
Match		82.3	**1.22**
Sampling	n=3, c=3	83.3	1.71
	n=3, c=2	72.3	1.98
	n=3, c=1	57.3	2.49
Consensus	n=3, k=3	97.2	3.09
	n=3, k=2	88.3	3.40
	n=3, k≥2	91.9	3.26

Table 1: A comparison of the precision (**P**) and cost (**C**) per correct transliteration of different QC mechanisms.

Fairness to requesters: This parameter relates to the cost per unit word and depends on the precision of the QC mechanism and the redundancy involved. First, to measure precision, for each QC mechanism we randomly collected 1000 words that were accepted by that QC mechanism. We manually evaluated these words to find out the percentage of words that were actually correct (a high number here is desired). Next, we estimate the cost per correct word. Note that, if we assume that the cost of collecting 1 transliteration is 1 unit then the effective cost of collecting one transliteration for the consensus based method is 3 units (since we need to collect 3 transliterations per word). Similarly the effective cost of collecting one transliteration for the sampling based method is 1.4 (10/7 since each HIT contains 3 redundant words for which the answers are already known). On the other hand, since our approach has no redundancy the effective cost of collecting one transliteration is just 1 unit. Further, if a QC mechanism accepts y answers as correct and only x of them are actually correct then the effective cost per correct answer (C) is

$$C = \textit{Effective cost per answer} \cdot \frac{\textit{Accepted answers}}{\textit{Correct Answers}}$$
$$= \frac{\textit{Effective cost per answer}}{\textit{Precision}}$$

The above formula thus accounts for both: (i) overhead due to redundancy and (ii) overhead due to incorrect answers. Table 1 compares the precision and cost per correct transliteration of different QC methods. For the sampling based method, we report numbers with n=3 and c=1,2,3. Similarly for the consensus based method we report numbers with n=3 and k=2,3. We note that the precision of the consensus based method is very high but its cost per correct transliteration is also high due to the high redundancy involved. Similarly, the cost per correct transliteration for the sampling based method is also high. Our approach has the least cost per correct transliteration (even though its precision is lower than consensus based QC) and is thus most fair to requesters.

Fairness to workers: To quantify this parameter, we took

QC mechanism	% of rejected answers that were actually correct
Match	19.0 %
Sampling(n=3, c=3)	64.0%
Consensus(n=3, k=3)	48.5%

Table 2: A comparison of different QC mechanisms based on fairness to workers.

100 answers each which were rejected by the three strategies and compared the fraction of these answers which were actually correct (**a low number here is desired**). We report numbers corresponding to the lowest cost per correct transliteration. As is evident from Table 2, our algorithm clearly outperforms the other two approaches on this parameter. The poor numbers for sampling based QC and consensus based QC can be explained as follows. In the sampling method, a HIT is rejected if the number of n sampled answers that are correct is less than k. This is unfair to the workers because some of the remaining $10 - n$ answers might be correct but they do not receive any payment for these correct answers. Similarly, in the redundancy based method, it is possible that one worker has given a correct transliteration which does not agree with the other 2 workers. Given that there are multiple ways of transliterating a word this situation could be quite common.
Note that, while the baseline (accepting all answers) is obviously most fair to workers, it may not be acceptable for the task since it compromises much more on precision, and is less fair to requesters as seen in Table 1.

We would like to briefly mention how our approach can be used in practice, to ensure that a requester pays only for correct answers. For this, we propose the following strategy which harnesses the ‘bonus payment’ feature provided by AMT (i) accept a HIT if all the 10 answers are marked as correct by the matching algorithm (ii) reject a HIT if k (>0) out of the 10 answers are marked as incorrect by the matching algorithm. Then pay the worker for his $10-k$ correct answers on a pro-rata basis using the ‘bonus payment’ feature provided by AMT. Alternatively, each HIT could contain only one word in which case it would be simple to just accept or reject the HIT based on the verdict of the matching algorithm. However, this is not advisable because workers might find it irritating to work on HITS containing only one word.

6.1. Error analysis

We split the input data into two parts : Indian origin words (*e.g.*, Shankar, Ram, *etc.*) and non-Indian origin words (*e.g.*, Stephen, George, *etc.*). We found that the ratio of Indian to non-Indian origin words in the data was roughly 3:2. We then randomly selected 1K words from the 15K words submitted by the crowd and manually evaluated the accuracy on these two sets. As shown in Table 3, the crowd did a much better job of transliterating Indian origin words (precision of 76%) than non-Indian origin words (precision of 66%). In Table 4, we list down some non-Indian origin words which were transliterated incorrectly by the crowd. Note that the incorrect transliterations are phonologically correct even though they are orthographically incorrect. Further, in most cases, these errors were mainly in transliterating vowels. This is mainly because, in English, vowels have an ambiguous phoneme to grapheme mapping. For example, the grapheme ’o’ represents different sounds in *orange* and *bison*. Hence, non-native speakers may use these vowels interchangeably to produce an incorrect transliteration.

Words	**Precision**
Indian origin	76.0
non-Indian origin	66.2
All	71.7

Table 3: Accuracy of crowd on words of different origins.

Crowd(incorrect)	**Expert(correct)**
pablic	public
skool	school
universitee	university
zyoorik	zurich
landan	london

Table 4: Errors made by the crowd on some non-Indian origin words.

7. Extrinsic Evaluation

As an extrinsic evaluation of the data collected using crowdsourcing, we measure its adequacy in training a transliteration engine. Specifically, we trained a state of the art transliteration engine using the 12K transliterations which were accepted by our QC mechanism (match). We train the same engine using manual transliterations for the same 12K Hindi words as provided in the NEWS 2010 training set (Li et al., 2010). As a baseline, we also train the same engine using 12K randomly selected transliteration pairs from the crowdsourced data (*i.e.*, no QC is used). For all experiments we use the open-source implementation of DirectL (Jiampojamarn et al., 2010) which is the current state of the art algorithm for transliteration. The performance was evaluated on the NEWS 2010 test set (Li et al., 2010). As shown in Table 5, the performance of DirectL when trained on quality controlled crowdsourced data is close to that of an engine trained on data created by experts even though the crowdsourced data was collected at 1/10th of the cost. Note that due to budget constraints we could not collect 12K words using the sampling and consensus based method. Hence, we could not do any extrinsic evaluation for the data collected using these methods (the intrinsic evaluation reported earlier was done using a sample of 1K words collected using these two methods).

8. Extending to other languages

Though all our experiments were done on Hindi-English, we provide some extrinsic proof that the proposed QC method can work for language pairs other than Hindi-English also. For this, we use a standard dataset which was released for the task of mining transliteration pairs

Data	top-1 Accuracy
Crowd (12K, no QC)	34.8%
Crowd (12K, QC with our approach)	37.1%
Expert (12K)	39.6%

Table 5: A comparison of the accuracy of a transliteration engine trained using different data sources.

Method	hi-en	ar-en	ru-en	ta-en
Our QC approach	95.2	89.6	83.8	91.5
NEWS 2010	94.4	91.5	87.5	91.4
ACL2011	92.2	87.4	76.0	90.1
ACL2012	95.7	92.4	79.4	93.2

Table 6: A comparison of our approach with other transliteration equivalence approaches.

from Wikipedia titles (Kumaran et al., 2010). This task deals with filtering correct transliterations from a mixture of correct and incorrect transliterations and is thus very similar to our task. We compare the performance of our system with the best systems (supervised) reported in NEWS 2010 (Kumaran et al., 2010) and two recent state of the art unsupervised systems (Sajjad et al., 2011; Sajjad et al., 2012) on four language pairs (Hindi-English, Arabic-English, Russian-English and Tamil-English). Table 6 shows that our approach compares very well to these approaches and thus has the potential of serving as a good QC mechanism for these language pairs also.

9. Conclusions

In this paper, we reported our experiments on collecting transliteration pairs using crowdsourcing. We proposed a low cost quality control (QC) mechanism for this task and showed that our approach outperforms two popular QC mechanisms in terms of fairness to workers and fairness to requesters. We also put the crowdsourced data to test by using it to train a transliteration engine, and found that the transliteration performance was close to the performance when trained on data created by experts. This establishes the practical utility of crowdsourcing for collecting transliteration data. We also showed that our approach can work for other language pairs by evaluating it on public data for a closely related task. Finally, our error analysis showed that the crowd is much better at transliterating Indian origin words than non-Indian origin words. This suggests that its best to use crowdsourcing for collecting transliterations of native origin words. This could be a useful insight while employing crowdsourcing to collect transliterations for other language pairs.

10. References

Akkaya, C., Conrad, A., Wiebe, J., and Mihalcea, R. (2010). Amazon mechanical turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 195–203, Los Angeles, June. Association for Computational Linguistics.

Ambati, V. and Vogel, S. (2010). Can crowds build parallel corpora for machine translation systems? In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Callison-Burch, C. and Dredze, M. (2010). Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, Los Angeles, June. Association for Computational Linguistics.

Choi, K.-S., Isahara, H., and Oh, J.-H. (2011). A comparison of different machine transliteration models. *CoRR*, abs/1110.1391.

Denkowski, M., Al-Haj, H., and Lavie, A. (2010). Turker-assisted paraphrasing for english-arabic machine translation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 66–70, Los Angeles, June. Association for Computational Linguistics.

Eickhoff, C. and de Vries, A. (2011). How Crowdsourcable is Your Task? In Lease, M., Carvalho, V., and Yilmaz, E., editors, *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 11–14, Hong Kong, China, February. Received Most Innovative Paper Award.

Evanini, K., Higgins, D., and Zechner, K. (2010). Using amazon mechanical turk for transcription of non-native speech. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 53–56, Los Angeles, June. Association for Computational Linguistics.

Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, New York, NY, USA. ACM.

Jansche, M. and Sproat, R. (2009). Named entity transcription with pair n-gram models. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 32–35, Suntec, Singapore, August.

Jiampojamarn, S., Bhargava, A., Dou, Q., Dwyer, K., and Kondrak, G. (2009). Directl: a language independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 28–31, Suntec, Singapore, August.

Jiampojamarn, S., Dwyer, K., Bergsma, S., Bhargava, A., Dou, Q., Kim, M.-Y., and Kondrak, G. (2010). Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, pages 39–47, Uppsala, Sweden, July. Association for Computational Linguistics.

Khapra, M. M., Kumaran, A., and Bhattacharyya, P. (2010). Everybody loves a rich cousin: an empirical study of transliteration through bridge languages. In *Hu-*

man Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT '10, pages 420–428, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kumaran, A., M. Khapra, M., and Li, H. (2010). Report of news 2010 transliteration mining shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 21–28, Uppsala, Sweden, July. Association for Computational Linguistics.

Le, J., Edmonds, A., Hester, V., and Biewald, L. (2010). Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 workshop on crowdsourcing for search evaluation*, pages 21–26.

Li, H., Kumaran, A., Zhang, M., and Pervouchine, V. (2010). Report of news 2010 transliteration generation shared task. In *Proceedings of the 2010 Named Entities Workshop*, pages 1–11, Uppsala, Sweden, July. Association for Computational Linguistics.

Oh, J.-H., Uchimoto, K., and Torisawa, K. (2009). Machine transliteration using target-language grapheme and phoneme: Multi-engine transliteration approach. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 36–39, Suntec, Singapore, August.

Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., and Biewald, L. (2011). Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Proceedings of HComp*.

Sajjad, H., Fraser, A., and Schmid, H. (2011). An algorithm for unsupervised transliteration mining with an application to word alignment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 430–439, Portland, Oregon, USA, June. Association for Computational Linguistics.

Sajjad, H., Fraser, A., and Schmid, H. (2012). A statistical model for unsupervised and semi-supervised transliteration mining. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 469–477, Jeju Island, Korea, July. Association for Computational Linguistics.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. (2008). Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yetisgen-Yildiz, M., Solti, I., Xia, F., and Halgrim, S. (2010). Preliminary experiments with amazon's mechanical turk for annotating medical named entities. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 180–183, Los Angeles, June. Association for Computational Linguistics.

Zaidan, O. F. and Callison-Burch, C. (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, Portland, Oregon, USA, June. Association for Computational Linguistics.