Feasibility of Human-in-the-loop Minimum Error Rate Training

Omar F. Zaidan and **Chris Callison-Burch**
Dept. of Computer Science, Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,ccb}@cs.jhu.edu

Abstract

Minimum error rate training (MERT) involves choosing parameter values for a machine translation (MT) system that maximize performance on a tuning set as measured by an automatic evaluation metric, such as BLEU. The method is best when the system will eventually be evaluated using the same metric, but in reality, most MT evaluations have a human-based component. Although performing MERT with a human-based metric seems like a daunting task, we describe a new metric, RYPT, which takes human judgments into account, but only requires human input to build a database that can be reused over and over again, hence eliminating the need for human input at tuning time. In this investigative study, we analyze the diversity (or lack thereof) of the candidates produced during MERT, we describe how this redundancy can be used to our advantage, and show that RYPT is a better predictor of translation quality than BLEU.

1 Introduction

Many state-of-the-art machine translation (MT) systems over the past few years (Och and Ney, 2002; Koehn et al., 2003; Chiang, 2007; Koehn et al., 2007; Li et al., 2009) rely on several models to evaluate the "goodness" of a given candidate translation in the target language. The MT system proceeds by searching for the highest-scoring candidate translation, as scored by the different model components, and returns that candidate as the hypothesis translation. Each of these models need not be a probabilistic model, and instead corresponds to a feature that is a function of a (candidate translation,foreign sentence) pair.

Treated as a log-linear model, we need to assign a weight for each of the features. Och (2003) shows that setting those weights should take into account the evaluation metric by which the MT system will eventually be judged. This is achieved by choosing the weights so as to maximize the performance of the MT system on a development set, as measured by that evaluation metric. The other insight of Och's work is that there exists an efficient algorithm to find such weights. This process has come to be known as the MERT phase (for **M**inimum **E**rror **R**ate **T**raining) in training pipelines of MT systems.

A problem arises if the performance of the system is not judged by an automatic evaluation metric such as BLEU or TER, but instead through an evaluation process involving a human. The GALE evaluation, for instance, judges the quality of systems as measured by human-targeted TER (HTER), which computes the edit distance between the system's output and a version of the output post-edited by a human. The IWSLT and WMT workshops also have a manual evaluation component, as does the NIST Evaluation, in the form of adequacy and fluency (LDC, 2005).

In theory, one could imagine trying to optimize a metric like HTER during the MERT phase, but that would require the availability of an HTER automatic scorer, which, by definition, does not exist. If done manually, the scoring of thousands of candidates produced during MERT would literally take weeks, and cost a large sum of money. For these reasons, researchers resort to optimizing an automatic metric (almost always BLEU) as a proxy for human judgment.

As daunting as such a task seems for any human-based metric, we describe a new metric, RYPT, that takes human judgment into accout when scoring candidates, but takes advantage of the redundancy in the candidates produced during MERT. In this investigative study, we describe how this redundancy can be used to our advantage to eliminate the need to involve a human at any

time except when building a database of reusable judgments, and furthermore show that RYPT is a better predictor of translation quality than BLEU, making it an excellent candidate for MERT tuning.

The paper is organized as follows. We start by describing the core idea of MERT before introducing our new metric, RYPT, and describing the data collection effort we undertook to collect the needed human judgments. We analyze a MERT run optimizing BLEU to quantify the level of redundancy in the candidate set, and also provide an extensive analysis of the collected judgments, before describing a set of experiments showing RYPT is a better predictor of translation quality than BLEU. Following a discussion of our findings, we briefly review related work, before pointing out future directions and summarizing.

2 Och's Line Search Method

A common approach to translating a source sentence f in a foreign language is to select the candidate translation e that maximizes the posterior probability:

$$Pr(e \mid f) \stackrel{\text{def}}{=} \frac{\exp(s_\Lambda(e,f))}{\sum_{e'} \exp(s_\Lambda(e',f))}.$$

This defines $Pr(e \mid f)$ using a *log-linear model* that associates a sentence pair (e, f) with a feature vector $\Phi(e,f) = \{\phi_1(e,f), ..., \phi_M(e,f)\}$, and assigns a score

$$s_\Lambda(e,f) \stackrel{\text{def}}{=} \Lambda \cdot \Phi(e,f) = \sum_{m=1}^{M} \lambda_m \phi_m(e,f)$$

for that sentence pair, with the feature weights $\Lambda = \{\lambda_1, ..., \lambda_M\}$ being the parameters of the model. Therefore, the system selects the translation $\hat{e}$:

$$\hat{e} = \operatorname*{argmax}_e Pr(e \mid f) = \operatorname*{argmax}_e s_\Lambda(e,f). \quad (1)$$

Och (2003) provides evidence that Λ should be chosen by optimizing an objective function basd on the evaluation metric of interest, rather than likelihood. Since the error surface is not smooth, and a grid search is too expensive, Och suggests an alternative, efficient, line optimization approach.

Assume we are performing a line optimization along the d^{th} dimension. Consider a foreign sentence f, and let the candidate set for f be $\{e_1, ..., e_K\}$. Recall from (1) that the 1-best candidate at a given Λ is the one with maximum $\sum_{m=1}^{M} \lambda_m \phi_m(e_k, f)$. We can rewrite the sum as $\lambda_d \phi_d(e_k, f) + \sum_{m \neq d} \lambda_m \phi_m(e_k, f)$. The second term is constant with respect to λ_d, and so is $\phi_d(e_k, f)$. Renaming those two quantities $\texttt{offest}_\Lambda(e_k)$ and $\texttt{slope}(e_k)$, we get

$$s_\Lambda(e_k, f) = \texttt{slope}(e_k)\lambda_d + \texttt{offset}_\Lambda(e_k).$$

Therefore, if we plot the score for a candidate translation vs. λ_d, that candidate will be represented by a line. If we plot the lines for all candidates (Figure 1), then the upper envelope of these lines indicates the best candidate at any value for λ_d.

Therefore, the objective function is piece-wise linear across any of the M dimensions[1], meaning we only need to evaluate it at the "critical" points corresponding to line intersection points. Furthermore, we only need to calculate the sufficient statistics once, at the smallest critical point, and then simply adjust the sufficient statistics to reflect changes in the set of 1-best candidates.

2.1 The BLEU Metric

The metric most often used with MERT is BLEU (Papineni et al., 2002), where the score of a candidate c against a reference translation r is:

$$BLEU = BP(len(c), len(r)) \cdot \exp(\sum_{n=1}^{4} \frac{1}{4} \log p_n),$$

where p_n is the n-gram precision[2] and BP is a brevity penalty meant to penalize short outputs, to discourage improving precision at the expense of recall.

There are several compelling reasons to optimize to BLEU. It is the most widely reported metric in MT research, and has been shown to correlate well with human judgment (Papineni et al., 2002; Coughlin, 2003). But BLEU is also particularly suitable for MERT, because it can be computed quite efficiently, and its sufficient statistics are decomposable, as required by MERT.[3,4]

[1] Or, in fact, along any linear combination of the M dimensions.

[2] *Modifed* precision, to be precise, based on clipped n-gram counts.

[3] Note that for the sufficient statistics to be decomposable, the metric itself need not be – this is in fact the case with BLEU.

[4] Strictly speaking, the sufficient statistics need not be de-

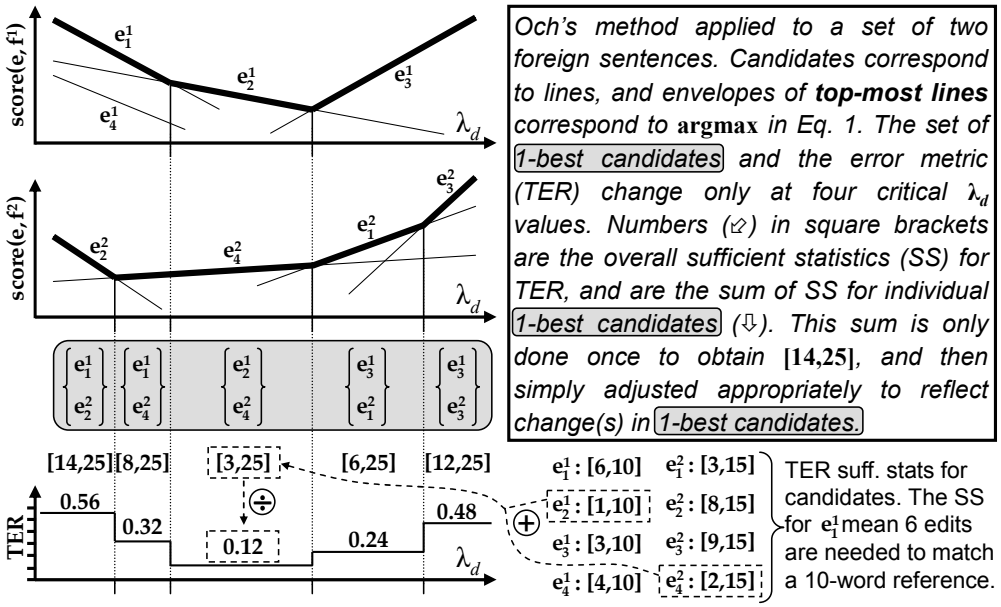

Figure 1: Och's method applied to a set of two foreign sentences. This figure is essentially a visualization of equation (1). We show here sufficient statistics for TER for simplicity, since there are only 2 of them, but the metric optimized in MERT is usually BLEU.

In spite of these advantages, recent work has pointed out a number of problematic aspects of BLEU that should cause one to pause and reconsider the reliance on it. Chiang et al. (2008) investigate several weaknesses in BLEU and show there are realistic scenraios where the BLEU score should not be trusted, and in fact behaves in a counter-intuitive manner. Furthermore, Callison-Burch et al. (2006) point out that it is not always appropriate to use BLEU to compare systems to each other. In particular, the quality of rule-based systems is usually underestimated by BLEU.

All this raises doubts regarding BLEU's adequacy as a proxy for human judgment, which is a particularly important issue in the context of setting parameters during the MERT phase. But what is the alternative?

2.2 (Non-)Applicability of Och's Method to Human Metrics

In principle, MERT is applicable to any evaluation metric, including HTER, as long as its sufficient statistics are decomposable.[4] In practice, of course, the method requires the evaluation of thousands of candidate translations. Whereas this is not a problem with a metric like BLEU, for which automatic (and fast) scorers are available, such an evaluation with a human metric would require a large amount of effort and money, meaning that a single MERT run would take weeks to complete, and would cost thousands of dollars. Assume a single candidate string takes 10 seconds to post-edit, at a cost of $0.10. Even with such an (overly) optimistic estimate, scoring 100 candidates for each of 1000 sentences would take 35 8-hour work days and cost $10,000. The cost would further grow linearly with the number of MERT iterations and the n-best list size. On the other hand, optimizing for BLEU takes on the order of minutes per iteration, and costs nothing.

[4] composable in MERT, as they can be recalculated at each critical point. However, this would slow down the optimization process quite a bit, since one cannot traverse the dimension by simply adjusting the sufficient statistics to reflect changes in 1-best candidates.

2.3 The RYPT Metric

We suggest here a new metric that combines the best of both worlds, in that it is based on human judgment, but that is a viable metric to be used in the MERT phase. The key to the feasiblity is the reliance on a *database* of human judgment rather than immendiate feedback for each candidate, and so human feedback is only needed once, and the collected human judgments can be reused over and over again by an automatic scorer.

The basic idea is to reward syntactic constituents in the source sentence that get aligned to "acceptable" translations in the candidate sen-

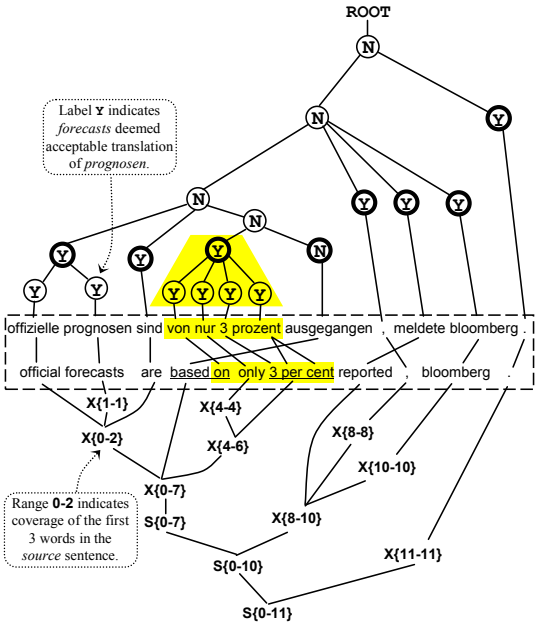

Figure 2: The source parse tree (top) and the candidate derivation tree (bottom). Nodes in the parse tree with a thick border correspond to the frontier node set with $\texttt{maxLen} = 4$. The human annotator only sees the portion surrounded by the dashed rectangle, including the highlighting (though excluding the word alignment links).

tence, and penalize constituents that do not. For instance, consider the source-candidate sentence pair of Figure 2. To evaluate the candidate translation, the source parse tree is first obtained (Dubey, 2005), and each subtree is matched with a substring in the candidate string. If the source substring covered by this subtree is translated into an acceptable substring in the candidate, that node gets a YES label. Otherwise, the node gets a NO label.

The metric we propose is taken to be the ratio of YES nodes in the parse tree (or RYPT). The candidate in Figure 2, for instance, would get a RYPT score of $13/18 = 0.72$.

To justify its use as a proxy for HTER-like metrics, we need to demonstrate that this metric correlates well with human judgment. But it is also important to show that we can obtain the YES/NO label assignments in an efficient and affordable manner. At first glance, this seems to require a human to provide judgments for each candidate, much like with HTER. But we describe in the next section strategies that minimize the number of judgments we need to actually collect.

3 Collecting Human Judgments

The first assumption we make to minimize the number of human judgments, is that once we have a judgment for a source-candidate substring pair, that same judgment can be used across all candidates for this source sentence. In other words, we build a database for each source sentence, which consists of <source substring,target substring,judgment> entries. For a given source substring, multiple entries exist, each with a different target candidate substring. The judgment field is one of YES, NO, and NOT SURE.

Note that the entries do not store the full candidate string, since we reuse a judgment across all the candidates of that source sentence. For instance, if we collect the judgment:

<der patient,the patient,YES>

from the sentence pair:

der patient wurde isoliert .
the patient was isolated .

then this would apply to any candidate translation of this source sentence. And so all of the following substrings are labeled YES as well:

the patient isolated .
the patient was in isolation .
the patient has been isolated .

Similarly, if we collect the judgment:

<der patient,of the patient,NO>

from the sentence pair:

der patient wurde isoliert .
of the patient was isolated .

then this would apply to any candidate translation of the source, and the following substrings are labeled NO as well:

of the patient isolated .
of the patient was in isolation .
of the patient has been isolated .

The strategy of using judgments across candidates reduces the amount of labels we need to collect, but evaluating a candidate translation for the source sentence of Figure 2 would still require obtaining 18 labels, one for each node in the parse tree. Instead of querying a human for each one

of those nodes, it is quite reasonable to percolate existing labels up and down the parse tree: if a node is labeled NO, this likely means that all its ancestors would also be labeled NO, and if a node is labeled YES, this likely means that all its descendents whould also be labeled YES.

While those two strategies (using judgments across candidates, and percolating labels up and down the tree) are only approximations for the true labels, employing them considerably reduces the amount of data we need to collect.

3.1 Obtaining Source-to-Candidate Alignments

How do we determine which segment of the candidate sentence aligns to a given source segment? Given a word alignment between the source and the candidate, we take the target substring to contain any word aligned with at least one word in the source segment. One could run an aligner (e.g. GIZA++) on the two sentences to obtain the word alignment, but we take a different approach.

We use Joshua (Li et al., 2009), in our experiments. Joshua is a hierarchical parsing-based MT system, and it can be instructed to produce derivation trees instead of the candidate sentence string itself. Furthermore, each node in the derivation tree is associated with the two indices in the source sentence that indicate the segment corresponding to this derivation subtree (the numbers indicated in curly brackets in Figure 2).

Using this information, we are able to recover most of the phrasal alignments. There are other phrasal alignments that can be deduced from the structure of the tree indirectly, by systematically discarding source words that are part of another phrasal alignment. For instance, in Figure 2, one can observe the alignment (offizielle,prognosen,sind)–(official,forecasts,are) and the alignment (prognosen)–(forecasts) to deduce (offizielle,sind)–(official,are).

Although some of the phrasal alignment are one-to-one mappings, many of them are many-to-many. By construction, any deduced many-to-many mapping has occurred in the training parallel corpus at least once. And so we recover the individual word alignments by consulting the parallel corpus from which the grammar rules were extracted (which requires maintaining the word alignments obtained prior to rule extraction).[5]

We emphasize here that our recovery of word alignment from phrasal alignment is independent from the hierarchical and parsing-based nature of the Joshua system. And so the alignment approach we suggest here can be applied to a different MT system as well, as long as that system provides phrasal alignment along with the output. In particular, a phrase-based system such as Moses can be modified in a straightforward manner to provide phrasal alignments, and then apply our method.

4 Data Collection

We chose the WMT08 German-English news dataset to work with, and since this is an investigative study of a novel approach, we collected judgments for a subset of 250 source sentences from the development set for the set of candidate sentences produced in the last iteration of a MERT run optimizing BLEU on the full 2051-sentence development set. The MT system we used is Joshua (Li et al., 2009), a software package that comes complete with a grammar extraction module and a MERT module, in addition to the decoder itself.

What segments of the source should be chosen to be judged? We already indicated that we limit ourselves, by definition of RYPT, to segments that are covered exactly by a subtree in the source parse tree. This has a couple of nice advantages: it allows us to present an annotator with a high number of alternatives judged simulataneously (since the annotator is shown a source segment and several candidates, not just one), and this probably also makes judging them easier – it is reasonable to assume that strings corresponding to syntactic constituents are easier to process by a human.

Our query selection strategy attempts to maximize the amount of YES/NO percolation that would take place. We therefore ensure that for any 2 queries, the corresponding source segments do not overlap: such overlap indicates that one subtree is completely contained within the other. Having both queries (in the same batch) might be redundant if we use the above percolation procedure.

The idea is to select source segments so that they fully cover the entire source sentence, but have no overlap amongst them. In one extreme, each query would correspond to an entire parse tree. This is not ideal since the overwhelming majority of the judgments will most likely be NO,

[5] We incorporated our implementation of the source-candidate aligner into the Joshua software as a new `aligner` package.

which does not help identify where the problem is. In the other extreme, each query would correspond to a subtree rooted at a preterminal. This is also not ideal, since it would place too much emphasis on translations of unigrams.

So we need a middle ground. We select a maximum-source-length `maxLen` to indicate how long we're willing to let source segments be. Then we start at the root of the parse tree, and propagate a "frontier" node set down the parse tree, to end up with a set of nodes that fully cover the source sentence, have no overlap amongst them, and with each covering no more than `maxLen` source words. For instance, with `maxLen` set to 4, the frontier set of Figure 2 are the nodes with a thick border. An algorithmic description is provided in Algorithm 1.

Algorithm 1 Constructing the frontier node set for a parse tree.

Input: A source parse tree T rooted at `ROOT`, and a maximum source length `maxLen`.
Return: A nonempty set `frontierSet`, containing a subset of the nodes in T.

```
1.  Initialize frontierSet to the empty set.
2.  Initialize currNodes to {ROOT}.
3.  while currNodes is not empty do
4.      Initialize newNodes to the empty set.
5.      for each node N in currNodes do
6.          if N covers ≤ maxLen source words
            then
7.              Add N to frontierSet.
8.          else
9.              Add children of N to newNodes.
10.         end if
11.         Set currNodes = newNodes
12.     end for
13. end while
14. Return frontierSet.
```

This would ensure that our queries cover between 1 and `maxLen` source words, and ensures they do not overlap, which would allow us to take full advantage of the downward-`YES` and upward-`NO` percolation. We set `maxLen` $= 4$ based on a pilot study of 10 source sentences and their candidates, having observed that longer segments tend to always be labeled as `NO`, and shorter segments tend to be so deep down the parse tree.

4.1 Amazon Mechanical Turk

We use the infrastructure of Amazon's Mechanical Turk (AMT)[6] to collect the labels. AMT is a virtual marketplace that allows "requesters" to create and post tasks to be completed by "workers" around the world. To create the tasks (called *Human Intelligence Tasks*, or HITs), a requester supplies an HTML template along with a comma-separated-values database, and AMT automatically creates the HITs and makes them available to workers. The queries are displayed as an HTML page (based on the provided HTML template), with the user indicating the label (`YES`, `NO`, or `NOT SURE`) by selecting the appropriate radio button. The instructions read, in part:[7]

> You are shown a "source" German sentence with a highlighted segment, followed by several candidate translations with corresponding highlighted segments. Your task is to decide if each highlighted English segment is an acceptable translation of the highlighted German segment.

In each HIT, the worker is shown up to 10 alternative translations of a highlighted source segment, with each itself highlighted within a full candidate string in which it appears. To aid the worker in the task, they are also shown the reference translation, with a highlighted portion that corresponds to the source segment, deduced using word alignments obtained with GIZA++.[8]

4.2 Cost of Data Collection

The total number of HITs created was 3873, with the reward for completing a HIT depending on how many alternative translations are being judged. On average, each HIT cost 2.1 cents and involved judging 3.39 alternatives. 115 distinct workers put in a total of 30.82 hours over a period of about 4 days. On average, a label required 8.4 seconds to determine (i.e. at a rate of 426 labels per hour). The total cost was $81.44: $21.43 for Amazon's commission, $53.47 for wages, and

[6] AMT's website: `http://www.mturk.com`.

[7] Template and full instructions can be viewed at `http://cs.jhu.edu/~ozaidan/hmert`.

[8] These alignments are not always precise, and we do note that fact in the instructions. We also deliberately highlight the reference substring in a different color to make it clear that workers should judge a candidate substring primarily based on the source substring, not the reference substring.

$6.54 for bonuses[9], for a cost per label of 0.62 cents (i.e. at a rate of 161.32 labels per dollar). Excluding Amazon's commission, the effective hourly 'wage' was $1.95.

5 Experimental Results and Analysis

By limiting our queries to source segments corresponding to frontier nodes with $\texttt{maxLen} = 4$, we obtain a total of 3601 subtrees across the 250 sentences, for an average of 14.4 per sentence. On average, each subtree has 3.65 alternative translations. Only about 4.8% of the judgments were returned as `NOT SURE` (or, occasionally, blank), with the rest split into 35.1% `YES` judgments and 60.1% `NO` judgments.

The coverage we get before percolating labels up and down the trees is 39.4% of the nodes, increasing to a coverage of 72.9% after percolation. This is quite good, considering we only do a single data collection pass, and considering that about 10% of the subtrees do not align to candidate substrings to begin with (e.g. single source words that lack a word alignment into the candidate string).

The main question, of course, is whether or not those labels allow us to calculate a RYPT score that is reliably correlated with human judgment. We designed an experiment to compare the predictive power of RYPT vs. BLEU. Given the candidate set of a source sentence, we rerank the candidate set according to RYPT and extract the top-1 candidate, and we rerank the candidate set according to BLEU, and extract the top-1 candidate. We then present the two candidates to human judges, and ask them to choose the one that is a more adequate translation. For reliability, we collect 3 judgments per sentence pair comparison, instead of just 1.

The results show that RYPT significantly outperforms BLEU when it comes to predicting human preference, with its choice prevailing in 46.1% of judgments vs. 36.0% for BLEU, with 17.9% judged to be of equal quality (left half of Table 1). This advantage is especially true when the judgments are grouped by sentence, and we examine cases of strong agreement among the three annotators (Table 2): whereas BLEU's candidate is strongly preferred in 32 of the candidate pairs (bottom 2 rows), RYPT's candidate is strongly preferred in about double that number: 60 candidate pairs (top 2 rows).

This is quite a remarkable result, given that BLEU, by definition, selects a candidate that has significant overlap with the reference shown to the annotators to aid in their decision-making. This means that BLEU has an inherent advantage in comparisons where both candidates are more or less of equal quality, since annotators are encouraged (in the instructions) to make a choice even if the two candidates seem of be of equal quality at first glance. Pressed to make such a choice, the annotator is likely to select the candidate that superficially 'looks' more like the reference to be the 'better' of the two candidates. That candidate will most likely be the BLEU-selected one.

To test this hypothesis, we repeated the experiment *without showing the annotators the reference translations*, and limited data collection to workers living in Germany, making judgments based only on the source sentences. (We only collected one judgment per source sentence, since German workers on AMT are in short supply.)

As expected, the difference is even more pronounced: human judges prefer the RYPT-selected candidate 45.2% of the time, while BLEU's candidate is preferred only 29.2% of the time, with 25.6% judged to be of equal quality (right half of Table 1). Our hypothesis is further supported by the fact that most of the gain of the "equal-quality" category comes from BLEU, which loses 6.8 percentage points, whereas RYPT's share remains largely intact, losing less than a single percentage point.

5.1 Analysis of Data Collection

Recall that we minimize data collection by performing label percolation and by employing a frontier node set selection strategy. While the results just presented indicate those strategies provide a good approximation of some 'true' RYPT score, label percolation was a strategy based primarily on intuition, and choosing $\texttt{maxLen} = 4$ for frontier set construction was based on examining a limited amount of preliminary data.

Therefore, and in addition to encouraging empricial results, we felt a more rigorous quantitative analysis was in order, especially with future, more ambitious annotation projects on the horizon. To this end, we collected a *complete* set of judgments for 50 source sentences and their candidates. That is, we generated a query for each and every node

[9] We would review the collected labels and give a 20% reward for good workers to encourage them to come back and complete more HITs.

	References shown; unrestricted		References **not** shown; restricted to DE workers	
Preferred candidate	# judgments	% judgments	# judgments	% judgments
Top-1 by RYPT	346	46.1	113	45.2
Top-1 by BLEU	270	36.0	73	29.2
Neither	134	17.9	64	25.6
Total	750	100.0	250	100.0

Table 1: Ranking comparison results. The left half corresponds to the experiment (open to all workers) where the English reference was shown, whereas the right half corresponds to the experiment (open only to workers living in Germany) where the English reference was *not* shown.

Aggregate	# sentences	% sentences	Aggregate	# sentences	% sentences
RYPT +3	45	18.0			
RYPT +2	15	6.0	RYPT +any	120	48.0
RYPT +1	60	24.0			
$\pm$ 0	42	16.8	$\pm$ 0	42	16.8
BLEU +1	55	22.0			
BLEU +2	5	2.0	BLEU +any	88	35.2
BLEU +3	28	11.2			
Total	250	100.0	Total	250	100.0

Table 2: Ranking comparison results, grouped by sentence. This table corresponds to the left half of Table 1. 3 judgments were collected for each comparison, with the "aggregate" for a comparison calculated from these 3 judgments. For instance, an aggregate of "RYPT +3" means all 3 judgments favored RYPT's choice, and "RYPT +1" means one more judgment favored RYPT than did BLEU.

in the source parse tree, instead of limiting ourselves to a frontier node set. (Though we did limit the length of a source segment to be ≤ 7 words.) This would allow us to judge the validity of label percolation, and under different `maxLen` values.

Furthermore, we collected *multiple* judgments for each query in order to minimize the effet of bad/random annotations. For each of 5580 generated queries, we collected five judgments, for a total of 27,900 judgments.[10] As before, the annotator would pick one of `YES`, `NO`, and `NOT SURE`.

First, collecting multiple judgments allowed us to investigate inter-annotator agreement. In 68.9% of the queries, at least 4 of the 5 annotators chose the same label, signifying a high degree of inter-annotator agreement. This is especially encouraging considering that we identified about 15% of the HITs as being of poor quality, and blocked the respective annotators from doing further HITs.[11]

We then examine the *applicability* and *validity* of label percolation. For each of 7 different values for Algorithm 1's `maxLen`, we ignore all but labels that would be requested under that `maxLen` value, and percolate the labels up and down the tree. In Figure 3 we plot the coverage before and after percolation (middle two curves), and observe expansion in coverage across different values of `maxLen`, peaking at about +33% for `maxLen`$= 4$ and 5, with most of the benefit coming from `YES` percolation (bottom two curves).

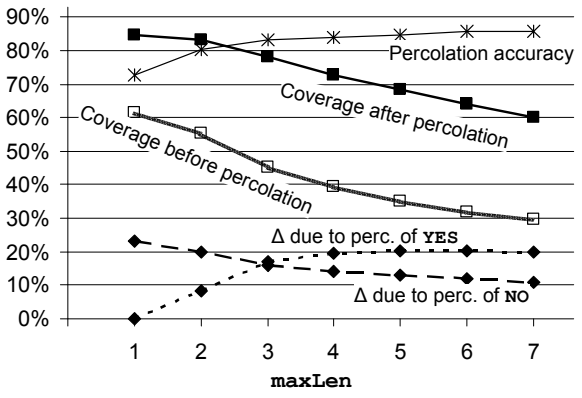

Figure 3: Label percolation under different `maxLen` values. The bottom two curves are the breakdown of the difference between the middle two. Accuracy is measured against majority votes.

[10] For a given query, the five collected judgments are from five different annotators, since AMT ensures an annotator is never shown the same HIT twice.

[11] It is especially easy to identify (and then block) such annotators when they submit a relatively large number of HITs, since inspecting some of their annotations would indicate they are answering randomly and/or inconsistently.

We also measure the accuracy of labels deduced from percolation (top curve of Figure 3). We define a percolated label to be correct if it matches the label given by a majority vote over the collected labels for that particular node. We find that accuracy at low `maxLen` values is significantly lower than at higer values (e.g. 72.6% vs. 84.1% for 1 vs. 4). This means a middle value such as 3 or 4 is optimal. Higher values could be suitable if we wish to emphasize translation fluency.

6 Related Work

Nießen et al. (2000) is an early work that also constructs a database of translations and judgments. There, a source sentence is stored along with all the translations that have already been manually judged, along with their scores. They utilize this database to carry out "semi-automatic" evaluation in a fast and convenient fashion thanks to tool they developed with a user-friendly GUI.

In their annual evaluation, the WMT workshop has effectively conducted manual evaluation of submitted systems over the past few years by distributing the work across tens of volunteers, though they relied on a self-designed online portal. On the other hand, Snow et al. (2008) illustrate how AMT can be used to collect data in a "fast and cheap" fashion, for a number of NLP tasks, such as word sense disambiguation. They go a step further and model the behavior of their annotators to reduce annotator bias. This was possible as they collect multiple judgments for each query from multiple annotators.

The question of how to design an automatic metric that best approximates human judgment has received a lot of attention lately. NIST started organizing the Metrics for Machine Translation Challenge (MetricsMATR) in 2008, with the aim of developing automatic evaluation metrics that correlate highly with human judgment of translation quality. The latest WMT workshop (Callison-Burch et al., 2009) also conducted a full assessment of how well a suite of automatic metrics correlate with human judgment.

7 Future Work

This pilot study has demonstrated the feasibility of collecting a large number of human judgments, and has shown that the RYPT metric is better than BLEU at picking out the best translation. The next step is to run a complete MERT run. This will involve collecting data for thousands of alternative translations for several hundreds source sentences. Based on our analysis, this it should be cost-effective to solicit these judgments using AMT. After training MERT using RYPT as an objective function the, the next logical step would be to compare two outputs of a system. One output would have parameters optimized to BLEU and the other to RYPT. The hope is that the RYPT-trained system would be better under the final HTER evaluation than the BLEU-trained system.

We are also investigating a probabilistic approach to percolating the labels up and down the tree, whereby the label of a node is treated as a random variable, and inference is performed based on values of the other observed nodes, as well as properties of the source/candidate segment. Cast this way, a probabilistic approach is actually quite appealing, and one could use collected data to train a prediction model (such as a Markov random field).

8 Summary

We propose a human-based metric, RYPT, that is quite feasible to optimize using MERT, relying on the redundancy in the candidate set, and collecting judgments using Amazon's Mechanical Turk infrastructure. We show this could be done in a quite cost-effective manner, and produces data of good quality. We show the effectiveness of the metric by illustrating that it is a better predictor of human judgment of translation quality than BLEU, the most commonly used metric in MT. We show this is the case even with a modest amount of data that does not cover the entirety of all parse trees, on which the metric is dependent. The collected data represents a database that can be reused over and over again, hence limiting human feedback to the initial phase only.

Acknowledgments

This research was supported by the EuroMatrix-Plus project funded by the European Commission (7th Framework Programme), by the Defense Advanced Research Projects Agency's GALE program under Contract No. HR0011-06-2-0001, and the US National Science Foundation under grant IIS-0713448. The views and findings are the authors' alone.

References

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of Bleu in machine translation research. In *Proceedings of EACL*, pages 249–256.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

David Chiang, Steve DeNeefe, Yee Seng Chan, and Hwee Tou Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP*, pages 610–619.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Deborah Coughlin. 2003. Correlating automated and human assessments of machine translation quality. In *Proceedings of MT Summit IX*.

Amit Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proceedings of ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 127–133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demo and Poster Sessions*, pages 177–180.

LDC. 2005. Linguistic data annotation specification: Assessment of fluency and adequacy in translations. Revision 1.5.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139.

Sonja Nießen, Franz Josef Och, Gregor Leusch, and Hermann Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. In *Proceedings of LREC*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

Kishore Papineni, Salim Poukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.