An Enriched MT Grammar for Under $100

Omar F. Zaidan and **Juri Ganitkevitch**
Dept. of Computer Science, Johns Hopkins University
Baltimore, MD 21218, USA
{ozaidan,juri}@cs.jhu.edu

Abstract

We propose a framework for improving output quality of machine translation systems, by operating on the level of grammar rule features. Our framework aims to give a boost to grammar rules that appear in the derivations of translation candidates that are deemed to be of good quality, hence making those rules more preferable by the system. To that end, we ask human annotators on Amazon Mechanical Turk to compare translation candidates, and then interpret their preferences of one candidate over another as an implicit preference for one derivation over another, and therefore as an implicit preference for one or more grammar rules. Our framework also allows us to generalize these preferences to grammar rules corresponding to a previously unseen test set, namely rules for which no candidates have been judged.

1 Introduction

When translating between two languages, state-of-the-art statistical machine translation systems (Koehn et al., 2007; Li et al., 2009) generate candidate translations by relying on a set of relevant grammar (or phrase table) entries. Each of those entries, or *rules*, associates a string in the source language with a string in the target language, with these associations typically learned by examining a large parallel bitext. By the very nature of the translation process, a target side sentence $\mathbf{e}$ can be a candidate translation for a source sentence $\mathbf{f}$ only if $\mathbf{e}$ can be constructed using a small subset of the grammar, namely the subset of rules with source side sequences relevant to the word sequence of $\mathbf{f}$. However, even this limited set of candidates (call it $\mathcal{E}(\mathbf{f})$) is quite large, with $|\mathcal{E}(\mathbf{f})|$ growing exponentially in the length of $\mathbf{f}$. The system is able to rank the translations within $\mathcal{E}(\mathbf{f})$ by assigning a score $s(\mathbf{e})$ to each candidate translation. This score is the dot product:

$$s(\mathbf{e}) = \vec{\boldsymbol{\varphi}}(\mathbf{e}) \cdot \vec{\boldsymbol{w}} \tag{1}$$

where $\vec{\boldsymbol{\varphi}}(\mathbf{e})$ is a feature vector characterizing $\mathbf{e}$, and $\vec{\boldsymbol{w}}$ is a system-specific weight vector characterizing the system's belief of how much the different features reflect translation quality. The features of a candidate $\mathbf{e}$ are computed by examining the way $\mathbf{e}$ is constructed (or *derived*), and so if we let $\boldsymbol{d}(\mathbf{e})$ be the derivation of $\mathbf{e}$, the feature vector can be denoted:[1]

$$\vec{\boldsymbol{\varphi}}(\boldsymbol{d}(\mathbf{e})) = \langle \varphi_1(\boldsymbol{d}(\mathbf{e})), \ldots, \varphi_m(\boldsymbol{d}(\mathbf{e})) \rangle \tag{2}$$

where $\varphi_i(\boldsymbol{d}(\mathbf{e}))$ is the value of ith feature function of $\boldsymbol{d}(\mathbf{e})$ (with a corresponding weight w_i in $\vec{\boldsymbol{w}}$).

To compute the score for a candidate, we examine its derivation $\boldsymbol{d}(\mathbf{e})$, enumerating the grammar rules used to construct $\mathbf{e}$: $\boldsymbol{d}(\mathbf{e}) = (r_1, \ldots, r_k)$. Typically, each of the rules will itself have a vector of m features, and we calculate the value of a derivation feature $\varphi_i(\boldsymbol{d}(\mathbf{e}))$ as the sum of the ith feature over all rules in the derivation:

$$\varphi_i(\boldsymbol{d}(\mathbf{e})) = \sum_{r \in \boldsymbol{d}(\mathbf{e})} \varphi_i(r) \tag{3}$$

[1] There are other features computed directly, without examining the derivation (e.g. candidate length, language model score), but we omit these features from the motivation discussion for clarity.

Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, pages 93–98,
Los Angeles, California, June 2010.

These features are usually either relative frequencies estimated from the training corpus, relating the rule's source and target sides, or features that characterize the structure of the rule itself, independently from the corpus.

Either way, the weight w_i is chosen so as to reflect some belief regarding the correlation between the ith feature and translation quality. This is usually done by choosing weights that maximize performance on a tuning set separate from the training bitext. Unlike system weights, the grammar rule feature values are fixed once extracted, and are not modified during this tuning phase. In this paper, we propose a framework to augment the feature set to incorporate additional intuition about how likely a rule is to produce a translation preferred by a human annotator. This knowledge is acquired by directly asking human judges to compare candidate translations, therefore determining which subset of grammar rules annotators seem to prefer over others. We also seek to generalize this intuition to rules for which no candidates were judged, hence allowing us to impact a much larger set of rules than just those used in translating the tuning set.

The paper is organized as follows. We first give a general description of our framework. We then discuss our data collection efforts on Amazon Mechanical Turk for an Urdu-English translation task, and make explicit the type of judgments we collect and how they can be used to augment grammar rules. Before concluding, we propose a framework for generalizing judgments to unseen grammar rules, and analyze the data collection process.

2 The General Framework

As initially mentioned, when tuning a SMT system on a development set, we typically only perform high-level optimization of the system weights. In this section we outline an approach that could allow for lower-level optimization, on the level of individual grammar rules.

We kick off the process by soliciting judgments from human annotators regarding the quality of a subset of candidates (the following section outlines how candidates are chosen). The resulting judgments on sentences are interpreted to be judgments on individual grammar rules used in the derivations of these candidates. And so, if an annotator declares a candidate to be of high quality, this is considered a vote of confidence on the individual rules giving rise to this candidate, and if an annotator declares a candidate to be of lowl quality, this is considered a vote of no confidence on the individual rules.

To make use of the collected judgments, we extend the set of features used in the decoder by a new feature λ:

$$\vec{\varphi}' = \langle \varphi_1, \ldots, \varphi_m, \lambda \rangle \quad (4)$$

This feature is the cornerstone of our framework, as it will hold the quantified and cumulated judgments for each rule, and will be used by the system at decoding time, in addition to the existing m features, incorporating the annotators' judgments into the translation process.[2]

The range of possible values for this feature, and how the feature is computed, depends on how one chooses to ask annotators to score candidates, and what form those judgments assume (i.e. are those judgments scores on a scale? Are they "better" vs. "worse" judgments, and if so, compared to how many other possibilities?). At this point, we will only emphasize that the value of λ should reflect the annotators' preference for the rule, and that it should be computed from the collected judgments. We will propose one such method of computing λ in Section 4, after describing the type of judgments we collected.

3 Data Collection

We apply our approach to an Urdu-to-English translation task. We used a syntactically rich SAMT grammar (Venugopal and Zollmann, 2006), where each rule in the grammar is characterized by 12 features. The grammar was provided by Chris Callison-Burch (personal communication), and was extracted from a parallel corpus of 88k sentence pairs.[3] One system using this grammar produced significantly improved output over submissions to the NIST 2009 Urdu-English task (Baker et al., 2009).

We use the Joshua system (Li et al., 2009) as a decoder, with system weights tuned using

[2] In fact, the collected judgments can only cover a small portion of the grammar. We address this coverage problem in Section 4.

[3] LDC catalog number LDC2009E12.

Z-MERT (Zaidan, 2009) on a tuning set of 981 sentences, a subset of the 2008 NIST Urdu-English test set.[4] We choose candidates to be judged from the 300-best candidate lists.[5]

Asking a worker to make a quantitative judgment of the quality of a particular candidate translation (e.g. on a 1–7 scale) is a highly subjective and annotator-dependent process. Instead, we present workers with pairs of candidates, and ask them to judge which candidate is of better quality.

How are candidate pairs chosen? We would like a judgment to have the maximum potential for being informative about specific grammar rules. In essense, we prefer a pair of candidates if they have highly similar derivations, yet differ noticeably in terms of how the decoder ranks them. In other words, if a relatively minimal change in derivation causes a relatively large difference in the score assigned by the decoder, we are likely to attribute the difference to very few rule comparisons (or perhaps only one), hence focusing the comparison on individual rules, all the while shielding the annotators from having to compare grammar rules directly.

Specifically, each pair of candidates $(\mathbf{e}, \mathbf{e}')$ is assigned a potential score $\pi(\mathbf{e}, \mathbf{e}')$, defined as:

$$\pi(\mathbf{e}, \mathbf{e}') = \frac{s(\mathbf{e})s(\mathbf{e}')}{lev(\boldsymbol{d}(\mathbf{e}), \boldsymbol{d}(\mathbf{e}'))}, \quad (5)$$

where $s(\mathbf{e})$ is the score assigned by the decoder, and $lev(\boldsymbol{d}, \boldsymbol{d'})$ is a distance measure between two derivations which we will now descibe in more detail. In Joshua, the derivation of a candidate is captured fully and exactly by a derivation tree, and so we define $lev(\boldsymbol{d}, \boldsymbol{d'})$ as a tree distance metric as follows. We first represent the trees as strings, using the familiar nested string representation, then compute the word-based Levenshtein edit distance between the two strings. An edit has a cost of 1 in general, but we assign a cost of zero to edit operations on terminals, since we want to focus on the structure of the derivation trees, rather than on terminal-level lexical choices.[6] Furthermore, we ignore differences in "pure" pre-terminal rules, that only have terminals as their right-hand side. These decisions effectively allow us to focus our efforts on grammar rules with at least one nonterminal in their right-hand side.

We perform the above potential computation on all pairs formed by the cross product of the top 10 candidates and the top 300 candidates, and choose the top five pairs ranked by potential.

Our HIT template is rather simple. Each HIT screen corresponds to a single source sentence, which is shown to the worker along with the five chosen candidate pairs. To aid workers who are not fluent in Urdu[7] better judge translation quality, the HIT also displays one of the available references for that source sentence. To eliminate potential bias associated with the order in which candidates are presented (an annotator might be biased to choosing the first presented candidate, for example), we present the two candidates in random or- der. Furthermore, for quality assurance, we embed a sixth candidate pair to be judged, where we pair up a randomly chosen candidate with another reference for that sentence.[8] Presumably, a faithful worker would be unlikely to prefer a random candidate over the reference, and so this functions as an embedded self-verification test. The order of this test, relative to the five original pairs, is chosen randomly.

4 Incorporating the Judgements

4.1 Judgement Quantification

The judgments we obtain from the procedure described in the previous section relate pairs of candidate translations. However, we have defined the accumulation feature λ as a feature for each rule. Thus, in order to compute λ, we need to project the judgments onto the rules that tell the two candidates apart. A simple way to do this is the following: for a judged candidate pair $(\mathbf{e}, \mathbf{e}')$ let $U(\mathbf{e})$ be the set of

[4] LDC catalog number LDC2009E11.

[5] We exclude source sentences shorter than 4 words long or that have fewer than 4 candidate translations. This eliminates roughly 6% of the development set.

[6] This is not to say that lexical choices are not important, but lexical choice is heavily influenced by context, which is not captured well by grammar rules. Furthermore, lexical choice is a phenomenon already well captured by the score assigned to the candidate by the language model, a feature typically included when designing $\vec{\varphi}$.

[7] We exclude workers from India and restrict the task to workers with an existing approval rating of 90% or higher.

[8] The tuning set contains at least three different human references for each source sentence, and so the reference "candidate" shown to the worker is not the same as the sentence already identified as a reference.

rules that appear in $\boldsymbol{d}(\mathbf{e})$ but not in $\boldsymbol{d}(\mathbf{e}')$, and vice versa.[9] We will assume that the jugdment obtained for $(\mathbf{e}, \mathbf{e}')$ applies for every rule pair in the cartesian product of $U(\mathbf{e})$ and $U(\mathbf{e}')$. This expansion yields a set of judged grammar rule pairs $\mathcal{J} = \{(a, b)\}$ with associated vote counts $v_{a>b}$ and $v_{b>a}$, capturing how often the annotators preferred a candidate that was set apart by a over a candidate containing b, and vice versa.

So, following our prior definiton as an expression of the judges' preference, we can calculate the value of λ for a rule r as the relative frequency of favorable judgements:

$$\lambda(r) = \frac{\sum_{(r,b)\in\mathcal{J}} v_{r>b}}{\sum_{(r,b)\in\mathcal{J}} v_{b>r} + v_{r>b}} \quad (6)$$

4.2 Generalization to Unseen Rules

This approach has a substantial problem: λ, computed as given above, is undefined for a rule that was never judged (i.e. a rule that never set apart a pair of candidates presented to the annotators). Furthermore, as described, the coverage of the collected judgments will be limited to a small subset of the entire grammar, meaning that when the system is asked to translate a new source sentence, it is highly unlikely that the relevant grammar rules would have already been judged by an annotator. Therefore, it is necessary to generalize the collected judgments/votes and propagate them to previously unexamined rules.

In order to do this, we propose the following general approach: when observing a judgment for a pair of rules $(a, b) \in \mathcal{J}$, we view that judgement not as a vote on one of them specifically, but rather as a comparison of rules similar to a versus rules similar to b. When calculating $\lambda(r)$ for any rule r we use a distance measure over rules, Δ, to estimate how each judgment in $\mathcal{J}$ projects to r. This leads to the following modified computatio of $\lambda(r)$:

$$\lambda(r) = \sum_{(a,b)\in\mathcal{J}} \frac{\Delta(a, r)v'_{b>a} + \Delta(b, r)v'_{a>b}}{\Delta(a, r) + \Delta(b, r)} \quad (7)$$

[9]The way we select candidate pairs ensures that $U(\mathbf{e})$ and $U(\mathbf{e}')$ are both small and expressive in terms of impact on the decoder ranking. On our data $U(\mathbf{e})$ contained an average of 4 rules.

where $v'_{a>b}$ (and analogously $v'_{b>a}$) is defined as the relative frequency of a being preferred over b:

$$v'_{a>b} = \frac{v_{a>b}}{v_{a>b} + v_{b>a}}$$

4.3 A Vector Space Realization

Having presented a general framework for judgment generalization, we will now briefly sketch a concrete realization of this approach.

In order to be able to use the common distance metrics on rules, we define a *rule vector space*. The basis of this space will be a new set of rule features designed specifically for the purpose of describing the structure of a rule, $\vec{\boldsymbol{\psi}} = \langle\psi_1, \ldots, \psi_k\rangle$. Provided the exact features chosen are expressive and well-distributed over the grammar, we expect any conventional distance metric to correlate with rule similarity.

We deem a particular ψ_i good if it quantifies a quality of the rule that describes the rule's nature rather than the particular lexical choices it makes, i.e. a statistic (such as the rule length, arity, number of lexical items in the target or source side or the average covered span in the training corpus), information relevant to the rule's effect on a derivation (such as nonterminals occuring in the rule and wheter they are re-ordered) or features that capture frequent lexical cues that carry syntactic information (such as the co-occurrence of function words in source and target language, possibly in conjunction with certain nonterminal types).

5 Results and Analysis

The judgments were collected over a period of about 12 days (Figure 1). A total of 16,374 labels were provided (2,729 embedded test labels + 13,645 'true' labels) by 658 distinct workers over 83.1 hours (i.e. each worker completed an average of 4.2 HITs over 7.6 minutes). The reward for each HIT was \$0.02, with an additional \$0.005 incurred for Amazon Fees. Since each HIT provides five labels, we obtain 200 (true) labels on the dollar. Each HIT took an average of 1.83 minutes to complete, for a labeling rate of 164 true labels/hour, and an effective wage of \$0.66/hour. The low reward does not seem to have deterred Turkers from completing our HITs faithfully, as the success rate on the embedded

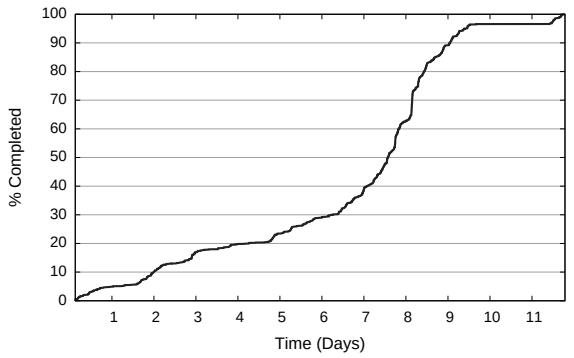

Figure 1: Progress of HIT submission over time. There was a hiatus of about a month during which we collected no data, which we are omitting for clairty.

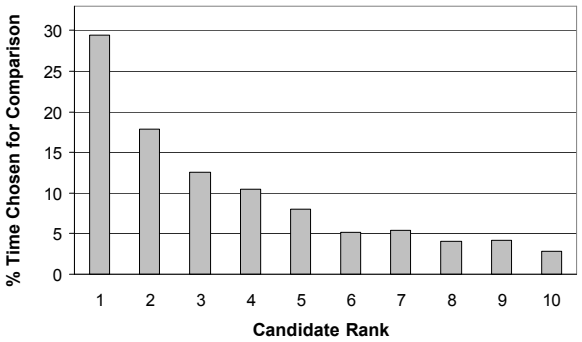

Figure 2: Histogram of the rank of the **higher**-ranked candidate chosen in pair comparisons. For instance, in about 29% of chosen pairs, the higher-ranked candidate was the top candidate (of 300) by decoder score.

True Questions	
Preferred	**%**
High-Ranked	40.0%
Low-Ranked	24.1%
No Difference	35.9%

Validation Questions	
Preferred	**%**
Reference	83.7%
Random Candidate	11.7%
No Difference	4.65%

Table 1: Distributions of the collected judgments over the true questions and over the embedded test questions. "High-Ranked" (resp. "Low-Ranked") refers to whether the decoder assigned a high (low) score to the candidate. And so, annotators agreed with the decoder 40.0% of the time, and disagreed 24.1% of the time.

(a) [NP] → ⟨ [NP] [NN+IN] کے | the [NN+IN] [NP] ⟩

(b) [NP] → ⟨ [NP] کے [NN] کے | [NN] of [NP] ⟩

Figure 3: A example pair of rules for which judgements were obtained. The first rule is preferred by the decoder, while human annotators favor the second rule.

questions was quite high (Table 1).[10] From our set of comparatively judged candidate translations we extracted competing rule pairs. To reduce the influence of lexical choices and improve comparability, we excluded pure preterminal rules and limited the extraction to rules covering the same span in the Urdu source. Figure 3 shows an interesting example of one such rule pair. While the decoder demonstrates a clear preference for rule (a) (including it into its higher-ranked translation 100% of the time), the Turkers tend to prefer translations generated using rule (b), disagreeing with the SMT system 60% of the time. This indicates that preferring the second rule in decoding may yield better results in terms of human judgment, in this case potentially due to the cleaner separation of noun phrases from the prepositional phrase.

We also examine the distribution of the chosen candidates. Recall that each pair consists of a **high**-ranked candidate from the top-ten list, and a **low**-ranked candidate from the top-300 list. The Histogram of the higher rank (Figure 2) shows that the high-ranked candidate is in fact a top-three candidate over 50% of the time. We also see (Figure 4) that the low-ranked candidate tends to be either close in rank to the top-ten list, or far away. This again makes sense given our definition of potential for a pair: potential is high if the derivations are very close (left mode) or if the decoder scores differ considerably (right mode).

Finally, we examine inter-annotator agreement, since we collect multiple judgments per query. We find that there is full agreement among the annotators in 20.6% of queries. That is, in 20.6% of queries, all three annotators answering that query gave the same answer (out of the three provided answers). This complete agreement rate is significantly higher than a rate caused by pure chance (11.5%). This is a positive result, especially given

[10] It should be mentioned that the human references themselves are of relatively low quality.

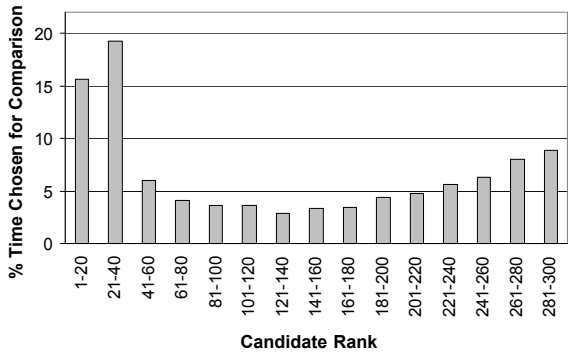

Figure 4: Histogram of the rank of the **lower**-ranked candidate chosen in pair comparisons. For instance, in about 16% of chosen candidate pairs, the lower-ranked candidate was ranked in the top 20.

how little diversity usually exists in n-best lists, a fact (purposely) exacerbated by our strategy of choosing highly similar pairs of candidates. On the other hand, we observe complete *dis*agreement in only 14.9% of queries, which is significantly lower than a rate caused by pure chance (which is 22.2%).

One thing to note is that these percentages are calculated after excluding the validation questions, where the complete agreement rate is an expectedly even higher 64.9%, and the complete disagreement rate is an expectedly even lower 3.60%.

6 Conclusions and Outlook

We presented a framework that allows us to "tune" MT systems on a finer level than system-level feature weights, going instead to the grammar rule level and augmenting the feature set to reflect collected human judgments. A system relying on this new feature during decoding is expected to have a slightly different ranking of translation candidates that takes human judgment into account. We presented one particular judgment collection procedure that relies on comparing candidate pairs (as opposed to evaluating a candidate in isolation) and complemented it with one possible method of propagating human judgments to cover grammar rules relevant to new sentences.

While the presented statistics over the collected data suggest that the proposed candidate selection procedure yields consistent and potentially informative data, the quantitative effects on a machine translation system remain to be seen.

Additionally, introducing λ as a new feature makes it necessary to find a viable weight for it. While this can be done trivially in running MERT on arbitrary development data, it may be of interest to extend the weight optimization procedure in order to preserve the partial ordering induced by the judgments as best as possible.

Acknowledgments

This research was supported by the EuroMatrix-Plus project funded by the European Commission, by the DARPA GALE program under Contract No. HR0011-06-2-0001, and the NSF under grant IIS-0713448.

References

Kathy Baker, Steven Bethard, Michael Bloodgood, Ralf Brown, Chris Callison-Burch, Glen Coppersmith, Bonnie Dorr, Wes Filardo, Kendall Giles, Anni Irvine, Mike Kayser, Lori Levin, Justin Martineau, Jim Mayfield, Scott Miller, Aaron Phillips, Andrew Philpot, Christine Piatko, Lane Schwartz, and David Zajic. 2009. Semantically informed machine translation (SIMT). In *SCALE 2009 Summer Workshop Final Report*, pages 135–139.

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

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

Ashish Venugopal and Andreas Zollmann. 2006. Syntax augmented machine translation via chart parsing. In *Proc. of the NAACL 2006 Workshop on Statistical Machine Translation*, pages 138–141. Association for Computational Linguistics.

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.