AMBER: A Modified BLEU, Enhanced Ranking Metric

Boxing Chen and Roland Kuhn
National Research Council of Canada, Gatineau, Québec, Canada
First.Last@nrc.gc.ca

Abstract

This paper proposes a new automatic machine translation evaluation metric: AMBER, which is based on the metric BLEU but incorporates recall, extra penalties, and some text processing variants. There is very little linguistic information in AMBER. We evaluate its system-level correlation and sentence-level consistency scores with human rankings from the WMT shared evaluation task; AMBER achieves state-of-the-art performance.

1 Introduction

Automatic evaluation metrics for machine translation (MT) quality play a critical role in the development of statistical MT systems. Several metrics have been proposed in recent years. Metrics such as BLEU (Papineni *et al.*, 2002), NIST (Doddington, 2002), WER, PER, and TER (Snover *et al.*, 2006) do not use any linguistic information - they only apply surface matching. METEOR (Banerjee and Lavie, 2005), METEOR-NEXT (Denkowski and Lavie 2010), TER-Plus (Snover *et al.*, 2009), MaxSim (Chan and Ng, 2008), and TESLA (Liu *et al.*, 2010) exploit some limited linguistic resources, such as synonym dictionaries, part-of-speech tagging or paraphrasing tables. More sophisticated metrics such as RTE (Pado *et al.*, 2009) and DCU-LFG (He *et al.*, 2010) use higher level syntactic or semantic analysis to score translations.

Though several of these metrics have shown better correlation with human judgment than BLEU, BLEU is still the *de facto* standard evaluation metric. This is probably due to the following facts:

1. BLEU is language independent (except for word segmentation decisions).
2. BLEU can be computed quickly. This is important when choosing a metric to tune an MT system.
3. BLEU seems to be the best tuning metric from a quality point of view - *i.e.*, models trained using BLEU obtain the highest scores from humans and even from other metrics (Cer *et al.*, 2010).

When we developed our own metric, we decided to make it a modified version of BLEU whose rankings of translations would (ideally) correlate even more highly with human rankings. Thus, our metric is called AMBER: "A Modified Bleu, Enhanced Ranking" metric. Some of the AMBER variants use an information source with a mild linguistic flavour – morphological knowledge about suffixes, roots and prefixes – but otherwise, the metric is based entirely on surface comparisons.

2 AMBER

Like BLEU, AMBER is composed of two parts: a score and a penalty.

$$AMBER = score \times penalty \quad (1)$$

To address weaknesses of BLEU described in the literature (Callison-Burch *et al.*, 2006; Lavie and Denkowski, 2009), we use more sophisticated formulae to compute the *score* and *penalty.*

2.1 Enhancing the *score*

First, we enrich the *score* part with geometric average of n-gram precisions (*AvgP*), F-measure derived from the arithmetic averages of precision and recall (*Fmean*), and arithmetic average of F-measure of precision and recall for each n-gram (*AvgF*). Let us define n-gram *precision* and *recall* as follows:

Proceedings of the 6th Workshop on Statistical Machine Translation, pages 71–77,
Edinburgh, Scotland, UK, July 30–31, 2011.

$$p(n) = \frac{\#ngrams(T \cap R)}{\#ngrams(T)} \quad (2)$$

$$r(n) = \frac{\#ngrams(T \cap R)}{\#ngrams(R)} \quad (3)$$

where T = translation, R = reference.

Then the geometric average of n-gram precisions *AvgP*, which is also the score part of the BLEU metric, is defined as:

$$AvgP(N) = \left(\prod_{n=1}^{N} p(n) \right)^{\frac{1}{N}} \quad (4)$$

The arithmetic averages for n-gram precision and recall are:

$$P(N) = \frac{1}{N} \sum_{n=1}^{N} p(n) \quad (5)$$

$$R(M) = \frac{1}{M} \sum_{n=1}^{M} r(n) \quad (6)$$

The F-measure that is derived from P(N) and R(M), (*Fmean*), is given by:

$$Fmean(N, M, \alpha) = \frac{P(N)R(M)}{\alpha P(N) + (1-\alpha)R(M)} \quad (7)$$

The arithmetic average of F-measure of precision and recall for each n-gram (*AvgF*) is given by:

$$AvgF(N, \alpha) = \frac{1}{N} \sum_{n=1}^{N} \frac{p(n)r(n)}{\alpha p(n) + (1-\alpha)r(n)} \quad (8)$$

The *score* is the weighted average of the three values: *AvgP, Fmean,* and *AvgF*.

$$\begin{aligned} score(N) &= \theta_1 \times AvgP(N) \\ &+ \theta_2 \times Fmean(N, M, \alpha) \\ &+ (1 - \theta_1 - \theta_2) \times AvgF(N, \alpha) \end{aligned} \quad (9)$$

The free parameters N, M, α, θ_1 and θ_2 were manually tuned on a dev set.

2.2 Various *penalties*

Instead of the original brevity penalty, we experimented with a product of various penalties:

$$penalty = \prod_{i=1}^{P} pen_i^{w_i} \quad (10)$$

where w_i is the weight of each penalty pen_i.

Strict brevity penalty (SBP): (Chiang *et al.*, 2008) proposed this penalty. Let t_i be the translation of input sentence i, and let r_i be its reference (or if there is more than one, the reference whose length in words $|r_i|$ is closest to length $|t_i|$). Set

$$SBP = \exp\left(1 - \frac{\sum_i |r_i|}{\sum_i \min\{|t_i|, |r_i|\}}\right) \quad (11)$$

Strict redundancy penalty (SRP): long sentences are preferred by recall. Since we rely on both recall and precision to compute the *score*, it is necessary to punish the sentences that are too long.

$$SRP = \exp\left(1 - \frac{\sum_i \max\{|t_i|, |r_i|\}}{\sum_i |r_i|}\right) \quad (12)$$

Character-based strict brevity penalty (CSBP) and **Character-based strict redundancy penalty (CSRP)** are defined similarly. The only difference with the above two penalties is that here, length is measured in characters.

Chunk penalty (CKP): the same penalty as in METEOR:

$$CKP = 1 - \gamma \times \left(\frac{\#chunks}{\#matches(word)} \right)^{\beta} \quad (13)$$

γ and β are free parameters. We do not compute the word alignment between the translation and reference; therefore, the number of chunks is computed as $\#chunks = \#matches(bigram) - \#matches(word)$. For example, in the following two-sentence translation (references not shown), let "m_i" stand for a matched word, "x" stand for zero, one or more unmatched words:

S1: $m_1 m_2$ x $m_3 m_4 m_5$ x m_6

S2: m_7 x $m_8 m_9$ x $m_{10} m_{11} m_{12}$ x m_{13}

If we consider only unigrams and bigrams, there are 13 matched words and 6 matched bigrams ($m_1 m_2$, $m_3 m_4$, $m_4 m_5$, $m_8 m_9$, $m_{10} m_{11}$, $m_{11} m_{12}$), so there are 13-6=7 chunks ($m_1 m_2$, $m_3 m_4 m_5$, m_6, m_7, $m_8 m_9$, $m_{10} m_{11} m_{12}$, m_{13}).

Continuity penalty (CTP): if all matched words are continuous, then $\frac{\#ngrams(T \cap R)}{\#(n-1)grams(T \cap R) - \#segment}$ equals 1. Example:

S3: $m_1 m_2 m_3 m_4 m_5 m_6$

S4: $m_7 m_8 m_9 m_{10} m_{11} m_{12} m_{13}$

There are 13 matched unigrams, and 11 matched bi-grams; we get 11/(13-2)=1. Therefore, a continuity penalty is computed as:

$$CTP = \exp\left(-\frac{1}{N-1}\sum_{n=2}^{N}\frac{\#ngrams(T \cap R)}{\#(n-1)grams(T \cap R) - \#segment}\right) \quad (14)$$

Short word difference penalty (SWDP): a good translation should have roughly the same number of stop words as the reference. To make AMBER more portable across all Indo-European languages, we use short words (those with fewer than 4 characters) to approximate the stop words.

$$SWDP = \exp(-\frac{|a-b|}{\#unigram(r)}) \quad (15)$$

where *a* and *b* are the number of short words in the translation and reference respectively.

Long word difference penalty (LWDP): is defined similarly to SWDP.

$$LWDP = \exp(-\frac{|c-d|}{\#unigram(r)}) \quad (15)$$

where *c* and *d* are the number of long words (those longer than 3 characters) in the translation and reference respectively.

Normalized Spearman's correlation penalty (NSCP): we adopt this from (Isozaki *et al*., 2010). This penalty evaluates similarity in word order between the translation and reference. We first determine word correspondences between the translation and reference; then, we rank words by their position in the sentences. Finally, we compute Spearman's correlation between the ranks of the n words common to the translation and reference.

$$\rho = 1 - \frac{\sum_i d_i^2}{(n+1)n(n-1)} \quad (16)$$

where d_i indicates the distance between the ranks of the *i*-th element. For example:

T: *Bob reading book likes*

R: *Bob likes reading book*

The rank vector of the reference is [1, 2, 3, 4], while the translation rank vector is [1, 3, 4, 2]. The Spearman's correlation score between these two vectors is $1-\frac{0+(3-2)^2+(4-3)^2+(2-4)^2}{(4+1)\cdot 4\cdot(4-1)}$=0.90.

In order to avoid negative values, we normalized the correlation score, obtaining the penalty NSCP:

$$NSCP = (1+\rho)/2 \quad (17)$$

Normalized Kendall's correlation penalty (NKCP): this is adopted from (Birch and Osborne, 2010) and (Isozaki *et al*., 2010). In the previous example, where the rank vector of the translation is [1, 3, 4, 2], there are $C_4^2 = 6$ pairs of integers. There are 4 increasing pairs: (1,3), (1,4), (1,2) and (3,4). Kendall's correlation is defined by:

$$\tau = 2\times\frac{\#increasing\ \ pairs}{\#all\ \ \ pairs} - 1 \quad (18)$$

Therefore, Kendall's correlation for the translation "*Bob reading book likes*" is $2\times 4/6 - 1$=0.33.

Again, to avoid negative values, we normalized the coefficient score, obtaining the penalty NKCP:

$$NKCP = (1+\tau)/2 \quad (19)$$

2.3 Term weighting

The original BLEU metric weights all n-grams equally; however, different n-grams have different amounts of information. We experimented with applying *tf-idf* to weight each n-gram according to its information value.

2.4 Four matching strategies

In the original BLEU metric, there is only one matching strategy: n-gram matching. In AMBER, we provide four matching strategies (the best AMBER variant used three of these):

1. N-gram matching: involved in computing *precision* and *recall.*
2. Fixed-gap n-gram: the size of the gap between words "word1 [] word2" is fixed; involved in computing *precision* only.
3. Flexible-gap n-gram: the size of the gap between words "word1 * word2" is flexible; involved in computing *precision* only.
4. Skip n-gram: as used ROUGE (Lin, 2004); involved in computing *precision* only.

2.5 Input preprocessing

The AMBER score can be computed with different types of preprocessing. When using more than one type, we computed the final score as an average over runs, one run per type (our default AMBER variant used three of the preprocessing types):

$$Final\_AMBER = \frac{1}{T}\sum_{t=1}^{T} AMBER(t)$$

We provide 8 types of possible text input:

0. Original - true-cased and untokenized.

1. Normalized - tokenized and lower-cased. (All variants 2-7 below also tokenized and lower-cased.)
2. "Stemmed" - each word only keeps its first 4 letters.
3. "Suffixed" - each word only keeps its last 4 letters.
4. Split type 1 - each longer-than-4-letter word is segmented into two sub-words, with one being the first 4 letters and the other the last 2 letters. If the word has 5 letters, the 4th letter appears twice: e.g., "gangs" becomes "gang" + "gs". If the word has more than 6 letters, the middle part is thrown away
5. Split type 2 - each word is segmented into fixed-length (4-letter) sub-word sequences, starting from the left.
6. Split type 3 - each word is segmented into prefix, root, and suffix. The list of English prefixes, roots, and suffixes used to split the word is from the Internet[1]; it is used to split words from all languages. Linguistic knowledge is applied here (but not in any other aspect of AMBER).
7. Long words only - small words (those with fewer than 4 letters) are removed.

3 Experiments

3.1 Experimental data

We evaluated AMBER on WMT data, using WMT 2008 all-to-English submissions as the dev set. Test sets include WMT 2009 all-to-English, WMT 2010 all-to-English and 2010 English-to-all submissions. **Table 1** summarizes the dev and test set statistics.

Set	Dev	Test1	Test2	Test3
Year	2008	2009	2010	2010
Lang.	xx-en	xx-en	xx-en	en-xx
#system	43	39	53	32
#sent-pair	7,861	13,912	14,212	13,165

Table 1: statistics of the dev and test sets.

[1] http://en.wikipedia.org/wiki/List_of_Greek_and_Latin_roots_in_English

3.2 Default settings

Before evaluation, we manually tuned all free parameters on the dev set to maximize the system-level correlation with human judgments and decided on the following default settings for AMBER:

1. The parameters in the formula
$$score(N) = \theta_1 \times AvgP(N) + \theta_2 \times Fmean(N, M, \alpha) + (1 - \theta_1 - \theta_2) \times AvgF(N, \alpha)$$
are set as N=4, M=1, α=0.9, θ_1 = 0.3 and θ_2 = 0.5.
2. All penalties are applied; the manually set penalty weights are shown in **Table 2**.
3. We took the average of runs over input text types 1, 4, and 6 (*i.e.* normalized text, split type 1 and split type 3).
4. In Chunk penalty (CKP), $\beta = 3$, and γ=0.1.
5. By default, *tf-idf* is not applied.
6. We used three matching strategies: n-gram, fixed-gap n-gram, and flexible-gap n-gram; they are equally weighted.

Name of penalty	Weight value
SBP	0.30
SRP	0.10
CSBP	0.15
CSRP	0.05
SWDP	0.10
LWDP	0.20
CKP	1.00
CTP	0.80
NSCP	0.50
NKCP	2.00

Table 2: Weight of each penalty

3.3 Evaluation metrics

We used Spearman's rank correlation coefficient to measure the correlation of AMBER with the human judgments of translation at the system level. The human judgment score we used is based on the "Rank" only, *i.e.*, how often the translations of the system were rated as better than the translations from other systems (Callison-Burch *et al.*, 2008). Thus, AMBER and the other metrics were evaluated on how well their rankings correlated with

the human ones. For the sentence level, we use consistency rate, *i.e.*, how consistent the ranking of sentence pairs is with the human judgments.

3.4 Results

All test results shown in this section are averaged over all three tests described in **3.1**. First, we compare AMBER with two of the most widely used metrics: original IBM BLEU and METEOR v1.0. **Table 3** gives the results; it shows both the version of AMBER with basic preprocessing, AMBER(1) (with tokenization and lowercasing) and the default version used as baseline for most of our experiments (AMBER(1,4,6)). Both versions of AMBER perform better than BLEU and METEOR on both system and sentence levels.

Metric		Dev	3 tests average	Δ tests
BLEU_ibm	sys	0.68	0.72	N/A
(baseline)	sent	0.37	0.40	N/A
METEOR	sys	0.80	0.80	+0.08
v1.0	sent	0.58	0.56	+0.17
AMBER(1)	sys	0.83	0.83	+0.11
(basic preproc.)	sent	0.61	0.58	+0.19
AMBER(1,4,6)	sys	0.84	**0.86**	**+0.14**
(default)	sent	0.62	**0.60**	**+0.20**

Table 3: Results of AMBER vs BLEU and METEOR

Second, as shown in **Table 4**, we evaluated the impact of different types of preprocessing, and some combinations of preprocessing (we do one run of evaluation for each type and average the results). From this table, we can see that splitting words into sub-words improves both system- and sentence-level correlation. Recall that input 6 preprocessing splits words according to a list of English prefixes, roots, and suffixes: AMBER(4,6) is the best variant. Although test 3 results, for target languages other than English, are not broken out separately in this table, they are as follows: input 1 yielded 0.8345 system-level correlation and 0.5848 sentence-level consistency, but input 6 yielded 0.8766 (+0.04 gain) and 0.5990 (+0.01) respectively. Thus, surprisingly, splitting non-English words up according to English morphology helps performance, perhaps because French, Spanish, German, and even Czech share some word roots with English. However, as indicated by the underlined results, if one wishes to avoid the use of any linguistic information, AMBER(4) performs almost as well as AMBER(4,6). The default setting, AMBER(1,4,6), doesn't perform quite as well as AMBER(4,6) or AMBER(4), but is quite reasonable.

Varying the preprocessing seems to have more impact than varying the other parameters we experimented with. In **Table 5**, "none+*tf-idf*" means we do one run without *tf-idf* and one run for "*tf-idf* only", and then average the scores. Here, applying *tf-idf* seems to benefit performance slightly.

Input		Dev	3 tests average	Δ tests
0	sys	0.84	0.79	N/A
(baseline)	sent	0.59	0.58	N/A
1	sys	0.83	0.83	+0.04
	sent	0.61	0.58	+0.00
2	sys	0.83	0.84	+0.05
	sent	0.61	0.59	+0.01
3	sys	0.83	0.84	+0.05
	sent	0.61	0.58	+0.00
4	sys	0.84	0.87	+0.08
	sent	0.62	0.60	+0.01
5	sys	0.82	0.86	+0.07
	sent	0.61	0.56	+0.01
6	sys	0.83	0.88	+0.09
	sent	0.62	0.60	+0.02
7	sys	0.34	0.56	-0.23
	sent	0.58	0.53	-0.05
1,4	sys	0.84	0.85	+0.07
	sent	0.62	0.60	+0.01
4,6	sys	0.83	**0.88**	**+0.09**
	sent	0.62	**0.60**	**+0.02**
1,4,6	sys	0.84	0.86	+0.07
	sent	0.62	0.60	+0.02

Table 4: Varying AMBER preprocessing (best linguistic = bold, best non-ling. = underline)

tf-idf		Dev	3 tests average	Δ tests
none	sys	0.84	0.86	N/A
(baseline)	sent	0.62	0.60	N/A
tf-idf	sys	0.81	**0.88**	**+0.02**
only	sent	0.62	0.61	+0.01
none+*tf-idf*	sys	0.82	0.87	+0.01
	sent	0.62	**0.61**	**+0.01**

Table 5: Effect of *tf-idf* on AMBER(1,4,6)

Table 6 shows what happens if you disable one penalty at a time (leaving the weights of the other penalties at their original values). The biggest system-level performance degradation occurs when LWDP is dropped, so this seems to be the most

useful penalty. On the other hand, dropping CKP, CSRP, and SRP may actually improve performance. Firm conclusions would require retuning of weights each time a penalty is dropped; this is future work.

Penalties		Dev	3 tests average	Δ tests
All (baseline)	sys	0.84	0.86	N/A
	sent	0.62	0.60	N/A
-SBP	sys	0.82	0.84	-0.02
	sent	0.62	0.60	-0.00
-SRP	sys	0.83	0.88	+0.01
	sent	0.62	0.60	+0.00
-CSBP	sys	0.84	0.85	-0.01
	sent	0.62	0.60	+0.00
-CSRP	sys	0.83	0.87	+0.01
	sent	0.62	0.60	-0.00
-SWDP	sys	0.84	0.86	-0.00
	sent	0.62	0.60	+0.00
-LWDP	sys	0.83	**0.83**	**-0.03**
	sent	0.62	**0.60**	**-0.00**
-CTP	sys	0.82	0.84	-0.02
	sent	0.62	0.60	-0.00
-CKP	sys	0.83	0.87	+0.01
	sent	0.62	0.60	-0.00
-NSCP	sys	0.83	0.86	-0.00
	sent	0.62	0.60	+0.00
-NKCP	sys	0.82	0.85	-0.01
	sent	0.62	0.60	+0.00

Table 6: Dropping penalties from AMBER(1,4,6) – biggest drops on test in bold

Matching		Dev	3 tests avg	Δ tests
n-gram + fxd-gap+ flx-gap (default)	sys	0.84	**0.86**	N/A
	sent	0.62	**0.60**	N/A
n-gram	sys	0.84	0.86	-0.00
	sent	0.62	0.60	-0.00
fxd-gap+ n-gram	sys	0.84	0.86	-0.00
	sent	0.62	0.60	-0.00
flx-gap+ n-gram	sys	0.83	0.86	-0.00
	sent	0.62	0.60	-0.00
skip+ n-gram	sys	0.83	0.85	-0.01
	sent	0.62	0.60	-0.00
All four matchings	sys	0.83	0.86	-0.01
	sent	0.62	0.60	0.00

Table 7: Varying matching strategy for AMBER(1,4,6)

Finally, we evaluated the effect of the matching strategy. According to the results shown in **Table 7**, our default strategy, which uses three of the four types of matching (n-grams, fixed-gap n-grams, and flexible-gap n-grams) is close to optimal; the use of skip n-grams (either by itself or in combination) may hurt performance at both system and sentence levels.

4 Conclusion

This paper describes AMBER, a new machine translation metric that is a modification of the widely used BLEU metric. We used more sophisticated formulae to compute the *score*, we developed several new *penalties* to match the human judgment, we tried different preprocessing types, we tried *tf-idf*, and we tried four n-gram matching strategies. The choice of preprocessing type seemed to have the biggest impact on performance. AMBER(4,6) had the best performance of any variant we tried. However, it has the disadvantage of using some light linguistic knowledge about English morphology (which, oddly, seems to be helpful for other languages too). A purist may prefer AMBER(1,4) or AMBER(4), which use no linguistic information and still match human judgment much more closely than either BLEU or METEOR. These variants of AMBER share BLEU's virtues: they are language-independent and can be computed quickly.

Of course, AMBER could incorporate more linguistic information: *e.g.*, we could use linguistically defined stop word lists in the SWDP and LWDP penalties, or use synonyms or paraphrasing in the n-gram matching.

AMBER can be thought of as a weighted combination of dozens of computationally cheap features based on word surface forms for evaluating MT quality. This paper has shown that combining such features can be a very effective strategy for attaining better correlation with human judgment. Here, the weights on the features were manually tuned; in future work, we plan to learn weights on features automatically. We also plan to redesign AMBER so that it becomes a metric that is highly suitable for tuning SMT systems.

References

S. Banerjee and A. Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of ACL Workshop on Intrinsic & Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

A. Birch and M. Osborne. 2010. LRscore for evaluating lexical and reordering quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 302–307.

C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz and J. Schroeder. 2008. Further Meta-Evaluation of Machine Translation. In *Proceedings of WMT*.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of EACL*.

D. Cer, D. Jurafsky and C. Manning. 2010. The Best Lexical Metric for Phrase-Based Statistical MT System Optimization. In *Proceedings of NAACL*.

Y. S. Chan and H. T. Ng. 2008. MAXSIM: A maximum similarity metric for machine translation evaluation. In *Proceedings of ACL*.

D. Chiang, S. DeNeefe, Y. S. Chan, and H. T. Ng. 2008. Decomposability of translation metrics for improved evaluation and efficient algorithms. In *Proceedings of EMNLP*, pages 610–619.

M. Denkowski and A. Lavie. 2010. Meteor-next and the meteor paraphrase tables: Improved evaluation support for five target languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 314–317.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*.

Y. He, J. Du, A. Way, and J. van Genabith. 2010. The DCU dependency-based metric in WMT-MetricsMATR 2010. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 324–328.

H. Isozaki, T. Hirao, K. Duh, K. Sudoh, H. Tsukada. 2010. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of EMNLP*.

A. Lavie and M. J. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23.

C.-Y. Lin. 2004. ROUGE: a Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, Barcelona, Spain.

C. Liu, D. Dahlmeier, and H. T. Ng. 2010. Tesla: Translation evaluation of sentences with linear-programming-based analysis. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 329–334.

S. Pado, M. Galley, D. Jurafsky, and C.D. Manning. 2009. Robust machine translation evaluation with entailment features. In *Proceedings of ACL-IJCNLP*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*.

M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, Adequacy, or HTER? Exploring Different Human Judgments with a Tunable MT Metric. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, Athens, Greece.