BLANC: Learning Evaluation Metrics for MT

Lucian Vlad Lita and **Monica Rogati** and **Alon Lavie**
Carnegie Mellon University
{llita,mrogati,alavie}@cs.cmu.edu

Abstract

We introduce BLANC, a family of dynamic, trainable evaluation metrics for machine translation. Flexible, parametrized models can be learned from past data and automatically optimized to correlate well with human judgments for different criteria (e.g. adequacy, fluency) using different correlation measures. Towards this end, we discuss ACS (all common skip-ngrams), a practical algorithm with trainable parameters that estimates reference-candidate translation overlap by computing a weighted sum of all common skip-ngrams in polynomial time. We show that the BLEU and ROUGE metric families are special cases of BLANC, and we compare correlations with human judgments across these three metric families. We analyze the algorithmic complexity of ACS and argue that it is more powerful in modeling both local meaning and sentence-level structure, while offering the same practicality as the established algorithms it generalizes.

1 Introduction

Although recent MT evaluation methods show promising correlations to human judgments in terms of adequacy and fluency, there is still considerable room for improvement (Culy and Riehemann, 2003). Most of these studies have been performed at a system level and have not investigated metric robustness at a lower granularity. Moreover, even though the emphasis on adequacy vs. fluency is application-dependent, automatic evaluation metrics do not distinguish between the need to optimize correlation with regard to one or the other.

Machine translation automatic evaluation metrics face two important challenges: the lack of powerful features to capture both sentence level structure and local meaning, and the difficulty of designing good functions for combining these features into meaningful quality estimation algorithms.

In this paper, we introduce BLANC[1], an automatic MT evaluation metric family that is a generalization of popular and successful metric families currently used in the MT community (BLEU, ROUGE, F-measure etc.). We describe an efficient, polynomial-time algorithm for BLANC, and show how it can be optimized to target adequacy, fluency or any other criterion. We compare our metric's performance with traditional and recent automatic evaluation metrics. We also describe the parameter conditions under which BLANC can emulate them.

Throughout the remainder of this paper, we distinguish between two components of automatic MT evaluation: the *statistics* computed on candidate and reference translations and the *function* used in defining evaluation metrics and generating translation scores. Commonly used statistics include bag-of-words overlap, edit distance, longest common subsequence, ngram overlap, and skip-bigram overlap. Preferred functions are various combinations of precision and recall (Soricut and Brill, 2004), including

[1] Since existing evaluation metrics (e.g. BLEU, ROUGE) are special cases of our metric family, it is only natural to name it Broad Learning and Adaptation for Numeric Criteria (BLANC) – white light contains light of all frequencies

weighted precision and F-measures (Van-Rijsbergen, 1979).

BLANC implements a practical algorithm with learnable parameters for automatic MT evaluation which estimates the reference-candidate translation overlap by computing a weighted sum of common subsequences (also known as skip-ngrams). Common skip-ngrams are sequences of words in their sentence order that are found both in the reference and candidate translations. By generalizing and separating the overlap statistics from the function used to combine them, and by identifying the latter as a learnable component, BLANC subsumes the ngram based evaluation metrics as special cases and can better reflect the need of end applications for adequacy/fluency tradeoffs .

1.1 Related Work

Initial work in evaluating translation quality focused on edit distance-based metrics (Su et al., 1992; Akiba et al., 2001). In the MT context, edit distance (Levenshtein, 1965) represents the amount of word insertions, deletions and substitutions necessary to transform a candidate translation into a reference translation. Another evaluation metric based on edit distance is the *Word Error Rate* (Niessen et al., 2000) which computes the normalized edit distance. BLEU is a weighted precision evaluation metric introduced by IBM (Papineni et al., 2001). BLEU and its extensions/variants (e.g. NIST (Doddington, 2002)) have become de-facto standards in the MT community and are consistently being used for system optimization and tuning. These methods rely on local features and do not explicitly capture sentence-level features, although implicitly longer n-gram matches are rewarded in BLEU. The General Text Matcher (GTM) (Turian et al., 2003) is another MT evaluation method that rewards longer ngrams instead of assigning them equal weight.

(Lin and Och, 2004) recently proposed a set of metrics (ROUGE) for MT evaluation. ROUGE-L is a longest common subsequence (LCS) based automatic evaluation metric for MT. The intuition behind it is that long common subsequences reflect a large overlap between a candidate translation and a reference translation. ROUGE-W is also based on LCS, but assigns higher weights to sequences that have fewer gaps. However, these metrics still do not distinguish among translations with the same LCS but different number of shorter sized subsequences, also indicative of overlap. ROUGE-S attempts to correct this problem by combining the precision/recall of skip-*bigrams* of the reference and candidate translations. However, by using skip-ngrams with n>=2, we might be able to capture more information encoded in the higher level sentence structure. With BLANC, we propose a way to exploit local contiguity in a manner similar to BLEU and also higher level structure similar to ROUGE type metrics.

2 Approach

We have designed an algorithm that can perform a full overlap search over variable-size, non-contiguous word sequences (skip-ngrams) efficiently. At first glance, in order to perform this search, one has to first exhaustively generate all skip-ngrams in the candidate and reference segments and then assess the overlap. This approach is highly prohibitive since the number of possible sequences is exponential in the number of words in the sentence. Our algorithm – ACS (all common skip-ngrams) – directly constructs the set of overlapping skip-ngrams through incremental composition of word-level matches. With ACS, we can reduce computation complexity to a fifth degree polynomial in the number of words.

Through the ACS algorithm, BLANC is not limited only to counting skip-ngram overlap: the contribution of different skip-ngrams to the overall score is based on a set of features. ACS computes the overlap between two segments of text and also allows local and global features to be computed during the overlap search. These local and global features are subsequently used to train evaluation models within the BLANC family. We introduce below several simple skip-ngram-based features and show that special-case parameter settings for these features emulate the computation of existing ngram-based metrics. In order to define the relative significance of a particular skip-ngram found by the ACS algorithm, we employ an exponential model for feature integration.

2.1 Weighted Skip-Ngrams

We define *skip-ngrams* as sequences of n words taken in sentence order allowing for arbitrary gaps. In algorithms literature skip-ngrams are equivalent to *subsequences*. As special cases, skip-ngrams with n=2 are referred to as skip-bigrams and skip-ngrams with no gaps between the words are simply $ngrams$. A sentence S of size $|S|$ has $C(|S|, n) = \frac{|S|!}{(|S|-n)!n!}$ skip-ngrams.

For example, the sentence "*To be or not to be*" has $C(6, 2) = 15$ corresponding skip-bigrams including "*be or*", "*to to*", and three occurrences of "*to be*". It also has $C(6, 4) = 15$ corresponding skip-4grams ($n = 4$) including "to be to be" and "to or not to".

Consider the following sample reference and candidate translations:

R_0:	machine translated text is evaluated automatically
K_1:	machine translated stories are chosen automatically
K_2:	machine and human together can forge a friendship that cannot be translated into words automatically
K_3:	machine code is being translated automatically

The skip-ngram "*machine translated automatically*" appears in both the reference R_0 and all candidate translations. Arguably, a skip-bigram that contains few gaps is likely to capture local structure or meaning. At the same time, skip-ngrams spread across a sentence are also very useful since they may capture part of the high level sentence structure.

We define a **weighting** feature function for skip-ngrams that estimates how likely they are to capture local meaning and sentence structure. The weighting function φ for a skip-ngram $w_1 .. w_n$ is defined as:

$$\varphi(w_1..w_n) = e^{-\alpha \cdot G(w_1..w_n)} \qquad (1)$$

where $\alpha \geq 0$ is a decay parameter and G($w_1..w_n$) measures the overall gap of the skip-ngram $w_1..w_n$ in a specific sentence. This overall skip-ngram weight can be decomposed into the weights of its constituent skip-bigrams:

$$\begin{aligned}\varphi(w_1..w_n) &= e^{-\alpha \cdot G(w_1,..,w_n)} && (2)\\ &= e^{-\alpha \cdot \sum_{i=1}^{n-1} G(w_i, w_{i+1})} \\ &= \prod_{i=1}^{n-1} \varphi(w_i\ w_{i+1}) && (3)\end{aligned}$$

In equation 3, $\varphi(w_i\ w_{i+1})$ is the number of words between w_i and w_{i+1} in the sentence. In the example above, the skip-ngram "machine translated automatically" has weight $e^{-3\alpha}$ for sentence K_1 and weight $e^{-12\alpha} = 1$ for sentence K_2.

In our initial experiments the gap G has been expressed as a linear function, but different families of functions can be explored and their corresponding parameters learned. The parameter α dictates the behavior of the weighting function. When $\alpha = 0$, φ equals $e^0 = 1$, rendering gap sizes irrelevant. In this case, skip-ngrams are given the same weight as contiguous ngrams. When α is very large, φ approaches 0 if there are any gaps in the skip-ngram and is 1 if there are no gaps. This setting has the effect of considering only contiguous ngrams and discarding all skip-ngrams with gaps.

In the above example, although the skip-ngram "machine translated automatically" has the same cumulative gap in both in K_1 and K_3, the occurrence in K_1 has is a gap distribution that more closely reflects that of the reference skip-ngram in R_0. To model gap distribution differences between two occurrences of a skip-ngram, we define a piece-wise distance function δ_{XY} between two sentences x and y. For two successive words in the skip-ngram, the distance function is defined as:

$$\delta_{XY}(w_1 w_2) = e^{-\beta \cdot |G_X(w_1,w_2) - G_Y(w_1,w_2)|} \qquad (4)$$

where $\beta \geq 0$ is a decay parameter. Intuitively, the β parameter is used to reward better aligned skip-ngrams. Similar to the φ function, the overall δ_{XY} distance between two occurrences of a skip-ngram with $n > 1$ is:

$$\delta_{XY}(w_1..w_n) = \prod_{i=1}^{n-1} \delta_{XY}(w_i w_{i+1}) \qquad (5)$$

Note that equation 5 takes into account pairs of skip-ngrams skip in different places by summing over piecewise differences. Finally, using an exponential model, we assign an overall score to the matched skip-ngram. The skip-ngram scoring function S_{xy} allows independent features to be incorporated into the overall score:

$$\begin{aligned}S_{xy}(w_i..w_k) = \varphi(w_i..w_k) \cdot \delta_{xy}(w_i..w_k) \\ \cdot e^{\lambda_1 f_1(w_i..w_k)} \cdot ... \cdot e^{\lambda_h f_h(w_i..w_k)} \qquad (6)\end{aligned}$$

where features $f_1..f_h$ can be functions based on the syntax, semantics, lexical or morphological aspects of the skip-ngram. Note that different models for combining skip-ngram features can be used in conjunction with ACS.

2.2 Multiple References

In BLANC we incorporate multiple references in a manner similar to the ROUGE metric family. We compute the precision and recall of each size skip-ngrams for individual references. Based on these we combine the maximum precision and maximum recall of the candidate translation obtained using all reference translations and use them to compute an aggregate F-measure.

The F-measure parameter β_F is modeled by BLANC. In our experiments we optimized β_F individually for fluency and adequacy.

2.3 The ACS Algorithm

We present a practical algorithm for extracting *All Common Skip-ngrams* (ACS) of any size that appear in the candidate and reference translations. For clarity purposes, we present the ACS algorithm as it relates to the MT problem: find all common skip-ngrams (ACS) of any size in two sentences X and Y:

$$\begin{aligned} & wSKIP \leftarrow Acs(\delta, \varphi, X, Y) \\ & = \{wSKIP_1..wSKIP_{min(|X|,|Y|)}\} \end{aligned} \tag{7}$$

where $wSkip_n$ is the set of all skip-ngrams of size n and is defined as:

$$\begin{aligned} wSKIP_n = \{``w_1..w_n" \mid w_i \in X, w_i \in Y, \forall i \in [1..n] \\ \text{and } w_i \prec w_j, \forall i < j \in [1..n]\} \end{aligned}$$

Given two sentences X and Y we observe a *match* (w, x, y) if word w is found in sentence X at index x and in sentence Y at index y:

$$\begin{aligned} (w, x, y) \equiv \{0 \leq x \leq |X|, 0 \leq y \leq |Y|, \\ w \in V, \textit{and } X[x] = Y[y] = w\} \end{aligned} \tag{8}$$

where V is the vocabulary with a finite set of words. In the following subsections, we present the following steps in the ACS algorithm:

1. *identify all matches* – find matches and generate corresponding nodes in the dependency graph
2. *generate dependencies* – construct edges according to pairwise match dependencies
3. *propagate common subsequences* – count all common skip-ngrams using corresponding weights and distances

In the following sections we use the following example to illustrate the intermediate steps of ACS.

X. "to be or not to be"
Y. "to exist or not be"

2.3.1 Step 1: Identify All Matches

In this step we identify all word matches (w, x, y) in sentences X and Y. Using the example above, the intermediate inputs and outputs of this step are:

Input: X. "to be or not to be"
Y. "to exist or not be"
Output: (*to*,1,1); (*to*,5,1); (*or*,3,3); (*be*,2,5); …

For each match we create a corresponding node N in a dependency graph. With each node we associate the actual word matched and its corresponding index positions in both sentences.

2.3.2 Step 2: Generate Dependencies

A dependency $N_1 \rightarrow N_2$ occurs when the two corresponding matches (w_1, x_1, y_1) and (w_2, x_2, y_2) can form a valid common skip-bigram: i.e. when $x_1 < x_2$ and $y_1 < y_2$. Note that the matches can cover identical words, but their indices cannot be the same ($x_1 \neq x_2$ and $y_1 \neq y_2$) since a skip-bigram requires two different word matches.

In order to facilitate the generation of all common subsequences, the graph is populated with the appropriate dependency edges:

```
for each node N in DAG
    for each node M≠N in DAG
        if N(x)≤M(x) and N(y)≤M(y)
            create edge E: N→M
                compute $\delta_{XY}$(E)
                compute $\varphi$(E)
```

This step incorporates the concepts of skip-ngram weight and distance into the graph. With each edge $E : N_1 \rightarrow N_2$ we associate step-wise weight and distance information for the corresponding skip-bigram formed by matches (w_1, x_1, y_1) and (w_2, x_2, y_2).

Note that rather than counting all skip-ngrams, which would be exponential in the worst case scenario, we only construct a structure of match dependencies (i.e. skip-bigrams). As in dynamic programming, in order to avoid exponential complexity, we compute individual skip-ngram scores only once.

2.3.3 Step 3: Propagate Common Subsequences

In this last step, the ACS algorithm *counts* all common skip-ngrams using corresponding weights and distances. In the general case, this step is equivalent measuring the overlap of the two sentences X and Y. As a special case, if no features are used, the ACS algorithm is equivalent to counting the number of common skip-ngrams regardless of gap sizes.

```
// depth first search (DFS)
for each node N in DAG
    compute node N's depth

// initialize skip-ngram counts
for each node N in DAG
    v_N[1] ← 1
    for i=2 to LCS(X,Y)
        v_N[i] = 0

// compute ngram counts
for d=1 to MAXDEPTH
    for each node N of depth d in DAG
        for each edge E: N→M
            for i=2 to d
                v_M[i] += $S_{xy}$($\delta$(E), $\varphi$(E), v_N[i-1])
```

After algorithm ACS is run, the number of skip-ngrams (weighted skip-ngram score) of size k is simply the sum of the number of skip-ngrams of size k ending in each node N's corresponding match:

$$wSKIP_k = \sum_{N_i \in DAG} v_{N_i}[k] \tag{9}$$

2.3.4 ACS Complexity and Feasibility

In the worst case scenario, both sentences X and Y are composed of exactly the same repeated word: X = "*w w w w ..* " and Y = "*w w w w ..*". We let $m = |X|$ and $n = |Y|$. In this case, the number of matches is $M = n \cdot m$. Therefore, Step 1 has worst case time and space complexity of $O(m \cdot n)$. However, empirical data suggest that there are far fewer matches than in the worst-case scenario and the actual space requirements are drastically reduced. Even in the worst-case scenario, if we assume the average sentences is fewer than 100 words, the number of nodes in the DAG would only be $10,000$. Step 2 of the algorithm consists of creating edges in the dependency graph. In the worst case scenario, the number of directed edges is $O(M^2)$ and furthermore if the sentences are uniformly composed of the same repeated word as seen above, the worst-case time and space complexity is $m(m+1)/2 \cdot n(n+1)/2 = O(m^2n^2)$. In Step 3 of the algorithm, the DFS complexity for computing of node depths is $O(M)$ and the complexity of $LCS(X,Y)$ is $O(m \cdot n)$. The dominant step is the propagation of common subsequences (skip-ngram counts). Let l be the size of the LCS. The upper bound on the size of the longest common subsequence is $min(|X|, |Y|) = min(m, n)$. In the worst case scenario, for each node we propagate l count values (the size of vector v) to all other nodes in the DAG. Therefore, the time complexity for Step 3 is $O(M^2 \cdot l) = O(m^2n^2l)$ (fifth degree polynomial).

3 BLANC as a Generalization of BLEU and ROUGE

Due to its parametric nature, the All Common Subsequences algorithm can emulate the ngram computation of several popular MT evaluation metrics. The weighting function φ allows skip-ngrams with different gap sizes to be assigned different weights. Parameter α controls the shape of the weighting function.

In one extreme scenario, if we allow α to take very large values, the net effect is that all contiguous ngrams of any size will have corresponding weights of $e^0 = 1$ while all other skip-ngrams will have weights that are zero. In this case, the distance function will only apply to contiguous ngrams which have the same size and no gaps. Therefore, the distance function will also be 1. The overall result is that the ACS algorithm collects contiguous common ngram counts for all ngram sizes. This is equivalent to computing the ngram overlap between two sentences, which is equivalent to the ngram computation performed BLEU metric. In addition to computing ngram overlap, BLEU incorporates a thresholding (clipping) on ngram counts based on reference translations, as well as a brevity penalty which makes sure the machine-produced translations are not too short. In BLANC, this is replaced by standard F-measure, which research (Turian et al., 2003) has shown it can be used successfully in MT evaluation.

Another scenario consists of setting the α and β parameters to 0. In this case, all skip-ngrams are assigned the same weight value of 1 and skip-ngram matches are also assigned the same distance value of 1 regardless of gap sizes and differences in gap sizes.

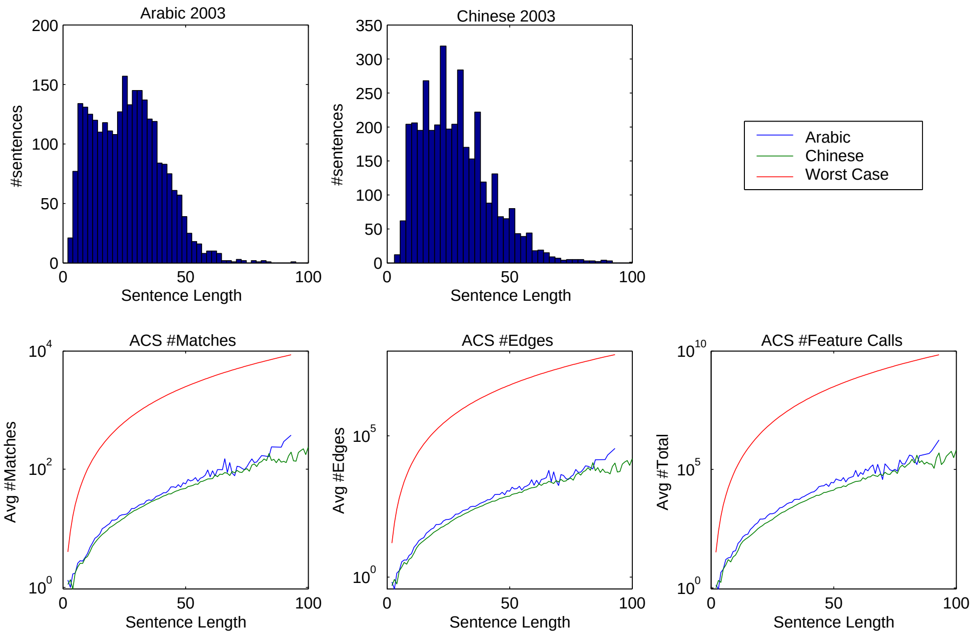

Figure 1: Empirical and theoretical behavior of ACS on 2003 machine translation evaluation data (semilog scale).

This renders all skip-ngrams equivalent and the ACS algorithm is reduced to counting the *skip-ngram overlap* between two sentences. Using these counts, precision and recall-based metrics such as the F-measure can be computed. If we let the α and β parameters to be zero, disregard redundant matches, and compute the ACS only for skip-ngrams of size 2, the ACS algorithm is equivalent to the ROUGE-S metric (Lin and Och, 2004). This case represents a specific parameter setting in the ACS skip-ngram computation.

The longest common subsequence statistic has also been successfully used for automatic machine translation evaluation in the ROUGE-L (Lin and Och, 2004) algorithm. In BLANC, if we set both α and β parameters to zero, the net result is a set of skip-bigram (common subsequence) overlap counts for all skip-bigram sizes. Although dynamic programming or suffix trees can be used to compute the LCS much faster, under this parameter setting the ACS algorithm can also produce the longest common subsequence:

$$LCS(X,Y) \leftarrow \underset{k}{\operatorname{argmax}}\ ACS(wSKIP_k) > 0$$

where $Acs(wSKIP_k)$ is the number of common skip-ngrams (common subsequences) produced by the ACS algorithm.

ROUGE-W (Lin and Och, 2004) relies on a weighted version of the longest common subsequence, under which longer contiguous subsequences are assigned a higher weight than subsequences that incorporate gaps. ROUGE-W uses the polynomial function x^a in the weighted LCS computation. This setting can also be simulated by BLANC by adjusting the parameters α to reward tighter skip-ngrams and β to assign a very high score to similar size gaps. Intuitively, α is used to reward skip-ngrams that have smaller gaps, while β is used to reward better aligned skip-ngram overlap.

4 Scalability & Data Exploration

In Figure 1 we show theoretical and empirical practical behavior for the ACS algorithm on the 2003 TIDES machine translation evaluation data for Arabic and Chinese. Sentence length distribution is somewhat similar for the two languages – only a very small amount of text segments have more than 50 tokens. We show the ACS graph size in the worst case scenario, and the empirical average number of matches for both languages as a function of sentence length. We also show (on a log scale) the upper bound on time/space complexity in terms of total number of feature computations. Even though the worst-case scenario is tractable (polynomial), the empirical amount of computation is considerably smaller in the form of polynomials of lower degree. In Figure 1, sentence length is the average between reference and candidate lengths.

Finally, we also show the total number of feature computations involved in performing a full overlap search and computing a numeric score for the reference-candidate translation pair. We have experimented with the ACS algorithm using a worst-case scenario where all words are exactly the same for a fifty words reference translation and candidate translation. In practice when considering real sentences the number of matches is very small. In this setting, the algorithm takes less than two seconds on a low-end desktop system when working on the worst case scenario, and less then a second for all candidate-reference pairs in the TIDES 2003 dataset. This result renders the ACS algorithm very practical for automatic MT evaluation.

5 Experiments & Results

In the dynamic metric BLANC, we have implemented the ACS algorithm using several parameters including the aggregate gap size α, the displacement feature β, a parameter for regulating skip-ngram size contribution, and the F-measure β_F parameter.

Until recently, most experiments that evaluate automatic metrics correlation to human judgments have been performed at a system level. In such experiments, human judgments are aggregated across sentences for each MT system and compared to aggregate scores for automatic metrics. While high scoring metrics in this setting are useful for understanding relative system performance, not all of them are robust enough for evaluating the quality of machine translation output at a lower granularity. Sentence-level translation quality estimation is very useful when MT is used as a component in a pipeline of text-processing applications (e.g. question answering). The fact that current automatic MT evaluation metrics including BLANC do not correlate well with human judgments at the sentence level, does not mean we should ignore this need and focus only on system level evaluation. On the contrary, further research is required to improve these metrics. Due to its trainable nature, and by allowing additional features to be incorporated into its model, BLANC has the potential to address this issue.

For comparison purposes with previous literature, we have also performed experiments at system level for Arabic. The datasets used consist of the MT translation outputs from all systems available through the Tides 2003 evaluation (663 sentences) for training and Tides 2004 evaluation (1353 sentences) for testing.

We compare (Table 1) the performance of BLANC on Arabic translation output with the performance of more established evaluation metrics: BLEU and NIST, and also with more recent metrics: ROUGE-L and ROUGE-S (using an unlimited size skip window), which have been shown to correlate well with human judgments at system level – as confirmed by our results. We have performed experiments in which case information is preserved as well as experiments that ignore case information. Since the results are very similar, we only show here experiments under the former condition. In order to maintain consistency, when using any metric we apply the same preprocessing provided by the MTEval script. When computing the correlation between metrics and human judgments, we only keep strictly positive scores. While this is not fully equivalent to BLEU smoothing, it partially mitigates the same problem of zero count ngrams for short sentences. In future work we plan to implement smoothing for all metrics, including BLANC.

We train BLANC separately for adequacy and fluency, as well as for system level and segment level correlation with human judgments. The BLANC parameters are currently trained using a simple hill-climbing procedure and using several starting points in order to decrease the chance of reaching a local maximum.

BLANC proves to be robust across criteria and granularity levels. As expected, different parameter values of BLANC optimize different criteria (e.g. adequacy and fluency). We have observed that training BLANC for adequacy results in more bias towards recall (β_F=3) compared to training it for fluency (β_F=2). This confirms our intuition that a dynamic, parametric metric is justified for automatic evaluation.

6 Conclusions & Future Work

Most of the established MT evaluation methods are static functions according to which automatic evaluation scores are computed. In this paper, we have laid the foundation for a more flexible, parametric ap-

Tides 2003 Arabic

	System Level		**Segment Level**	
Method	Adequacy	Fluency	Adequacy	Fluency
BLEU	0.950	0.934	0.382	0.286
NIST	0.962	0.939	0.439	0.304
ROUGE-L	0.974	0.926	0.440	0.328
ROUGE-S	0.949	0.935	0.360	0.328
BLANC	**0.988**	**0.979**	**0.492**	**0.391**

Tides 2004 Arabic

	System Level		**Segment Level**	
Method	Adequacy	Fluency	Adequacy	Fluency
BLEU	0.978	0.994	0.446	0.337
NIST	**0.987**	0.952	0.529	0.358
ROUGE-L	0.981	0.985	0.538	0.412
ROUGE-S	0.937	0.980	0.367	0.408
BLANC	0.982	**0.994**	**0.565**	**0.438**

Table 1: Pearson correlation of several metrics with human judgments at system level and segment level for fluency and adequacy.

proach that can be trained using existing MT data and that can be optimized for highest agreement with human assessors, for different criteria.

We have introduced ACS, a practical algorithm with learnable parameters for automatic MT evaluation and showed that ngram computation of popular evaluation methods can be emulated through different parameters by ACS. We have computed time and space bounds for the ACS algorithm and argued that while it is more powerful in modeling local and sentence structure, it offers the same practicality as established algorithms.

In our experiments, we trained and tested BLANC on data from consecutive years, and therefore tailored the metric for two different operating points in MT system performance. In this paper we show that BLANC correlates well with human performance when trained on previous year data for both sentence and system level.

In the future, we plan to investigate the stability and performance of BLANC and also apply it to automatic summarization evaluation. We plan to optimize the BLANC parameters for different criteria in addition to incorporating syntactic and semantic features (e.g. ngrams, word classes, part-of-speech).

In previous sections we have defined simple distance functions. More complex functions can also be incorporated in ACS. Skip-ngrams in the candidate sentence might be rewarded if they contain fewer gaps in the candidate sentence and penalized if they contain more. Different distance functions could also be used in ACS, including functions based on surface-form features and part-of-speech features.

Looking beyond the BLANC metric, this paper makes the case for the need to shift to trained, dynamic evaluation metrics which can adapt to individual optimization criteria and correlation functions.

We plan to make available an implementation of BLANC at *http://www.cs.cmu.edu/∼llita/blanc*.

References

Y. Akiba, K. Iamamurfa, and E. Sumita. 2001. Using multiple edit distances to automatically rank machine translation output. *MT Summit VIII*.

C. Culy and S.Z. Riehemann. 2003. The limits of n-gram translation evaluation metrics. *Machine Translation Summit IX*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *Human Language Technology Conference (HLT)*.

V.I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*.

C.Y. Lin and F.J. Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip bigram statistics. *ACL*.

S. Niessen, F.J. Och, G. Leusch, and H. Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. *LREC*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. *IBM Research Report*.

R. Soricut and E. Brill. 2004. A unified framework for automatic evaluation using n-gram co-occurence statistics. *ACL*.

K.Y. Su, M.W. Wu, and J.S. Chang. 1992. A new quantitative quality measure for machine translation systems. *COLING*.

J.P. Turian, L. Shen, and I.D. Melamed. 2003. Evaluation of machine translation and its evaluation. *MT Summit IX*.

C.J. Van-Rijsbergen. 1979. Information retrieval.