A Customizable MT Evaluation Metric for Assessing Adequacy

Machine Translation Term Project

Jason Adams

May 7, 2007

Abstract

This project describes a customizable MT evaluation metric that provides system-dependent scores for the purposes of tuning an MT system. The features presented focus on assessing adequacy over fluency. Rather than simply examining features, this project frames the MT evaluation task as a classification question to determine whether a given sentence was produced by a human or a machine. Support Vector Machines are applied to the classification task and their confidence score forms the basis of the evaluation score. In addition, syntactic features are introduced via headword chain precision and recall features derived from dependency parses. Results are presented showing higher sentence-level correlation with adequacy than BLEU as well as an analysis of which features contribute most to assessing adequacy.

1 Introduction

1.1 Background

Evaluating machine translation system output is a difficult but important task for the advancement of machine translation research. The need to monitor incremental changes to MT systems has been one of the largest motivations for creating automatic metrics such as BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005). Metrics tend to assign a single score that attempts to capture the quality of the MT output in a consistent way that can be compared to other results. Often, it is helpful to not only be able to evaluate a large quantity of MT output, but output at the sentence or segment level, for which BLEU is not well suited[1]. Other metrics, such as METEOR, address this by putting more emphasis on recall and synonymy than on exact-match precision. BLEU has also been criticized by a number of researchers for the following flaws:

- it admits a large amount of variation for equally scored hypotheses (Callison-Burch et al., 2006)
- it favors "local word choice over global accuracy" (Charniak et al., 2003), thereby favoring certain types of MT systems over others (Callison-Burch et al., 2006)
- it does not correlate well with human judgments (Turian et al., 2003) contrary to claims made by (Papineni et al., 2002) and (Doddington, 2002)
- it lacks flexibility for tuning to a particular MT task (Kuleska and Shieber, 2004; Och et al., 2003)

These shortcomings with BLEU motivate the need for a better way of measuring the quality of MT output.

A different approach to MT evaluation is to frame the task in terms of a classification problem (Kuleska and Shieber, 2004; Blatz et al., 2003). The challenge then becomes distinguishing between whether a sentence is the output of an MT system or a human translator. This project builds on the work by (Kuleska and Shieber, 2004), which used a support vector machine as a learning mechanism for classifying a hypothesis translation as coming from a human or a machine. In that experiment, the feature space was constructed by comparing the hypothesis against all the reference translations and calculating feature values (see Table 1). Absent in this list of features is recall, which is more highly correlated with human judgments than precision (Lavie et al., 2004). The absence of a measure for recall is another criticism of BLEU (Banerjee and Lavie, 2005). Another learning approach to MT evaluation that addresses the lack of flexibility in BLEU is BLANC (Lita et al., 2005). BLANC trains parameters using *all common skip-grams* and computes the overlap with reference translations.

[1]BLEU uses the geometric mean for n-gram precision, so if higher order n-grams do not occur in a given sentence, the BLEU score is zero.

A challenge for the development of MT evaluation systems is the human translation effort in creating the reference translations. One approach for incorporating paraphrases into MT evaluation by (Russo-Lassner et al., 2005), which increases the number of reference translations artificially, describes a large number of features that may be helpful to the classification task (see Table 2). Related work on using paraphrases for MT evaluation has been done by introducing syntax and aligning the reference translations (Pang et al., 2003). More relevant to this project, however, is the use of syntactic information as features for training the learning mechanism. Incorporating this information into an evaluation metric has been shown to improve correlation with human judgments at the sentence and corpus level (Liu and Gildea, 2005).

1. n-gram precisions (similar to BLEU)
2. ratio of hypothesis length to reference length
3. word error rate as the minimum edit distance between hypothesis and reference
4. word error rate independent of position

Table 1: Feature space considered by (Kuleska and Shieber, 2004).

1.2 Properties of Evaluation Metrics

There are many desirable features an MT evaluation metric should possess. The background information in § 1.1 describes some of the shortcomings of current metrics and motivates the need for further research into finding the most valuable features for efficient, extensible MT evaluation. Ideally, MT researchers would like a metric that measures different aspects of MT quality. Toward that end, this project evaluates adequacy in machine translation output using support vector machines. Adequacy can be defined as a measure of the amount of meaning preserved in the target language after translation from the source language. A completely adequate translation would convey all of the meaning and nuances intended in the source language. Solving the evaluation of adequacy problem would involve solving the machine translation problem, since being able to tell if all information in the source language was conveyed through the target language could just be reversed to generate the target language output. For the purposes of MT evaluation, therefore, the measure of adequacy we are computing is the amount of meaning in the reference translation(s) that is preserved in the MT output.

Evaluation metrics can give two types of scores. The first type is a *system-independent score* given by common metrics such as BLEU, NIST, or METEOR, where the scores for a given language are not dependent on a single system and

1. Primitive Features:
    (a) stemmed words co-occurrence
    (b) noun phrase matching
    (c) matching words that appear in the same WordNet synset
    (d) matching verbs that share the same semantic class
    (e) matching proper names based on POS tags
2. Composite Features
    (a) matching pairs of primitive features that appear in the same relative order
    (b) matching pairs of primitive features that occur within a window of 2-5 words
3. BLEU score
4. Translation Error Rate (TER)
5. METEOR score

Table 2: Features considered by (Russo-Lassner et al., 2005).

can be compared against the scores of other systems for the same language pair. Another type of score, while not as valuable for comparing separate systems, is a *system-dependent score*. The purpose of a system-dependent score is to assist in tuning the system. For such a metric to be beneficial, it should correlate well with human judgments and allow for nearly complete customization of which parts of the MT output are being evaluated. System-independent metrics offer some customizability. For example, BLEU and NIST can look at different numbers of n-grams, while METEOR allows for various stemming modules. However, a large number of the features examined by these metrics cannot be modified, and doing so would damage their ability to offer system-independent scores. A good system-dependent metric does not suffer from this weakness.

1.3 Evaluating MT Evaluation Metrics

For any MT evaluation metric to be truly useful, it is necessary to evaluate it as well. The most common way that this has been done in the literature is to determine how well MT evaluation scores of hypothesis sentences correlate with human judgments of those sentences. To determine correlation, the two most commonly used approaches are Pearson correlation and Spearman's rank correlation. These two formulations will be discussed in further detail in § 5.3. Human judgments are known to correlate poorly with each other, since there are a number of subjective factors that go into human decisions about a sentences adequacy and fluency. To mitigate the effect that the differences in scoring between different judges has, human scores are percentile normalized (Blatz et al., 2003).

1.4 Goals and Outline

The goals of this project are to create an MT evaluation metric that will produce system-dependent scores for the purposes of tuning an MT system and examine how various features contribute to adequacy in MT system output. The framework for the system should allow for different types of machine learning approaches to be plugged in without changing the underlying feature generation mechanism. In addition, the features themselves should be customizable to allow the specific tuning of certain features in the MT system. In this way, we will be able to explore the feature space to determine the contribution that different features have in distinguishing between high and low quality in MT system output.

In § 2, we will discuss the various features that were explored in this project. Then in § 3, we will examine the role and nature of the classifier. We describe our implementation in § 4 along with descriptions of issues that came up during development and how they were addressed. Our experimental procedure will be discussed in § 5 followed by a presentation of the results and error analysis in § 6. We conclude in § 7 with a discussion of the overall results of this research as well as possible future work in this area.

2 Features

There are many features that can be extracted from the comparison of a hypothesis and reference translation at the sentence level. In the case of a single reference translation, comparisons are made directly between the two sentences. Having multiple reference translations can cause complications, since there are a number of different decisions that must be made. It is necessary to decide, for example, whether to find the maximum of a measure among all the references, or to average the values of the measure. If the average is used, another decision must be made as to whether to use the geometric mean, the arithmetic mean or the harmonic mean. This complication is part of the reason why BLEU relies solely on precision and avoids recall. This project takes a different approach and relies on a single reference translation for producing scores. The remaining references are used as training examples in the machine learning phase. This has the benefit of making the computations easier to perform, but the disadvantage of losing some information that could contribute to better assessing MT output quality.

2.1 Token-based Features

Token-based features are those features that are determined by comparing tokens in the reference and hypothesis translations. These features examine only the ways in which tokens do or do not appear in the hypothesis. These features include precision, recall, word error rate, length ratio, and minimum edit distance, amongst others. Each of these features attempt to get at how closely the hypothesis tokens match the reference tokens. Two common problems in MT system output are lexical choice and word ordering. Intuitively, n-gram-based features should be able to capture adequacy (and, to a certain extent, fluency) since they look at matching words and sequences of words. Precision and recall measures of n-grams provide helpful tools for determining whether the correct lexical choices and orderings were made.

2.1.1 Precision and Recall

Precision is a measure that captures the ratio of the hypothesis tokens that were valid. Recall measures the ratio of the reference tokens that were covered in the hypothesis. When there is only a single reference translation, these two measures are simple to calculate. Things become more complicated when there are multiple reference translations. BLEU avoids the issue of multiple-reference recall altogether and relies *modified n-gram precision* (Papineni et al., 2002). Modified n-gram precision counts the maximum number of occurrences of any given any word in any of the reference translations. That is the cap for the number of times a hypothesis token of the same word type may be counted. METEOR, recognizing that recall correlates better with human judgments than precision, incorporates recall into its metric by comparing the number of words in the MT output that are in the reference translation to the number of words in the reference (Banerjee and Lavie, 2005).

Since our system relies on a single reference translation for computing measures, the method for calculating these measures on n-grams is straightforward. We chose modified n-gram precision as separate features for $n = \{1, 2, \ldots, 9\}$ (eq. 1). Similarly, we calculated recall for the same values of n (eq. 2). Another common practice is to use the harmonic mean of precision and recall as the F1 score (eq. 3. We also included F1 scores on n-grams of the same lengths, for a total of 27 features.

$$P_n = \frac{\text{count(n-grams in hypothesis also in reference)}}{\text{count(n-grams in hypothesis)}} \tag{1}$$

$$R_n = \frac{\text{count(n-grams in hypothesis also in reference)}}{\text{count(n-grams in reference)}} \tag{2}$$

$$F1_n = \frac{2P_nR_n}{P_n + R_n} \tag{3}$$

```
function minEditDist(target, source):
     n = len(target)
     m = len(source)
     DM = createDistanceMatrix(n+1, m+1)
     DM[0,0] = 0
     for i = 0 to n:
          for j = 0 to m:
               DM[i,j] = min(DM[i-1,j] + 1,
                       DM[i-1,j-1] + 2 * (source_j !=
                       target_i), DM[i,j-1] + 1)
     return DM[i,j]
```

Figure 1: Minimum edit distance algorithm (Jurafsky and Martin, 2000).

2.1.2 Word Error Rate

In this project, we looked at two measures of word error rate, both described by (Kuleska and Shieber, 2004). The first form looks at the minimum edit distance between the hypothesis and reference in terms of words. The minimum edit distance seeks to capture the number of insertions (words that were missing), deletions (words that should not have been included), and substitutions (delete and insert) that must occur in order to make the hypothesis and reference sentences identical. The algorithm for calculating minimum edit distance is given in figure 1. The second form of word error rate is *position independent word error rate*. This measure is calculated by removing the words in the shorter sentence from the longer sentence and the taking the ratio of the remaining set to the length of the longer sentence.

2.1.3 Length Ratio

The final non-syntactic feature used in this project was ratio of hypothesis length to reference length. This feature was described in (Kuleska and Shieber, 2004) and so was included for completeness. Similar to BLEU's brevity penalty, this feature attempts to penalize MT output that is not of sufficient length.

2.1.4 Other Metrics

One of the advantages of this approach is that anything that can be calculated can be used as a feature. Therefore it is possible to add the scores of other metrics as additional feature types. However, due to the fact that this project frames the question of MT evaluation as determining whether sentences are produced by humans or by machines, the effectiveness of other metrics scores are degraded, since the classifier will see examples with high metric scores that are labeled machine and examples with low metric scores that are labeled human.

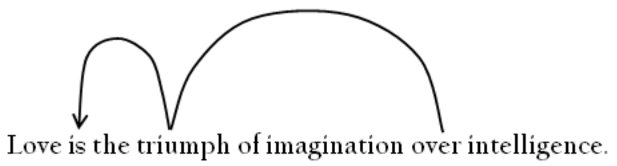

Figure 2: An example of a headword tri-chain.

2.2 Syntax-based Features

The main MT evaluation metrics in use to date do not use syntactic features as part of the evaluation criteria. There are several problems that must be confronted before syntactic features can play a significant role in MT evaluation. The first problem is that automatic parsers must be trained on data. If that training data is from a different domain than the sentences being parsed, quality will be degraded. Also, MT output is often filled with noise that requires the parser to be robust to ungrammatical or unusual constructions. While the parser's failure to parse a sentence may be a strong indicator that the hypothesis is the output of an MT system, it does not offer much help in providing fine-grained quality estimates of MT output. Given a robust parser capable of producing results for the majority of MT output, the next difficulty is determining which particular facets of the parses should be compared to best distinguish between human and MT output. Finally, it remains an open question whether parser output is a sufficiently discriminating factor to improve the quality of MT evaluation metrics.

One approach to using syntactic features in MT evaluation has been the work done by (Liu and Gildea, 2005). They used dependency parses to find matching structures in the reference and hypothesis translations and computed a single score for each sentence based on the number of matching *headword chains*. In a dependency parse, each node of the tree is a word in the sentence. From each node there is a link to the node that it depends on. Dependence is derived from head-modifier relations in the sentence. For example, determiners modify their head nouns, object nouns modify their head verbs, etc. By creating arcs from nodes to their immediate head nodes, a simple tree is produced that avoids finer distinctions such as phrasal nodes (NP, VP, etc.). Typically these trees are not very deep, although that depends on the length and complexity of the sentence. A headword chain is a recursive series of arcs leading from a child to its parent up the tree (see fig. 2). In this example, *over* depends on *triumph* since it is part of the constituent headed by *triumph*. Likewise, *triumph* depends on *is* since it is a part of the constituent headed by *is*. These three words form a headword *trichain*.

The headword chain metric (HWCM) used by (Liu and Gildea, 2005) calculated the value based on multiple reference translations. The reformulation for a single reference translation is simply the precision of headword chain matches

Kernel	Equation
linear	$\vec{x} \cdot \vec{x}'$
polynomial	$(\gamma(\vec{x} \cdot \vec{x}') + c)^d$
radial basis function (rbf)	$e^{-\gamma\|\vec{x}-\vec{x}'\|^2}$
Gaussian	$e^{-\frac{\|\vec{x}-\vec{x}'\|^2}{2\sigma^2}}$
sigmoid	$tanh(\gamma(\vec{x} \cdot \vec{x}') + c)$

Table 3: Common kernel functions.

in the hypothesis compared to the reference. Similar measures for recall and F1 score can also be calculated from this. In our system, all three measures were implemented for headword chains of length $n = \{1, 2, \ldots, 9\}$.

3 Classifier

At the heart of this project is a binary classification task: whether the sentence was produced by a human (positive) or a machine (negative). The choice of a classifier is an important one and depends on many factors. In the case of evaluating MT output, it is helpful to have a continuous score that represents the confidence the classifier has in the output. Also, the classifier should be adaptable to different feature spaces so that the ability to customize the system is not compromised. Support Vector Machines (SVMs) have been used with good results in text classification tasks (Joachims, 2003). Previous work by (Kuleska and Shieber, 2004) also used SVMs as the classifier. Due to these factors, SVMs were chosen as the classifier for this project.

3.1 Properties of SVMs

Support Vector Machines classify items by finding the maximum margin hyperplane that best separates the feature space. Each feature represents a dimension, with each example being a vector in that n-dimensional space of features. Each example is assigned a label (positive or negative, true or false, etc.). In this n-dimensional space of features, if an n-1 dimensional hyperplane can be drawn to separate the data according to their labels, we have a linear classifier. SVMs take this concept further, by adjusting this hyperplane so that the margin between examples and the hyperplane is maximized. The examples that reside on the edge of this margin are called support vectors. If the space is not perfectly separable, SVMs use slack variables so that counter-examples do not hinder the classification task too much.

One of the most important properties of SVMs is their ability to use a kernel function. In the case where the examples in the feature space cannot be separated by a hyperplane, the kernel function can transform this non-linearly separable space into one that is. The kernel function may be applied at any point in the classifier calculation where a dot product is taken. Common kernel functions are given in table 3.

Aside from the kernel parameters, there are several other parameters for an SVM that can be tuned. While it is possible to overfit the data by tuning too much, sometimes it makes sense from a theoretical perspective. The most important non-kernel parameter is *cost*. This is the trade-off between errors on positive and negative examples. The higher the cost, the more costly it is to have an error on a positive example, the lower the cost, the more important it is to classify negative examples correctly. In the case of classifying sentences as human or machine, it may be more helpful to consider accuracy on human examples as more important than accuracy on machine examples, since some machine examples will have human-level quality.

One final note that is important to make: SVMs can be used for more than just classification. They are also frequently used for regression with great success. In the case of multi-class classification, the binary classifier can be used by splitting the problem into n binary classifications. So in the case of movie recommendations, the overall classification problem is to predict what rating a user would assign a movie in the given range $\{1, 2, \ldots, 5\}$. Split into binary tasks, the classifications would be *Is it a 1 or something else?*, *Is it a 2 or something else?*, etc. While this project makes no use of regression or multiclass classification, there are possibilities for future work that will be discussed in § 7.

3.2 Handling SVM Output

In classification mode, an SVM outputs the distance from the test example to the hyperplane. This distance is a loose measure of confidence. The further the distance (and thus the larger the absolute value of the output), the more confident the SVM is of the classification of this example. However, while these distances are rough values of confidence, they do not translate well to probability estimates, since these values are not exactly proportional to membership in the class (Zadrozny and Elkan, 2002). Several approaches have been taken to transform SVM output into probabilities. Two approaches attempted in this project are given below.

3.2.1 Fitting the Sigmoid

The simplest approach to obtaining probability estimates from SVM output is to apply the logistic equation[2] (Platt, 1999). This approach requires that the parameterized form of the logistic equation be found so as to minimize the negative log likelihood of the data. Probability estimates for the data are then calculated as follows:

$$P(y = 1|f) = \frac{1}{1 + e^{Af+B}}. \tag{4}$$

[2]The logistic equation is also known as the sigmoid function. This term is more ambiguous, because it is often used by researchers in neural networks and other fields to refer to the function given in table 3.

```
Given u = svm output, v = labels
function pav(u, v):
     n = length(u)        \\ == length(v)
     for i = 1 to $n-1$:
          if $v_i > v_{i+1}$:
               replace $v_i$, $v_{i+1}$ with their average
               for j = $i-1$ to $0$:
                    if $v_j > v_{j+1}$:
                         replace $v_j$, $v_{j+1}$, ..., $v_{i+1}$ with average
```

Figure 3: The Pool Adjacent Violators Algorithm.

While minimizing the negative log likelihood gives better estimates for probabilities, it is possible to optimize the values to different functions. For this project, the parametric logistic equation was found that maximized the Pearson correlation of human judgments. This will be discussed in more detail in § 5.

3.2.2 Isotonic Regression

While fitting the sigmoid is straightforward and has been shown to give good results, the *pool adjacent violators* (PAV) algorithm has been shown to give more accurate results on certain data sets (Zadrozny and Elkan, 2002). PAV works by performing isotonic regression on the correct labels of the data ordered by their corresponding SVM output scores. SVM output is a numerical value in the interval $[-a, +b]$. These scores are sorted in ascending order and then given a label of 1 for positive examples and 0 for negative examples. If these labels are already monotonically increasing, then nothing need be done and we have correct probability estimates corresponding to the values of the SVM output (0 and 1). If the sequence of labels is not monotonically increasing, then we proceed from left to right and stop at the first pair of violators. These adjacent violators are pooled by replacing both with their average. It is possible for this to cause another adjacency violation to the left in the sequence, so now computation goes from right to left until there are no more violations. All violators that were found are pooled and replaced by their average. Computation resumes from left to right until the end of the sequence has been reached and there are no more violators. The resulting sequence consists of values in the interval $[0, 1]$ and represent accurate probability estimates for their corresponding SVM output scores. The SVM output scores can be used to create bins by which new examples are assigned probabilities. The PAV algorithm is given in figure 3.

The output of the PAV algorithm is a list of SVM output scores mapped to a list of probability estimates, both monotonically increasing. Often there will be sequences of probability estimates that are equal. These may be combined to form bins of SVM output scores. Estimating the probability of unlabeled data is done by finding the bin of SVM output scores that it was classified as and returning the corresponding probability estimate for that bin. Other techniques can be applied to smooth the probability estimate according to its location in that bin such as fitting it to the line formed along with the previous bin's probability estimate.

4 Implementation

In keeping with the goal of customizability, the proposed framework for this project does not rely on one specific classifier, parser or set of features. This framework allows for greater flexibility and choice by researchers seeking to tune their system for specific tasks. Also, it allows the feature space to be explored more systematically by allowing the user to be able to add and drop features and classifiers as needed. The general framework is given in figure 4. The code for this project was written in the python programming language. Parsers and classifiers are called from within the framework. The project version of the code is not yet ready for distribution, but all of the features described in this paper have been implemented.

In the text pre-processing phase, sentences are converted to lower case and punctuation is removed. The resulting tokenized sentences are given beginning and end sentence tags. After the pre-processing phase, sentence pairs consisting of hypothesis and reference translations are fed into the feature generation modules. These modules produce output that maps a feature type to numerical weight for that feature. In the case of the syntactic feature module, the hypothesis and reference translations are fed into the parser prior to calculating feature weights. All of the feature maps are passed to the feature space constructor, where the features are combined and examples are labeled.

Once the feature space has been created, training and testing files are created and the classifier binary is called. We used two implementations for support vector machines in this project. SVMLight is a light-weight SVM implementation that returns distance scores from the example to the hyperplane (Joachims, 1999). The other implementation we used was LIBSVM, which returns probability estimates for the classification decision (Chang and Lin, 2001). Our primary focus was on SVMLight since that offered more control over the SVM output as it gives direct access to the distance scores. The external classifier returns its classification scores for each example and the scoring module post-processes these results to produce a final score. Scores may be calculated using the sigmoid fitting (§ 3.2.1) or pool adjacent violators (§ 3.2.2) methods.

There were two main obstacles in the coding phase of this project. The first is the use of a parser and the requirements specific to each parser. The solution we decided upon was creating a single module that handled all of the syntactic parsing and feature creation that is specific to the Charniak parser and dependency extractor (Hwa and Lopez, 2004). This sacrifices modularity in terms of parser choice, but given more time, the module can be broken down into parts and the framework shown in fig. 4 can be restored. The second obstacle we faced was transforming classifier output into a standard format. For example, SVMLight returns the distance to the hyperplane while LIBSVM returns a probability estimate. Due to time constraints, we have not solved

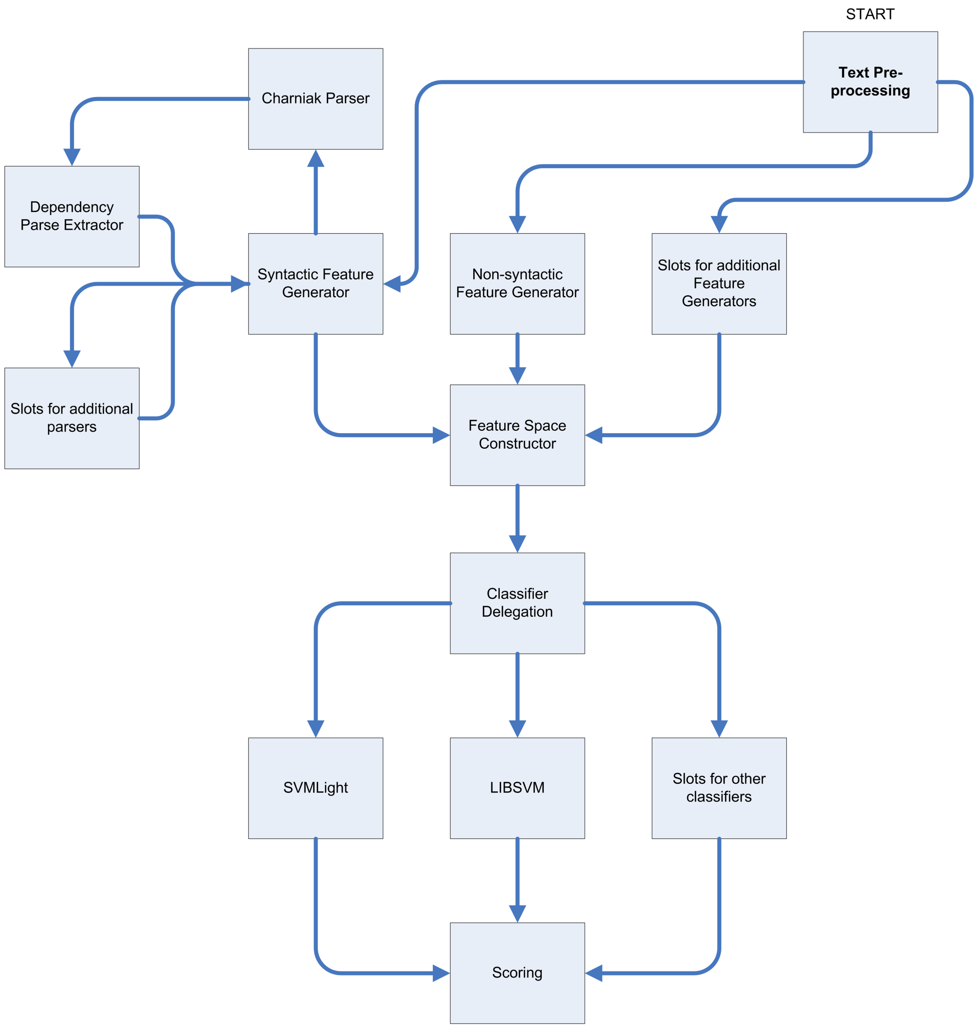
START
Text Pre-processing
Charniak Parser
Dependency Parse Extractor
Syntactic Feature Generator
Non-syntactic Feature Generator
Slots for additional Feature Generators
Slots for additional parsers
Feature Space Constructor
Classifier Delegation
SVMLight
LIBSVM
Slots for other classifiers
Scoring

Figure 4: System components.

this problem in the code, however the solution we envision will be to create a sub-module for each implementation of a classifier that handles the output of that classifier and puts it in a standard form.

5 Experiments

5.1 Data Sets

The data set used in this project is a subset of the 2003 TIDES/MT Evaluation Data that was used in the 2005 ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization containing translations into English from Arabic and Chinese. The data set consists of four human reference translations for both languages and the output of seven MT systems for Arabic and eleven for Chinese. Two of the Arabic MT systems are commercial off the shelf systems (COTS), while four of the chinese systems are COTS. Human judgments have been provided all of the research MT systems. The judges scores were normalized in the fashion described by (Blatz et al., 2003). Since the COTS systems are lacking human judgments, that data was not used in this project. As a result, each source language sentence in Arabic has four human and five MT system translations. Each source language sentence in Chinese has four human and seven MT system translations.

5.2 Experimental Setup

One human reference translation from each language will be held out for calculating feature values. For each set of translations, the data is split at random. Half of the data is broken off to form the training set, while a quarter each go to the development and testing set. The development set is used to tune parameters and the test set is used for final evaluation. Human examples are labeled as $+1$ while machine examples are labeled -1. The number of human examples and machine examples are balanced so that an equal number appear in each set. The final metric scores for the system are given after applying sigmoid fitting (§ 3.2.1) or PAV (§ 3.2.2). Experiments are repeated ten times using different random seeds and are all run on a single 3.06 GHz processor with 2 GB RAM.

5.3 Evaluation

Evaluation of the metric's ability to evaluate MT system output is done by finding the correlation between the metric scores and the percentile-normalized human judgments. There are two primary ways of calculating correlation: Pearson correlation and Spearman's rank correlation. Pearson correlation is calculated

Metric	Pearson	Error	Spearman	Error
Metric$_{\text{syn}}$	0.2001	±0.0083	0.1084	±0.0094
BLEU	0.3315	±0.0073	0.2979	±0.0066
Metric$_{\text{non}}$	0.3550	±0.0064	0.3337	±0.0066
Metric$_{\text{all}}$	0.3555	±0.0057	0.3316	±0.0064
METEOR	0.4076	±0.0055	0.3893	±0.0052
NIST	0.4083	±0.0056	0.3997	±0.0049

Table 4: Pearson and Spearman's rank correlation results.

as follows[3]:

$$r = \frac{\sum XY - \frac{1}{N}\sum X \sum Y}{\sqrt{\left(\sum X^2 - \frac{1}{N}(\sum X)^2\right)\left(\sum Y^2 - \frac{1}{N}(\sum Y)^2\right)}}, \tag{5}$$

where X and Y are distributions corresponding to human judgments and the metric output. Spearman's rank correlation requires that the two distributions be converted to ranks. The difference in ranks from an item in one distribution to the corresponding item in the other distribution forms the distance measure d_i and the rank coefficient is computed as follows:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}, \tag{6}$$

where n is the number of values (Wikipedia, 2007). To compare this metric to the predominant metrics in use currently, the correlation for BLEU, METEOR, and NIST are computed and reported as well. BLEU and NIST scores are computed using the `mteval` toolkit version 11b. METEOR scores were computed using METEOR version 0.5.1 with the exact and porter stemming modules. WordNet synonymy modules were not used, which probably led to lower correlations for that metric.

6 Results

The final best results for Pearson and Spearman's rank correlation are presented in table 4. The metric used in this project outperforms BLEU, but consistently does worse than NIST or METEOR. The reported figures for our metric in the row *Metric*$_{\text{all}}$ include all features described in § 2, except for F1 measures, which tended to hurt correlation. The row *Metric*$_{\text{non}}$ shows results without any syntactic features, and the row labeled *Metric*$_{\text{syn}}$ shows the results using only syntactic features. These results were obtained using the linear kernel and SVMLight. The SVM cost parameter and parameterized sigmoid parameters were tuned using the development set by maximizing Pearson correlation with human judgments. The reported results were evaluated on the held out test set.

[3]Credit: Hyperstat Online, available http://wwwdavidmlane.com/hyperstat/A51911.html.

1.	Bigram Recall
2.	Bigram Precision
3.	Length Ratio
4.	Headword Bichain Recall
5.	Headword Bichain Precision
6.	Word Error Rate
7.	Unigram Recall
8.	Unigram Precision
9.	Five-gram Recall
10.	Six-gram Recall

Table 5: Ten most important features derived from the SVM hyperplane.

Other kernels were attempted, but the correlation was typically substantially lower.

To determine which features were contributing most to the classification decision, we examined the weights of the hyperplane using a script by Thorsten Joachims[4]. The ten features that were consistently ranked most important are given in table 5. While not the most useful, syntactic features do offer some informativeness for the classifier. Based on the results in table 4, they did not hamper accuracy and slightly decreased variance in correlation. In another experiment, METEOR scores (using the single reference translation) were added to the feature vector. The METEOR score moved to the top of the list as the most important feature. This is not surprising since METEOR scores correlate well with human judgments and so should be helpful in this task. However, it also means that METEOR scores diminish the effect of other features. Also, the improvement in correlation does not translate to higher correlation than METEOR itself.

The classification accuracy of the SVM varied depending on what value for the cost parameter was chosen during tuning. Typically, accuracy averaged 68% overall, with 87% accuracy on human examples and 49% accuracy on MT examples. When examining syntactic features only, it improved correlation when accuracy on MT examples improved, however when all the features were combined, correlation improved when accuracy on human examples improved.

7 Conclusions

The goals of this project were to create an MT evaluation system that could be customized to evaluate specific features in the MT output and to examine which features contribute best to assessing adequacy. To that end, we created a modular MT evaluation system that is highly customizable and has better sentence-level correlation than BLEU.

The limitation of using a single reference translation restricted the prob-

[4]Available http://www.cs.cornell.edu/People/tj/svm_light/svm2weight.pl.txt.

lem to evaluating adequacy in terms of similarity to the reference translation. Within this realm, the features we explored correlated more strongly with adequacy than with fluency. While not reported, resutls for fluency were typically 15% lower. We examined matching headword chains between the reference and hypothesis sentences and found that while there was some correlation with adequacy, there was stronger correlation with non-syntactic features. Because of the weak correlation of the syntactic features, the SVM discounted the features dealing with syntax when combined with non-syntactic features. The effect of this discounting is that syntactic features can contribute little benefit even in the cases where they provide stronger informativeness in predicting the correct classification.

A major theoretical shortcoming of this approach to MT evaluation is that the classification question is fundamentally flawed. Some MT output will be human quality and so when the classifier has to learn this example, it observes features corresponding to human output with a contradictory label. It is our belief that this contributes to error. Possible future work in this area could include removing machine examples that have been ranked highly by human judges to see what effect that would have on correlation performance. Another possible extension to this work is to formulate the question as a multi-class classification question. The classification task could then be to distinguish between human and machine, and between good machine output and bad machine output. In this case it may be beneficial to use regression over labels of $\{0, 1, 2\}$, with 0 corresponding to bad machine output and 2 corresponding to human output.

It also would be interesting to see what effect the underlying parser has (Lavie, 2007). Different parsers with different levels of robustness may produce much different results. Likewise, it would also be interesting to see how different classifiers would perform on this data and if another classifier is better suited to this task. However, without addressing the underlying theoretical concern of the classification question, we do not believe that improvement would be very substantial.

8 Acknowledgements

I would like to extend my gratitude to Alon Lavie for his advice and direction on this project, including suggesting focusing on adequacy and seeing what effect headword chain precision and recall have on the classification task. I would also like thank Abhaya Agarwal for his assistance in providing scripts for the parser as well as the data used in this project.

References

Banerjee, S. and Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceed-*

ings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation measures for MT and/or Summarization.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., and Ueffing, N. (2003). Confidence Estimation for Machine Translation. Technical report, JHU / CLSP Summer Workshop.

Callison-Burch, C., Osborne, M., and Koehn, P. (2006). Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of EACL-2006.*

Chang, C.-C. and Lin, C.-J. (2001). *LIBSVM: A Library for Support Vector Machines.* Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Charniak, E., Knight, K., and Yamada, K. (2003). Syntax-based Language Models for Machine Translation. In *Proceedings MT Summit IX.*

Doddington, G. (2002). Automatic Evaluation of MT Quality using N-gram Co-occurence Statistics. In *Proceedings of Human Language Technology Conference 2002*, pages 138–145.

Hwa, R. and Lopez, A. (2004). On the Conversion of Constituency Parsers to Dependency Parsers. Technical Report TR-04-118, Department of Computer Science, University of Pittsburgh.

Joachims, T. (1999). Making Large-scale SVM Learning Practical. In Schoelkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods.* MIT Press.

Joachims, T. (2003). *Learning to Classify Text Using Support Vector Machines.* Kluwer Academic Publishers.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall.

Kuleska, A. and Shieber, S. M. (2004). A Learning Approach to Improving Sentence-level MT Evaluation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation.*

Lavie, A. (2007). Private conversation.

Lavie, A., Sagae, K., and Jayaraman, S. (2004). The Significance of Recall in Automatic Metrics for MT Evaluation. In *Proceedings of AMTA-2004.*

Lita, L. V., Rogati, M., and Lavie, A. (2005). BLANC: Learning Evaluation Metrics for MT. In *HLT '05: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 740–747, Morristown, NJ, USA. Association for Computational Linguistics.

Liu, D. and Gildea, D. (2005). Syntactic Features for Evaluation of Machine Translation. In *Proceedings of the ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization.*

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., and Radev, D. (2003). Syntax for Statistical Machine Translation. Technical report, Johns Hopkins Workshop on Speech and Language Engineering.

Pang, B., Knight, K., and Marcu, D. (2003). Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *NAACL: '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 102–109, Morristown, NJ, USA. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *ACL '02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Platt, J. C. (1999). Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*. MIT Press.

Russo-Lassner, G., Lin, J., and Resnik, P. (2005). A Paraphrase-based Approach to Machine Translation Evaluation. Technical Report Technical Report LAMP-TR-125/CS-TR-4754/UMIACS-TR-2005-57, University of Maryland, College Park, MD.

Turian, J. P., Shen, L., and Melamed, I. D. (2003). Evaluation of Machine Translation and its Evaluation. In *Machine Translation Summit IX*. International Association for Machine Translation.

Wikipedia (2007). Spearman's rank correlation coefficient — Wikipedia, the Free Encyclopedia. Online; accessed 7-May-2007.

Zadrozny, B. and Elkan, C. (2002). Transforming classifier scores into accurate multiclass probability estimates. In *KDD '02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699, New York, NY, USA. ACM Press.