Description of Algorithm and motivation:

Implemented machine translation evaluation classifier, which was trained on 35k training data set and was tested on 15k test data set separated from hyp1-hyp2-ref file. dev.answers file was spitted accordingly as well. Compare-human script was changed to test the accuracy of 15k test data set. Once it gave sufficient accuracy (On test data) after experimentation, it was used to generate eval.out file as stated in readme file.

I implemented variation of chunking penalty for METEOR classifier. A hypothesis specific score was calculated using different features.

Training was done using following features:

1) **Stemmed N-gram precision** (N = 1,2,3,4): ngram precision score for (hyp, ref) pair.
2) **Stemmed N-gram recall** (N = 1,2,3,4): ngram recall score for (hyp, ref) pair.
3) **Meteor score**: meteor score between hypothesis and reference was used as one of the feature. Score was calculated using chunk penalty.
4) **Readability score**: Text readability score was used to one of the feature where we get SMOG index of the text. Python's textstat package was used.
5) **Word ratio** (hyp len / ref len): This feature gave increase of accuracy by 0.05 total.
6) **Rogue score**: Longest Common Subsequence (LCS) based statistics. Longest common subsequence problem takes into account sentence level structure similarity naturally and identifies longest co-occurring in sequence n-grams automatically.
7) **WordNet as matcher**: I used WordNet in the matching process where it was used in the process of calculating intersection of hypothesis and reference. WordNet path distance for similarity used and accuracy was increased by cutoff of 0.t for similarity measure.
8) **Normalization**: data was normalized to improve matches. Unigrams used stemming and WordNet similarity distance whereas bigrams used only stemming matches. Lower cased text was used on sentence and punctuations were removed using regex matcher.
9) **Cosine Similarity**: Cosine similarity between hypothesis and reference sentence was calculated using sklearn TFIDF Vectorizer and it was used as one of the prominent feature in classifier.

Support vector machine (SVM) classifier is used in training and nltk's porter stemmer is also used for creating stem version of words. We divide sentence into three classes, **positive (+1), negative (-1) and neutral (0).** Score is calculated by using sklearnClassifier using SVC classifier and polynomial kernel function. The scores for each classified sentence is the probability of being positive and neutral minus the probability of being negative. We compare the scores of the two sentences and assign the one with higher score as better.
I also experimented with logic to determine which sentence is better by adjusting weights to the probabilities or proposing new score functions.

Results:

Data set	Accuracy
Training (35k)	0.62
Testing (15k)	0.54

References:

[1] Banerjee, S. and Lavie, A. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. in Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association of Computational Linguistics (ACL-2005), Ann Arbor, Michigan, June 2005

[2] Regression and Ranking based Optimisation for Sentence Level Machine Translation Evaluation - Xingyi Song and Trevor Cohn.