Homework #3
Machine Translation
600.468
Spring 2015

Sumit Pawar - (spawar3@jhu.edu)

April 2, 2015

1 Evaluation Challenge Problem 3

For this assignment I implement various variants to the *BLEU* and *METEOR* automatic evaluation metrics.The two criteria for evauation are - fluency and adequacy, that define correctness of measure.

Fluency: Involves both grammatical correctness and idiomatic word choices.

Adequacy: Involves the coherence and conveyance of meaning of the message.

Due to the vagueness of these definitions we want to normalize judgments, for example by averaging the outputs.

We access the quality of a given translation against a human reference translation using precision and recall measures (and in some cases F-measure). Our challenge is to come up with a good similarity measure.

Precision is the ratio of the fraction of correct words to the length of the output words.

$$precision = \frac{correct}{output - length}$$

Or,

$$precision = \frac{|h \cap e|}{|e|}$$

Recall is the ratio of the fraction of correct words to the length of the reference words.

$$recall = \frac{correct}{reference - length}$$

Or,

$$recall = \frac{|h \cap e|}{|h|}$$

2 Metrics Implemented:

1. Baseline BLEU metric:
   The baseline BLEU metric is simply the basic uni-gram or n-gram counts that occur in both the candidate translation and the reference translation. This gives a very weak score of around 0.38 - 0.42 at best.
   The n-gram precision is the familiarity precision measure. We simply count up the the number of candidate translation words (uni-grams) which occur in an reference translation, divided by the total number of words in the candidate translation. The BLEU metric is defined as follows:

$$BLEU - n \quad = \quad brevity - penalty \quad \cdot \exp(\sum_{i=1}^{n} \lambda_i \log precision_i)$$

$$brevity - penalty \quad = \quad \min(1, \frac{output - length}{reference - length})$$

   Best score achieved: 0.42.

   *Reference: Statistical Machine Translation, Koehn $(8.2.3)$.

2. METOER (Metric for Evaluation of Translation with Explicit ORdering):
   The metric is based on the harmonic mean of uni-gram precision and recall, with recall weighted higher than precision. We further employ stemming and synonym matching in a further modification. First let's start by presenting the basic equations for this computation. We have already defined Precision and Recall with equations earlier. The metric was designed to fix some of the problems found in the BLEU metric. The basic formula is given as follows:

$$l(h,e) \quad = \quad \frac{P(h,e) \cdot R(h,e)}{(1-\alpha)R(h,e) + \alpha\, P(h,e)}$$

$$R(h,e) \quad = \quad \frac{|h \cap e|}{|e|}$$

$$P(h,e) \quad = \quad \frac{|h \cap e|}{|h|}$$

The optimum value that was discovered for α is, $\alpha = 0.65$.
Accuracy Acheived: $0.5005866708385481 = 0.501$.

*Reference: Statistical Machine Translation, Koehn $(8.2.4)$.
*Wikipedia: METEOR

3. METEOR with Penalty :
   Here we make a modification to accommodate a penalty term. We use the same equation as above but simply add a penalty term.

$$Pen \quad = \quad \gamma \cdot (frag)^{\beta}$$

The term $l(h, e)$ from the previous sections is now replaced by the term: F_{mean}.

$$score \quad = \quad (1 - Pen) \cdot F_{mean}$$

More clearly, the above formulation can be stated as follows:

$$l(h,e) \quad = \quad \left(1 - \gamma \left(\frac{c}{m}\right)^{\beta}\right) \cdot \frac{P(h,e) \cdot R(h,e)}{(1-\alpha)\, R(h,e) + \alpha\, P(h,e)}$$

Where, α is determined as above.
β, γ are tunable parameters, c is the number of chunks and m is the number of matched uni-grams. The chunks c are the contiguous streams of matched uni-grams that appear consecutively in both, the candidate and reference sentence. β governs the shape of the penalty as a function of the fragmentation, i.e., a continuous matched set of unigrams will be penalized less as compared to that of distributed matched uni-grams. γ is the relative weight assigned to the fragmentation. We gain optimum values for α, γ and β at which we attain highest accuracy and any values above or below the optimum value causes a degradation in the performance. This is like a hill-climbing problem in that sense.

**We try n-gram matches instead of simple uni-gram matches. Besides, we can also give weights to each n-gram matches instead of simple uni-gram matches. For example, if a stream of trigrams match, then that match is weighed higher than the other uni-gram or bi-gram matches. These are eventually normalized, but this is an effective and has proven to give a slightly

better score.

** Overlapped unigram matching: Here I have tried to have overlapped bigram, trigram, 4-gram, $\cdots$, $(n-1)$-gram matches. Each higher gram matched was boosted by a slight $(+1)$ score. These did give a slight increase in performance. I am not sure if the above two optimizations/modifications have been tried before.

The numbers for accuracy and α, β, γ are as follows:

(a) For the basic version of METEOR with penalty:
$\alpha = 0.65, \beta = 3.0, \gamma = 0.9$
accuracy: 0.51071

(b) **n-gram matching:
$\alpha = 0.78, \beta = 4.0, \gamma = 0.9$
accuracy: 0.51125

(c) **overlapped-ngrams with increasing weights per matched n-gram:
$\alpha = 0.78, \beta = 4.0, \gamma = 0.9$
Here the weights are added as $(50+n)$ for each matched set where n ranges from $[1, 2, \cdots, (len_{ref} - 1)]$. We also add a factor to the length for division.
The cumulative score before calculating $l(h,e)is$:

$$p+ = \frac{(n+50)*m}{(len+j+6)}$$

$$r+ = \frac{(n+50)*m}{(h1l+j+6)}$$

where, p is the precision and r is the recall.
These paramters give the best accuracy of:
accuracy: 0.5185

*Reference: Meteor: An Automatic Metric for MT Evaluation with High - Levels of Correlation with Human Judgments
*Wikipedia: METEOR

4. Single-sentence variant of BLEU:
   This is a modification to the BLEU metric discussed initially. Here we use exponents to narrow the gap in difference and give a more realistic score. This is a single-sentence variant for BLEU.

$$BLEU = BP \cdot \exp(\sum_{i=1}^{n-1} \frac{1}{n-1} \log p_i)$$

The BLEU penalty BP is given as:

$$BP = 1 \qquad iff \;\; c \geq r$$

$$= \exp(1 - \frac{r}{c}) \quad otherwise$$

Where, r is the reference translation and c and c is the candidate translation.

We further do a modified n-gram precision on blocks as follows:

$$p_n = \frac{\sum_{C \in Candidates} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in Candidates} \sum_{n-gram' \in C'} Count(n-gram')}$$

** Here we implement similar modifications as in the above case, i.e., we try and optimize the parameters as much as possible within the limited settings. We are already implementing multiple n-grams and averaging, we can also add weights to each level of n-gram matches. We multiply by a factor of $(50 + n)$. This gives a slight improvement. Any other values, lower or higher will cause a degradation ot no change. Here are the scores:

(a) Single-sentence BLEU - accuracy: 0.5105

(b) Weighted n-gram single-sentence BLEU - accuracy: 0.51465

These scores do not beat the METEOR scores, probably because they do not consider chunks matched.

*Reference: BLEU: a Method for Automatic Evaluation of Machine Translation
*BLEU assignment at Edinburgh

5. BLEU with synonyms:
   This is a modification to BLEU where I am comparing each candidate with the candidates synonyms as well as the reference synonyms for each word. This obviously takes a long time, not only to fetch synonyms and compare them but also to construct this setup for each candidate-reference pair. I tried using map-reduce for this, but as can be imagined the main points of pain were involved with seting up Hadoop, and transforming the BLEU algorithm to satisfy the Map-Reduce paradigm. In addition to the overhead of debugging. It runs quicker thought, but the output needs to be finally sorted in the simplest Map reduce setting because the keys of the output and that of "dev_ans" may not match, i.e., the ordering is lost, but can be tracked with keys (line numbers). However, this did lead to a slight increase in the score. I used the wordnet JAWS library available online *jaws-bin.jar*. Of course I did install wordnet on my local Ubuntu VM too, without which it won't work.

   Accuracy: 0.514687

   *Reference: Wordnet
   * wordnet for java - jaws-bin.jar

6. METEOR with synonyms and word stems :
   Here I tried the same synonyms and stemming methods using wordnet as described above. I used the best METEOR version which had weighted n-gram optimizations to get my best score on this assignment. I did observe that synonyms did not work well for anything above unigrams and therefore restricted n-gram and synonyms to unigrams. I was hopping word net might return single words for a bigrams, trigrams, etc, but this does not seem to be the case and hence to avoid the overhead I made the restriction to unigrams. I could not try this on hadoop due to the overheads involved. Here too I used the java JAWS wordnet library.

   Accuracy: 0.5301687

*Reference: Wordnet
* wordnet for java - jaws-bin.jar

7. Meteor with SVM classification:
   For this I used SVM lib where each candidate translation's unigrams were classified against the references. This was very time consuming and the initial setup took a long time to run with average results. Further efforts were abandoned due to this. The best result obtained were as good as the initial METEOR and I did not find it much useful to pursue with this. The library used was SVM. The algorithm used was simple vectorization. I had also added synonyms and stems and therefore this took atleast a couple of hours on the first run with just unigrams.

   Best Accuracy: 0.50058

   *References: SVMlib for java
   Learn a classifier from the training data

3 Conclusion:

Even after trying multiple methods, the improvement in the automated scores was not much and fairly around the 0.53 mark at best. Which we can say is slightly better than random guessing, but there can be more that can be done. A lot of the papers boast of great scores but even after implementing the papers I could not replicate the scores. So these may be data specific scores or our data may be too limited.

Another problem would be that as the methods applied to get a better score are added as layers over the basic BLEU and METEOR metrics, the complexity of the programs increase and evaluation takes a fairly long time. One modification that I tried at the system level was using Hadoop Map Reduce (parallel programming) for a version of BLEU. This did give a fairly improved performance but the overhead is in re-designing the problem to satisfy the Map-reduce paradigm. And, further the overhead of debugging was beyond the scope of this assignment, and ofcourse very time consuming.

Other interesting techniques that I would have loved to try would have been using syntax and parse trees, better methods of classification and string subsequence kernels.

-