Statistical Machine Translation using Ordinal Regression and Beam Search Decoder

Ankit Patel ampatel@sfu.ca
Shruthi Narayan shruthin@sfu.ca
Manekta Bedi mbedi@sfu.ca
Srikanth Muralidharan smuralid@sfu.ca

Abstract

In this work, we explore several approaches to improve machine translation, especially focusing on iteratively improving the performance of decoder-reranker system. We train our system using the dev set, and provide experimental results on the test set. We use the dev-test filtered phrase table, and n-gram language model for the extraction of translation model based features, and language model based features respectively. We use the beam search decoder, with eight features, and ordinal regression algorithm for learning the feature weights, and use the BLEU metric for evaluation of translations. We find that this process improves the performance of the system, compared the baseline system which uses PRO algorithm[2] for reranking.

1 Motivation

Phrase based decoders are popular in the machine translation systems. They are considered more appropriate for finding a good translation for a given input, compared to word-by-word translations like IBM models, which does not consider the context (e.g. the phrase to which it belongs) of a particular word of concern. Therefore, in this work, we are mainly interested in building phrase based translation models.

The noisy channel model is a popular generative model representation of a machine translation system for long time. It states that for a given set of source sentences *f*, we can derive the set of target sentences *e* as:

$$\begin{aligned} e &= \arg\max_e P(e|f) \\ &= \arg\max_e P(f|e)P(e) \end{aligned}$$

Here, $P(f|e)$ corresponds to the translation model based score, and $p(e)$ corresponds to the n-gram language model score. The original search space of this problem has infinitely many candidate translation, and therefore finding the optimal translation is NP-hard problem. Therefore, we employ a dynamic programming based Viterbi approximation to the decoding task. In this work, we used a simple, fast beam search decoder which uses top-k candidates to construct the dynamic programming table. For this work, we use one IBM model 1 alignment feature, language model feature, four translation features, a feature for counting the number of words in the sentence, and another feature that computes the number of untranslated source words. Our approach allows translation of Chinese phrases in any order . We find that this better improves the performance than a baseline algorithm which does not change the order of Chinese phrases.

In order to efficiently consolidate for the imperfect long distance reordering that is performed by the phrase based decoder technique mentioned above, we use a reranking algorithm that partially accounts for these errors. We use n-best translations for a sentence produced by the decoder as input to a discriminative re-ranking algorithm that learns the weights for each of the decoder's features. More specifically, we employ, a re-ranker based on ordinal regression, that is shown to be efficient ranking algorithm in machine learning.

In this work, we try to improve the performance of phrase based decoder after by learning appropriate feature weights through this reranker algorithm.

The **salient contributions** of our work are:

1. Implementation of beam search decoder to come up with N-best translation list using the weights for the features involved.
2. Use of descriminative ordinal regression based reranking algorithm to come up with optimal weights for the features used.
3. Combine both the decoder and reranker using the learned weights as the feedback loop to improve the performace of the translation.

2 Proposed approach

As mentioned in the previous section, we iteratively employ the ranking, and decoding algorithms.

2.1 Features used in our experiment

We experimented with adding a variety of features for scoring candidate translations.[5]

- Target language model: As a language model feature, we use a word-based N-gram language model from the given dataset.
- Direct phrase translation probability $p(e|f)$: Gives the probability of target phrase e given the source phrase f.
- Inverse phrase translation probability $p(f|e)$: Gives the probability of source phrase f given the target phrase e.
- Direct Lexical weighting $lex(e|f)$: The sum of target word probabilities for the given source words and the alignment of the phrase pairs
- Inverse lexical weighting $lex(f|e)$: The sum of source word probabilities for the given target words and alignment.
- Target sentence word count feature: Count of number of words included in the translation.
- Untranslated word count feature: Count of number of untranslated source sentence words in the translation.
- IBM Model-1 score as a feature: IBM Model 1 is a bag-of-word translation model and it gives the sum of all possible alignment probabilities over the source and target sentence pair. Model 1 gives a probability of any given translation pair, which is given by

$$[h]P(e|f;M1) = \frac{\in}{(l+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{l} t(f_j|e_i)$$

For a missing translation word pair or unknown words, where $t(f_j, e_i) = 0$ according to the model, we used a constant $t(f_j, e_i) = 10^{-50}$ as a smoothing value.

2.2 Beam search decoder

Algorithm 1 Stack Decoding Algorithm

Require: Source sentences $x_1, ..., x_n$, translation model $\mathcal{L}$, language model h, distortion penalty η, distortion limit **d**

initialization: set $Q_0 = \{q_0\}, Q_i = \phi$ for $i = 1...n$.
for all stacks $0, ..., n-1$ **do**
    for each state $q \in BEAM(Q_i)$, for each phrase $p \in ph(q)$ **do**
        (1) q' =next(q, p)
        (2) ADD (Q_j, q', q, p) where j =len(q')
    end for
end for
return : Highest scoring state in Q_n. Backpointers can be used to find the underlying sequence of phrases

function ADD(Q, q', q, p)
    if there is some $q'' \in Q$ such that $eq(q'', q') = True$:
        if $\alpha(q') > \alpha(q'')$
            $Q = \{q'\} \bigcup Q \setminus \{q''\}$
            $set\ bp(q') = (q, p)$
        else return
        end if
    else:
        $Q = \{q'\} \bigcup Q$
        $set\ bp(q') = (q, p)$
    end if
end function

function BEAM(Q)
    $\alpha^* = \arg\max_{q \in Q} \alpha(q)$
    i.e., α^* is the highest score for any state in Q.
    Define $\beta \geq 0$ to be the beam width parameter
    Then
        beam(Q) = $\{q \in Q \ : \alpha(q) \geq \alpha^* - \beta\}$
end function

Algorithm 1 is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set [1]. The implementation defines a data structure namely 'state' to capture the path and extent of translation of a particular graph branch. The state also holds the score for that translation. The baseline model uses the weighted sum of scores of first 5 features mentioned in the features section. For every word in the source sentence we define a stack to hold the highest scoring states. We initialize all the stacks to be empty. We then iterate: for each

$i \in \{1, 2, ..n\}$ we consider states within the stack. For each state within the stack we retrieve a set of candidate phrases that can be appended to the state. For each such phrase we compute the new state. We then add this state to the next stack provided a similar state with better score is not already available. Once this process is completed for all the states within the current stack, only the highest scoring states are retained in the next stack. Once this process completed for all the stacks, the top k states from the last stack are considered as n-best.

2.3 Reranking algorithm

Reranking techniques as described in **Algorithm 2** are used to rerank model-generated N-best candidates, using a rich set of local and global features. To apply reranking technique to our list of N-best sentences, we used ordinal regression with uneven margin algorithm proposed by [4]. The reason for using uneven margin along with $\in$-insensitive ordinal regression is to penalize errors that might be more significant than others. For instance, if $\in$ is 10, if ordinal regression cannot recognize that the error on (r_1,r_{11}) is more significant than the error on (r_{21}, r_{31}) , r_j being parse that ranks j for a sentence. So, in order to fix this problem, we use this parse to improve the performance of the ordinal regression algorithm. In our reranker we took the margin as 0.001, $\in$ as 20 and computed g as $g(p, q) = \frac{1}{p} - \frac{1}{q}$, and searched for a hyperplane such that

$$score(r_p) - score(r_q) > g(p, q) * margin$$

where $g(1, 10) > g(1, 2) > g(10, 11)$. The pseudocode of the algorithm we used in reranker is as given below.

Algorithm 2 Ordinal regression with uneven margin

Require: a postive learning margin τ
t $\leftarrow 0$, initialize $\mathbf{w}^0$
repeat
  for (sentence $i = 1, ..., m$) **do**
    compute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ and $u_j \leftarrow 0$ for all *j*
    for $(1 \leq j < l \leq n)$ **do**
      if ($y_{i,j} < y_{i,l}$ and $dis(y_{i,j}, y_{i,l}) > \in$ and $\mathbf{w}^t \cdot$
        $x_{i,j} - \mathbf{w}^t \cdot x_{i,l} < g(y_{i,j}, y_{i,l})\tau$)) **then**
        $u_j \leftarrow u_j + g(y_{i,j}, y_{i,l})\tau$
        $u_l \leftarrow u_l + g(y_{i,j}, y_{i,l})\tau$
      end if
    end for
    $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_j u_j x_{i,j}$; $t \leftarrow t + 1$;
  end for
until no updates made in the outer **for** loop

2.4 IBM Model 1

We used IBM model 1 [3] score as one of the features. As described in **Algorithm 3** we build the model $Pr(f \mid e)$ using a conditional probability $t(f \mid e)$ where f is a French word and e is an English word. Let the French sentence be $\mathbf{f} = (f_1, \cdots, f_I)$ and the English sentence $\mathbf{e} = (e_1, \cdots, e_J)$. There is one a_i for each French word f_i which corresponds to an English

word e_{ai}. Let the alignment vector be $\mathbf{a} = (a_1, \cdots, a_I)$.

$$P(\mathbf{f}|\mathbf{a}; \mathbf{e}) = \prod_{i=1}^{I} t(f_j|e_{ai})$$

In order to estimate the parameters $t(f \mid e)$ we start with an initial estimate t_0 and modify it iteratively to get $t_1, t_2, \cdots$ The parameter updates are derived for each French word f_i and English word e_j as follows:

$$t_k(f_i|e_j) = \sum_{s=1}^{N} \sum_{(f_i|e_j)\in(\mathbf{f}^{(s)},\mathbf{e}^{(s)})} \frac{count(f_i, e_j, \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}{count(e_j, \mathbf{f}^{(s)}, \mathbf{e}^{(s)})}$$

These counts are expected counts over all possible alignments, and each alignment has a probability computed using t_{k-1}.Pseudo-code is as given below. The initial parameter t_0 is such in a way that diagonal entries in $\mathbf{f} \rightarrow \mathbf{e}$ matrix has higher value than the others.The reason for doing this is under assumption that the french word and its english translation will be at same index in the sentence pair.

Algorithm 3 IBM Model 1

```
k ← 0
Initialize t_0
repeat
    for each (f, e) in D do
        for each f_i in f do
            Z ← 0
            for each e_j in e do
                Z ← Z + t_{k-1}(f_i | e_j)
            end for
            for each e_j in e do
                c ← t_{k-1}(f_i | e_j)/Z
                count(f_i, e_j) ← count(f_i, e_j) + c
                count(e_j) ← count(e_j) + c
            end for
        end for
    end for
    for each (f, e) in count do
        Set new parameters: t_k(f | e) = count(f,e) / count(e)
    end for
until convergence
```

3 Code

3.1 Decoder

For the decoder, we used our homework implementation of the beam search decoder mentioned in the previous section. As mentioned in the algorithm, we allowed our decoder to re-order the source phrase translations, but with a distortion limit, and also a penalty for re-ordering during the translations. We also used our version of ordinal regression for reranking, with a step size of 0.01, margin was set to 0.001, and convergence threshold was set to 0.0001. For applying

line search algorithm to learn the feature weights, we considered top 5 and bottom 5 candidate translations, that are at least 20 ranks away from each other, and used them as a set of positive and negative examples.

3.2 Reranker

For the reranker, we used our code for homework assignment where we implemented ordinal regression using uneven margins mentioned in the previous section. As demanded by this algorithm we set the parameters in such a way that it maintains a larger margin in translations of high ranks and a smaller margin in translations of low ranks. We experimented with many informative features that correlate with BLEU, and effective learning algorithms that optimize θ for BLEU, and tried to improve the score to promising scale.

We compared the results of this algorithm with the baseline PRO (pairwise ranking optimization) algorithm[2]. We kept samples generated from n-best list per input sentence(τ) as 5000. Sampler acceptance cutoff(α) and perceptron learning rate(η) was set to 0.1. Also we kept 5 epochs for perceptron training.

3.3 Aligner

We used our IBM model 1 aligner implementation for generating the alignment based features. We first learn the aligner weight table, and use it to construct alignment feature in our decoder by computing alignments each time, when new phrase is considered. We compute the alignment feature using the expression mentioned in the previous section.

4 Experiments

4.1 Datasets

We performed our experiments on the test dataset provided in the project. We used the above mentioned features for our work. For the translation model, we used the dev-test filtered phrase table provided in the project For the language model, we used the language model that was filtered for dev and test files. We produced the alignment weights using the four reference english files given in the dev folder to build the alignment based features described in the previous section. We tested the performance of our translation system on the test dataset given in the test folder.

4.2 Results

We provide experimental results on the given test dataset. We use the complete dev, and test dataset for performing our experiments. In our experiment we use 10 translations per phrase, for producing 100 best translations, in the decoder, and also for training our reranker. We used the same decoder parameter settings that we had for the assignment. For setting the parameters of reranker, we used five features that were given to us (TM phrase table, and LM), and tuned the parameters for reranker. We found that the minimum distance function (between rank of two given translations), when set as 20, gave us the optimal results. The main experiments that we performed are:

Case 1. Baseline Method: We used the perceptron algorithm along with the beam search decoder, and the five given TM plus LM features as our baseline algorithm.

Case 2. Baseline + 2 Features: This test tries to improve the baseline method mentioned above, by adding new features. We additionally added the count of number of words, and the number of common words in the source and the target.

Case 3. Baseline + 2 Features + Ordinal regression: In this step, we compare the performance of perceptron algorithm, and the ordinal regression algorithm. In this experiment, we replace the perceptron algorithm with the ordinal regression algorithm.

Case 4. Baseline + 3 Features + Ordinal regression algorithm: In this step, we add the IBM model 1's alignment features to the set of available features. In this step, we wanted to study the importance of alignment features in the performance of our machine translation system.

The following table shows the BLEU scores obtained for each of these cases:

Table 1: Translation Results

Case	BLEU scores obtained
1	0.043
2	0.056
3	0.067
4	0.082
Case 2 for 2 iterations	0.074
Beam search decoding with uniform weights	0.0314

From the table, we find that beam search decoder without reranking provides a BLEU score of just 0.0314, whereas the baseline algorithm, which uses perceptron based reranking, it learns better feature weights, and provides a BLEU score of 0.043. We also find that using ordinal regression for reranking improves the performance of our translation system. As shown, ordinal regression (0.067) provides better BLEU score than the perceptron algorithm (0.056). We also find that adding the word count, and untranslated word's count increases the BLEU metric score further to 0.056, which is better than the baseline case.

5 Challenges

- Setting of ordinal regression parameters and margin compatible to the project data files.
- Prolonged execution time of the decoder for large dataset.

6 Conclusion

In this paper, we have successfully applied the discriminative reranking to machine translation. Through our experiments, we show that ordinal regression with uneven margins, along with the beam search decoder provide the best set of results. We have also shown that the IBM model features improve the performance of our reranker significantly. We also observed that the performance of our translation system improves with the number of iterations.

7 Discussion

We have shown through our experiments that reranking process helps to improve the quality of machine translation. However, there is a potential scope of improvement in our existing methodology for decoding and reranking. For instance, during each iteration of decoding, we prune the set of candidate translations to the ones that have higher translation score upto that position, where we might end up removing the candidate which is an actual translation. We

believe that using other dynamic programming techniques like Lagrangian relaxation based decoding method could handle this problem more robustly, and could yield better solutions.

We could also have tested other reranking methods, and we could also have combined the results of several reranking methods which, on an average,might perform better than a single reranking algorithm. The current version of reranker performs a linear search for learning feature weights, and scalability of these kinds of methods to large number of features seems to be a bit of concern to us. Therefore, we can also try representations that scale well to large number of features.

References

[1] Michael Collins. Phrase-based translation models. April 2013.

[2] Mark Hopkins and Jonathan May. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics, 2011.

[3] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alexander Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, et al. A smorgasbord of features for statistical machine translation. In *HLT-NAACL*, pages 161–168, 2004.

[4] Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *HLT-NAACL*, pages 177–184, 2004.

[5] Ruiqiang Zhang and Eiichiro Sumita. Chinese unknown word translation by subword resegmentation. In *IJCNLP*, pages 225–232, 2008.