A Continuous Relaxation of Beam Search for End-to-End Training of Neural Sequence Models

Kartik Goyal
Carnegie Mellon University
kartikgo@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu

Chris Dyer
Deepmind
cdyer@google.com

Taylor Berg-Kirkpatrick
Carnegie Mellon University
tberg@cs.cmu.edu

Abstract

Beam search is a desirable choice of test-time decoding algorithm for neural sequence models because it potentially avoids search errors made by simpler greedy methods. However, typical cross entropy training procedures for these models do not directly consider the behaviour of the final decoding method. As a result, for cross-entropy trained models, beam decoding can sometimes yield reduced test performance when compared with greedy decoding. In order to train models that can more effectively make use of beam search, we propose a new training procedure that focuses on the final loss metric (e.g. Hamming loss) evaluated on the output of beam search. While well-defined, this "direct loss" objective is itself discontinuous and thus difficult to optimize. Hence, in our approach, we form a sub-differentiable surrogate objective by introducing a novel continuous approximation of the beam search decoding procedure. In experiments, we show that optimizing this new training objective yields substantially better results on two sequence tasks (Named Entity Recognition and CCG Supertagging) when compared with both cross entropy trained greedy decoding and cross entropy trained beam decoding baselines.

1 Introduction

Sequence-to-sequence (seq2seq) models have been successfully used for many sequential decision tasks such as machine translation (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015), parsing (Dyer et al. 2016; 2015), summarization (Rush, Chopra, and Weston 2015), dialog generation (Serban et al. 2015), and image captioning (Xu et al. 2015). *Beam search* is a desirable choice of test-time decoding algorithm for such models because it potentially avoids search errors made by simpler greedy methods. However, the typical approach to training neural sequence models is to use a locally normalized maximum likelihood objective (cross-entropy training) (Sutskever, Vinyals, and Le 2014). This objective does not directly reason about the behaviour of the final decoding method. As a result, for cross-entropy trained models, beam decoding can sometimes yield reduced test performance when compared with greedy decoding (Koehn and Knowles 2017; Neubig 2017;

Cho et al. 2014). These negative results are not unexpected. The training procedure was not *search-aware*: it was not able to consider the effect that changing the model's scores might have on the ease of search while using a beam decoding, greedy decoding, or otherwise.

We hypothesize that the under-performance of beam search in certain scenarios can be resolved by using a better designed training objective. Because beam search potentially offers more accurate search when compared to greedy decoding, we hope that appropriately trained models should be able to leverage beam search to improve performance. In order to train models that can more effectively make use of beam search, we propose a new training procedure that focuses on the final loss metric (e.g. Hamming loss) evaluated on the output of beam search. While well-defined and a valid training criterion, this "direct loss" objective is discontinuous and thus difficult to optimize. Hence, in our approach, we form a sub-differentiable surrogate objective by introducing a novel continuous approximation of the beam search decoding procedure. In experiments, we show that optimizing this new training objective yields substantially better results on two sequence tasks (Named Entity Recognition and CCG Supertagging) when compared with both cross-entropy trained greedy decoding and cross-entropy trained beam decoding baselines.

Several related methods, including reinforcement learning (Ranzato et al. 2016; Bahdanau et al. 2017), imitation learning (Daumé, Langford, and Marcu 2009; Ross, Gordon, and Bagnell 2011; Bengio et al. 2015), and discrete search based methods (Wiseman and Rush 2016; Andor et al. 2016; Daumé III and Marcu 2005; Gormley, Dredze, and Eisner 2015), have also been proposed to make training search-aware. These methods include approaches that forgo direct optimization of a global training objective, instead incorporating credit assignment for search errors by using methods like *early updates* (Collins and Roark 2004) that explicitly track the reachability of the gold target sequence during the search procedure. While addressing a related problem – credit assignment for search errors during training – in this paper, we propose an approach with a novel property: we directly optimize a continuous and global training objective using backpropagation. As a result, in our approach, credit assign-

Algorithm 1 Standard Beam Search

1: **Initialize:**
$\qquad h_{0,i} \leftarrow \vec{0}, e_{0,i} \leftarrow embedding(\texttt{<s>}), s_{0,i} \leftarrow 0, i = 1, \ldots, k$
2: **for** t = 0 to T **do**
3: $\qquad$ **for** i = 1 to k **do**
4: $\qquad\qquad$ **for all** $v \in V$ **do**
5: $\qquad\qquad\qquad \tilde{s}_t[i,v] \leftarrow s_{t,i} + f(h_{t,i}, v)$ $\hfill \triangleright f$ is the local output scoring function
6: $\qquad s_{t+1} \leftarrow$ *top-k-max*$(\tilde{s}_t)$ $\hfill \triangleright$ Top k values of the input matrix
7: $\qquad b_{t+1,*}, y_{t,*} \leftarrow$ *top-k-argmax*$(\tilde{s}_t)$ $\hfill \triangleright$ Top k argmax index pairs of the input matrix
8: $\qquad$ **for** i = 1 to k **do**
9: $\qquad\qquad e_{t+1,i} \leftarrow embedding(y_{t,i})$
10: $\qquad\qquad h_{t+1,i} \leftarrow r(h_{t,i}, e_{t+1,i})$ $\hfill \triangleright r$ is a nonlinear recurrent function that returns state at next step
11: $\hat{y} \leftarrow$ *follow-backpointer*$((b_{1,*}, y_{1,*}), \ldots, (b_{T,*}, y_{T,*}))$
12: $s(\hat{y}) \leftarrow max(s_T)$

---

ment is handled directly via gradient optimization in an end-to-end computation graph. The most closely related work to our own approach was proposed by Goyal, Dyer, and Berg-Kirkpatrick (2017). They do not consider beam search, but develop a continuous approximation of greedy decoding for scheduled sampling objectives. Other related work involves training a generator with a Gumbel reparamterized sampling module to more reliably find the MAP sequences at decode-time (Gu, Im, and Li 2017), and constructing surrogate loss functions (Bahdanau et al. 2016) that are close to task losses.

2 Model

We denote the seq2seq model parameterized by θ as $\mathcal{M}(\theta)$. We denote the input sequence as x, the gold output sequence as y^* and the result of beam search over $\mathcal{M}(\theta)$ as $\hat{y} = Beam(x, \mathcal{M}(\theta))$. Ideally, we would like to directly minimize a final evaluation loss, $L(\hat{y}, y^*)$, evaluated on the result of running beam search with input x and model $\mathcal{M}(\theta)$. Throughout this paper we assume that the evaluation loss decomposes over time steps t as: $L(\hat{y}, y^*) = \sum_{t=1}^{T} d(\hat{y}_t, y^*)$[1]. We refer to this idealized training objective that directly evaluates prediction loss as the "direct loss" objective and define it as:

$$\min_\theta G_{\text{DL}}(x, \theta, y^*) = \min_\theta L(Beam(x, \mathcal{M}(\theta)), y^*) \quad (1)$$

Unfortunately, optimizing this objective using gradient methods is difficult because the objective is discontinuous. The two sources of discontinuity are:

1. As we describe later in more detail, beam search decoding (referred to as the function *Beam*) involves discrete argmax decisions and thus represents a discontinuous function.

2. The output, $\hat{y}$, of the *Beam* function, which is the input to the loss function, $L(\hat{y}, y^*)$, is discrete and hence the evaluation of the final loss is also discontinuous.

We introduce a surrogate training objective that avoids these problems and as a result is fully continuous. In order to

---

[1] This assumption does not hold for some popular evaluation metrics (e.g. BLEU). In these cases, surrogate evaluation losses such as Hamming distance can be used .

accomplish this, we propose a continuous relaxation to the *composition* of our final loss metric, L, and our decoder function, $Beam$:

$$softLB(x, \mathcal{M}(\theta), y^*) \approx (L \circ Beam)(x, \mathcal{M}(\theta), y^*)$$

Specifically, we form a continuous function *softLB* that seeks to approximate the result of running our decoder on input x and then evaluating the result against y^* using L. By introducing this new module, we are now able to construct our surrogate training objective:

$$\min_\theta \tilde{G}_{\text{DL}}(x, \theta, y^*) = \min_\theta softLB(x, \mathcal{M}(\theta), y^*) \quad (2)$$

Specified in more detail in Section 2.3, our surrogate objective in Equation 2 will additionally take a hyperparameter α that trades approximation quality for smoothness of the objective. Under certain conditions, Equation 2 converges to the objective in Equation 1 as α is increased. We first describe the standard discontinuous beam search procedure and then our training approach (Equation 2) involving a continuous relaxation of beam search.

2.1 Discontinuity in Beam Search

Formally, *beam search* is a procedure with hyperparameter k that maintains a beam of k elements at each time step and expands each of the k elements to find the k-best candidates for the next time step. The procedure finds an approximate argmax of a scoring function defined on output sequences.

We describe beam search in the context of seq2seq models in Algorithm 1 – more specifically, for an encoder-decoder (Sutskever, Vinyals, and Le 2014) model with a nonlinear auto-regressive decoder (e.g. an LSTM (Hochreiter and Schmidhuber 1997)). We define the global model score of a sequence y with length T to be the sum of local output scores at each time step of the seq2seq model: $s(y) = \sum_{t=1}^{T} f(h_t, y_t)$. In neural models, the function f is implemented as a differentiable mapping, $\mathbb{R}^{|h|} \rightarrow \mathbb{R}^{|V|}$, which yields scores for vocabulary elements using the recurrent hidden states at corresponding time steps. In our notation, $h_{t,i}$ is the hidden state of the decoder at time step t for beam element i, $e_{t,i}$ is the embedding of the output symbol at time-step t for beam element i, and $s_{t,i}$ is the cumulative model

Algorithm 2 continuous-top-k-argmax

1: **Inputs:**
   $s \in \mathbb{R}^{k \times |V|}$
2: **Outputs:**
   $p_i \in \mathbb{R}^{k \times |V|}$, s.t. $\sum_j p_{ij} = 1, i = 1, \ldots, k$
3: $m \in \mathbb{R}^k = \textit{top-k-max}(s)$
4: **for** $i = 1$ to k **do**   ▷ *peaked-softmax* will be dominated by scores closer to m_i
5:     $p_i = \textit{peaked-softmax}_\alpha(-(s - m_i \cdot \mathbf{1})^2)$ ▷ The square operation is element-wise

---

score at step t for beam element i. In Algorithm 1, we denote by $\tilde{s}_t \in \mathbb{R}^{k \times |V|}$ the cumulative candidate score matrix which represents the model score of each successor candidate in the vocabulary for each beam element. This score is obtained by adding the local output score (computed as $f(h_{t,i}, w)$) to the running total of the score for the candidate. The function r in Algorithms 1 and 3 yields successive hidden states in recurrent neural models like RNNs, LSTMs etc. The *embedding* operation maps a word in the vocabulary V, to a continuous embedding vector. Finally, backpointers at each time step to the beam elements at the previous time step are also stored for identifying the best sequence $\hat{y}$, at the conclusion of the search procedure. A backpointer at time step t for a beam element i is denoted by $b_{t,i} \in \{1, \ldots, k\}$ which points to one of the k elements at the previous beam. We denote a vector of backpointers for all the beam elements by $b_{t,*}$. The *follow-backpointer* operation takes as input backpointers $(b_{t,*})$ and candidates $(y_{t,*} \in \{1, \ldots, |V|\}^k)$ for all the beam elements at each time step and traverses the sequence in reverse (from time-step T through 1) following backpointers at each time step and identifying candidate words associated with each backpointer that results in a sequence $\hat{y}$, of length T.

The procedure described in Algorithm 1 is discontinuous because of the *top-k-argmax* procedure that returns a pair of vectors corresponding to the k highest-scoring indices for backpointers and vocabulary items from the score matrix $\tilde{s}_t$. This index selection results in hard backpointers at each time step which restrict the gradient flow during backpropagation. In the next section, we describe a continuous relaxation to the *top-k-argmax* procedure which forms the crux of our approach.

2.2 Continuous Approximation to *top-k-argmax*

The key property that we use in our approximation is that for a real valued vector z, the argmax with respect to a vector of scores, s, can be approximated by a temperature controlled softmax operation. The argmax operation can be represented as:

$$\hat{z} = \sum_i z_i \mathbb{1}[\forall i' \neq i, \ s_i > s_{i'}],$$

which can be relaxed by replacing the indicator function with a *peaked-softmax* operation with hyperparameter α:

$$\tilde{z} = \sum_i z_i \frac{\exp(\alpha \, s_i)}{\sum_{i'} \exp(\alpha \, s_{i'})} = z^T \cdot \frac{\text{elem-exp}(\alpha \, s)}{\sum_{i'} \exp(\alpha \, s_{i'})}$$
$$= z^T \cdot \textit{peaked-softmax}_\alpha(\mathbf{s})$$

As $\alpha \to \infty$, $\tilde{z} \to \hat{z}$ so long as there is only one maximum value in the vector z. This *peaked-softmax* operation has been shown to be effective in recent work (Maddison, Mnih, and Teh 2017; Jang, Gu, and Poole 2016; Goyal, Dyer, and Berg-Kirkpatrick 2017) involving continuous relaxation to the argmax operation, although to our knowledge, this is the first work to apply it to approximate the beam search procedure.

Using this *peaked-softmax* operation, we propose an iterative algorithm for computing a continuous relaxation to the *top-k-argmax* procedure in Algorithm 2 which takes as input a score matrix of size $k \times |V|$ and returns k peaked matrices p of size $k \times |V|$. Each matrix p_i represents the index of i-th max. For example, p_1 will have most of its mass concentrated on the index in the matrix that corresponds to the argmax, while p_2 will have most of its mass concentrated on the index of the 2nd-highest scoring element. Specifically, we obtain matrix p_i by computing the squared difference between the i-highest score and all the scores in the matrix and then using the *peaked-softmax* operation over the negative squared differences. This results in scores closer to the i-highest score to have a higher mass than scores far away from the i-highest score.

Hence, the continuous relaxation to *top-k-argmax* operation can be simply implemented by iteratively using the *max* operation which is continuous and allows for gradient flow during backpropagation. As $\alpha \to \infty$, each p vector converges to hard index pairs representing hard backpointers and successor candidates described in Algorithm 1. For finite α, we introduce a notion of a soft backpointer, represented as a vector $\tilde{b} \in \mathbb{R}^k$ in the k-probability simplex, which represents the contribution of each beam element from the previous time step to a beam element at current time step. This is obtained by a row-wise sum over p to get k values representing soft backpointers.

2.3 Training with Continuous Relaxation of Beam Search

We describe our approach in detail in Algorithm 3 and illustrate the soft beam recurrence step in Figure 1. For composing the loss function and the beam search function for our optimization as proposed in Equation 2, we make use of decomposability of the loss function across time-steps. Thus for a sequence y, the total loss is: $L(y, y^*) = \sum_{t=1}^T d(y_t)$. In our experiments, $d(y_t)$ is the Hamming loss which can be easily computed at each time-step by simply comparing gold y_t^* with y_t. While exact compu-

Algorithm 3 Continuous relaxation to beam search

1: **Initialize:**
$\quad h_{0,i} \leftarrow \vec{0}, e_{0,i} \leftarrow embedding(\texttt{<s>}), s_{0,i} \leftarrow 0, D_t \in \mathbb{R}^k \leftarrow \vec{0}, i = 1, \dots, k$
2: **for** t = 0 to T **do**
3: $\quad$ **for all** $w \in V$ **do**
4: $\quad\quad$ **for** i=1 to k **do**
5: $\quad\quad\quad \tilde{s}_t[i, w] \leftarrow s_{t,i} + f(h_{t,i}, w)$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ f is a local output scoring function
6: $\quad\quad\quad \tilde{D}_{t,w} = d(w)$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ $\tilde{D}_t$ is used to compute D_{t+1}
7: $\quad p_1, \dots, p_k \leftarrow continuous\text{-}top\text{-}k\text{-}argmax(\tilde{s}_t)$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Call Algorithm 2
8: $\quad$ **for** i = 1 to k **do**
9: $\quad\quad \tilde{b}_{t,i} \leftarrow row\_sum(p_i)$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Soft back pointer computation
10: $\quad\quad a_i \in \mathcal{R}^{|V|} \leftarrow column\_sum(p_i)$ $\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ Contribution from vocabulary items
11: $\quad\quad e_{t+1,i} \leftarrow a_i^T \times E$ $\qquad\qquad\qquad\qquad\qquad$ ▷ Peaked distribution over the candidates to compute e, D, S
12: $\quad\quad D_{t+1,i} \leftarrow a_i^T \cdot \tilde{D}_t$
13: $\quad\quad s_{t+1,i} = sum(\tilde{s}_t \odot p_i)$
14: $\quad\quad \tilde{h}_{t,i} \leftarrow \vec{0}$
15: $\quad\quad$ **for** j = 1 to k **do** $\qquad\qquad\qquad\qquad\qquad$ ▷ Get contributions from soft backpointers for each beam element
16: $\quad\quad\quad \tilde{h}_{t,i} + = h_{t,j} * \tilde{b}_{t,i}[j]$
17: $\quad\quad\quad D_{t+1,i} + = D_{t,j} * \tilde{b}_{t,i}[j]$
18: $\quad\quad h_{t+1,i} \leftarrow r(\tilde{h}_{t,i}, e_{t+1,i})$ $\qquad\qquad\qquad$ ▷ r is a nonlinear recurrent function that returns state at next step
19: $L = peaked\text{-}softmax_\alpha(s_T) \cdot D_T$ $\quad$ ▷ Pick the loss for the sequence with highest model score on the beam in a soft manner.

---

tation of $d(y)$ will vary according to the loss, our proposed procedure will be applicable as long as the total loss is decomposable across time-steps. While decomposability of loss is a strong assumption, existing literature on structured prediction (Taskar, Guestrin, and Koller 2004; Tsochantaridis et al. 2005) has made due with this assumption, often using decomposable losses as surrogates for non-decomposable ones. We detail the continuous relaxation to beam search in Algorithm 3 with $D_{t,i}$ being the cumulative loss of beam element i at time step t and E being the embedding matrix of the target vocabulary which is of size $|V| \times l$ where l is the size of the embedding vector.

In Algorithm 3, all the discrete selection functions have been replaced by their soft, continuous counterparts which can be backpropagated through. This results in all the operations being matrix and vector operations which is ideal for a GPU implementation. An important aspect of this algorithm is that we no longer rely on exactly identifying a discrete search prediction $\hat{y}$ since we are only interested in a continuous approximation to the direct loss L (line 18 of Algorithm 3), and all the computation is expressed via the *soft beam search* formulation which eliminates all the sources of discontinuities associated with the training objective in Equation 1. The computational complexity of our approach for training scales linearly with the beam size and hence is roughly k times slower than standard CE training for beam size k. Since we have established the pointwise convergence of *peaked-softmax* to argmax as $\alpha \to \infty$ for all vectors that have a unique maximum value, we can establish pointwise convergence of objective in Equation 2 to objective in Equation 1 as $\alpha \to \infty$, as long as there are no ties among the top-k scores of the beam expansion candidates at any time step. We posit that absolute ties are unlikely due to random

initialization of weights and the domain of the scores being $\mathbb{R}$. Empirically, we did not observe any noticeable impact of potential ties on the training procedure and our approach performed well on the tasks as discussed in Section 4.

$$\tilde{G}_{\mathrm{DL},\alpha}(x, \theta, y^*) \xrightarrow[p]{\alpha \to \infty} G_{\mathrm{DL}}(x, \theta, y^*) \qquad (3)$$

We experimented with different annealing schedules for α starting with *non-peaked softmax* moving toward *peaked-softmax* across epochs so that learning is stable with informative gradients. This is important because cost functions like Hamming distance with very high α tend to be non-smooth and are generally flat in regions far away from changepoints and have a very large gradient near the changepoints which makes optimization difficult.

2.4 Decoding

The motivation behind our approach is to make the optimization aware of beam search decoding while maintaining the continuity of the objective. However, since our approach doesn't introduce any new model parameters and optimization is agnostic to the architecture of the seq2seq model, we were able to experiment with various decoding schemes like locally normalized greedy decoding, and hard beam search, once the model has been trained.

However, to reduce the gap between the training procedure and test procedure, we also experimented with *soft beam search decoding*. This decoding approach closely follows Algorithm 3, but along with soft back pointers, we also compute hard back pointers at each time step. After computing all the relevant quantities like model score, loss etc., we follow the *hard* backpointers to obtain the best sequence $\hat{y}$. This is very different from hard beam decoding because at each time step, the selection decisions are made via our soft continuous
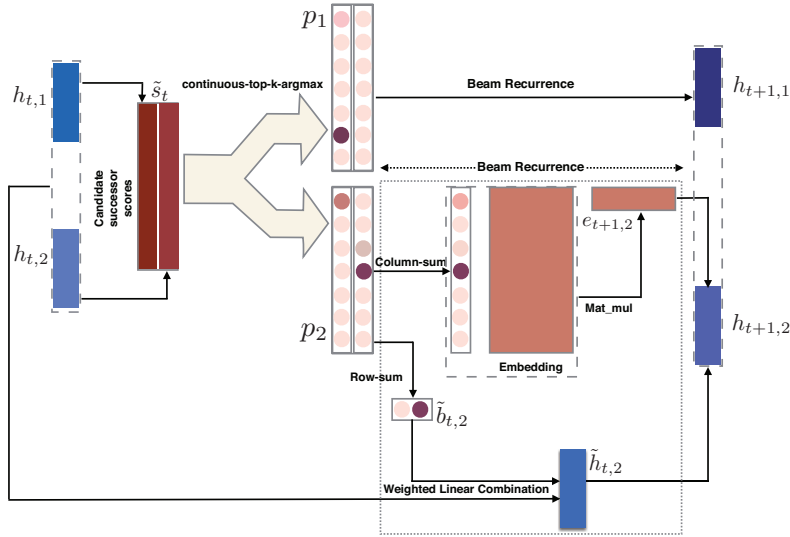
Figure 1: Illustration of our approximate continuous beam search (Algorithm 3) module to obtain hidden states for beam elements at the next time step ($h_{t+1,*}$), starting from the hidden states corresponding to beam elements are current time step ($h_{t,*}$) with beam size of 2. 'Beam recurrence' module has been expanded for $h_{t+1,2}$ and similar procedure is carried out for $h_{t+1,1}$.

relaxation which influences the scores, LSTM hidden states and input embeddings at subsequent time-steps. The hard backpointers are essentially the MAP estimate of the soft backpointers at each step. With small, finite α, we observe differences between soft beam search and hard beam search decoding in our experiments.

2.5 Comparison with Max-Margin Objectives

Max-margin based objectives are typically motivated as another kind of surrogate training objective which avoid the discontinuities associated with direct loss optimization. Hinge loss for structured prediction typically takes the form:

$$G_{hinge} = \max(0, \max_{y \in \mathcal{Y}}(\Delta(y, y^*) + s(y)) - s(y^*))$$

where x is the input sequence, y^* is the gold target sequence, $\mathcal{Y}$ is the output search space and $\Delta(y, y^*)$ is the discontinuous cost function which we assume is decomposable across the time-steps of a sequence. Finding the cost augmented maximum score is generally difficult in large structured models and often involves searching over the output space and computing the approximate cost augmented maximal output sequence and the score associated with it via beam search. This procedure introduces discontinuities in the training procedure of structured max-margin objectives and renders it non amenable to training via backpropagation. Related work (Wiseman and Rush 2016) on incorporating beam search into the training of neural sequence models does involve cost-augmented max-margin loss but it relies on discontinuous beam search forward passes and an explicit mechanism to ensure that the gold sequence stays in the beam during train-

ing, and hence does not involve back propagation through the beam search procedure itself.

Our continuous approximation to beam search can very easily be modified to compute an approximation to the structured hinge loss so that it can be trained via backpropagation if the cost function is decomposable across time-steps. In Algorithm 3, we only need to modify line 5 as:

$$\tilde{s}_t[i, w] \leftarrow s_{t,i} + d(w) + f(h_{t,i}, w)$$

and instead of computing L in Algorithm 3, we first compute the cost augmented maximum score as:

$$s_{max} = peaked\text{-}softmax_\alpha(s_T) \cdot s_T$$

and also compute the target score $s(y^*)$ by simply running the forward pass of the LSTM decoder over the gold target sequence. The continuous approximation to the hinge loss to be optimized is then: $\tilde{G}_{hinge,\alpha} = \max(0, s_{max} - s(y^*))$. We empirically compare this approach with the proposed approach to optimize direct loss in experiments.

3 Experimental Setup

Since our goal is to investigate the efficacy of our approach for training generic seq2seq models, we perform experiments on two NLP tagging tasks with very different characteristics and output search spaces: Named Entity Recognition (NER) and CCG supertagging. While seq2seq models are appropriate for CCG supertagging task because of the long-range correlations between the sequential output elements and a large search space, they are not ideal for NER which has a considerably smaller search space and weaker correlations

between predictions at subsequent time steps. In our experiments, we observe improvements from our approach on *both* of the tasks. We use a seq2seq model with a bi-directional LSTM encoder (1 layer with tanh activation function) for the input sequence x, and an LSTM decoder (1 layer with tanh activation function) with a fixed attention mechanism that deterministically attends to the i-th input token when decoding the i-th output, and hence does not involve learning of any attention parameters. Since, computational complexity of our approach for optimization scales linearly with beam size for each instance, it is impractical to use very large beam sizes for training. Hence, beam size for all the beam search based experiments was set to 3 which resulted in improvements on both the tasks as discussed in the results. For both tasks, the direct loss function was the *Hamming distance cost* which aims to maximize word level accuracy.

3.1 Named Entity Recognition

For named entity recognition, we use the CONLL 2003 shared task data (Tjong Kim Sang and De Meulder 2003) for German language and use the provided data splits. We perform no preprocessing on the data. The output vocabulary size (label space) is 10. A peculiar characteristic of this problem is that the training data is naturally skewed toward one default label ('O') because sentences typically do not contain many named entities and the evaluation focuses on the performance recognizing entities. Therefore, we modify the Hamming cost such that incorrect prediction of 'O' is doubly penalized compared to other incorrect predictions. We use the hidden layers of size 64 and label embeddings of size 8. As mentioned earlier, seq2seq models are not an ideal choice for NER (tag-level correlations are short-ranged in NER – the unnecessary expressivity of full seq2seq models over simple encoder-classifier neural models makes training harder). However, we wanted to evaluate the effectiveness of our approach on different instantiations of seq2seq models.

3.2 CCG Supertagging

We used the standard splits of CCG bank (Hockenmaier and Steedman 2002) for training, development, and testing. The label space of supertags is 1,284 which is much larger than NER. The distribution of supertags in the training data exhibits a long tail because these supertags encode specific syntactic information about the words' usage. The supertag labels are correlated with each other and many tags encode similar information about the syntax. Moreover, this task is sensitive to the long range sequential decisions and search effects because of how it holistically encodes the syntax of the entire sentence. We perform minor preprocessing on the data similar to the preprocessing in Vaswani et al. (2016). For this task, we used hidden layers of size 512 and the supertag label embeddings were also of size 512. The standard evaluation metric for this task is the word level label accuracy which directly corresponds to Hamming loss.

3.3 Hyperparameter tuning

For tuning all the hyperparameters related to optimization we trained our models for 50 epochs and picked the models with the best performance on the development set. We also ran multiple random restarts for all the systems evaluated to account for performance variance across randomly started runs. We pretrained all our models with standard cross entropy training which was important for stable optimization of the non convex neural objective with a large parameter search space. This warm starting is a common practice in prior work on complex neural models (Ranzato et al. 2016; Rush, Chopra, and Weston 2015; Bengio et al. 2015).

3.4 Comparison

We report performance on validation and test sets for both the tasks in Tables 1 and 2. The baseline model is a cross entropy trained seq2seq model (Baseline CE) which is also used to warm start the the proposed optimization procedures in this paper. This baseline has been compared against the approximate direct loss training objective (Section 2.3), referred to as $\tilde{G}_{\mathrm{DL},\alpha}$ in the tables, and the approximate max-margin training objective (Section 2.5), referred to as $\tilde{G}_{\mathrm{hinge},\alpha}$ in the tables. Results are reported for models when trained with annealing α, and also with a constant setting of $\alpha = 1.0$ which is a very smooth but inaccurate approximation of the original direct loss that we aim to optimize[2]. Comparisons have been made on the basis of performance of the models under different decoding paradigms (represented as different column in the tables): locally normalized decoding (CE greedy), hard beam search decoding and soft beam search decoding described in Section 2.4.

4 Results

As shown in Tables 1 and 2, our approach $\tilde{G}_{\mathrm{DL},\alpha}$ shows significant improvements over the locally normalized CE baseline with greedy decoding for both the tasks (+5.5 accuracy points gain for supertagging and +1.5 F1 points for NER). The improvement is more pronounced on the supertagging task, which is not surprising because: (i) the evaluation metric is tag-level accuracy which is congruent with the Hamming loss that $\tilde{G}_{\mathrm{DL},\alpha}$ directly optimizes and (ii) the supertagging task itself is very sensitive to the search procedure because tags across time-steps tend to exhibit long range dependencies as they encode specialized syntactic information about word usage in the sentence.

Another common trend to observe is that annealing α always results in better performance than training with a constant $\alpha = 1.0$ for both $\tilde{G}_{\mathrm{DL},\alpha}$ (Section 2.3) and $\tilde{G}_{\mathrm{hinge},\alpha}$ (Section 2.5). This shows that a stable training scheme that smoothly approaches minimizing the actual direct loss is important for our proposed approach. Additionally, we did not observe a large difference when our soft approximation is used for decoding (Section 2.4) compared to hard beam search decoding, which suggests that our approximation to the hard beam search is as effective as its discrete counterpart.

For supertagging, we observe that the baseline cross entropy trained model improves its predictions with beam search decoding compared to greedy decoding by 2 accuracy

---

[2]Our pilot experiments that involved training with a very large constant α resulted in unstable optimization.

Training procedure	Greedy		Hard Beam Search		Soft Beam Search	
	Dev	Test	Dev	Test	Dev	Test
Baseline CE	80.15	80.35	82.17	82.42	81.62	82.00
$\tilde{G}_{\text{hinge},\alpha}$ annealed α	-	-	83.03	83.54	82.82	83.05
$\tilde{G}_{\text{hinge},\alpha}\,\alpha$=1.0	-	-	83.02	83.36	82.49	82.85
$\tilde{G}_{\text{DL},\alpha}\,\alpha$=1.0	-	-	83.23	82.65	82.58	82.82
$\tilde{G}_{\text{DL},\alpha}$ annealed α	-	-	**85.69**	**85.82**	85.58	85.78

Table 1: Results on CCG Supertagging. Tag-level accuracy is reported in this table which is a standard evaluation metric for supertagging.

Training procedure	CE Greedy		Hard Beam Search		Soft Beam Search	
	Dev	Test	Dev	Test	Dev	Test
Baseline CE	50.21	54.92	46.22	51.34	47.50	52.78
$\tilde{G}_{\text{hinge},\alpha}$ annealed α	-	-	41.10	45.98	41.24	46.34
$\tilde{G}_{\text{hinge},\alpha}\,\alpha$=1.0	-	-	40.09	44.67	39.67	43.82
$\tilde{G}_{\text{DL},\alpha}\,\alpha$=1.0	-	-	49.88	54.08	50.73	54.77
$\tilde{G}_{\text{DL},\alpha}$ annealed α	-	-	51.86	56.15	**51.96**	**56.38**

Table 2: Results on Named Entity Recognition. Macro F1 over the prediction of different named entities is reported that is a standard evaluation metric for this task.

points, which suggests that beam search is already helpful for this task, even without search-aware training. Both the optimization schemes proposed in this paper improve upon the baseline with soft direct loss optimization ($\tilde{G}_{\text{DL},\alpha}$), performing better than the approximate max-margin approach.[3]

For NER, we observe that optimizing $\tilde{G}_{\text{DL},\alpha}$ outperforms all the other approaches but we also observe interesting behaviour of beam search decoding and the approximate max-margin objective for this task. The pretrained CE baseline model yields worse performance when beam search is done instead of greedy locally normalized decoding. This is because the training data is heavily skewed toward the 'O' label and hence the absolute score resolution between different tags at each time-step during decoding isn't enough to avoid leading beam search toward a wrong hypothesis path. We observed in our experiments that hard beam search resulted in predicting more 'O's which also hurt the prediction of tags at future time steps and hurt precision as well as recall. Encouragingly, $\tilde{G}_{\text{DL},\alpha}$ optimization, even though warm started with a CE trained model that performs worse with beam search, led to the NER model becoming more search aware, which resulted in superior performance. However, we also observe that the approximate max-margin approach ($\tilde{G}_{\text{hinge},\alpha}$) performs poorly here. We attribute this to a deficiency in the max-margin objective when coupled with approximate

search methods like beam search that do not provide guarantees on finding the supremum: one way to drive this objective down is to learn model scores such that the search for the best hypothesis is difficult, so that the value of the loss augmented decode is low, while the gold sequence maintains higher model score. Because we also warm started with a pre-trained model that results in a worse performance with beam search decode than with greedy decode, we observe the adverse effect of this deficiency. The result is a model that scores the gold hypothesis highly, but yields poor decoding outputs. This observation indicates that using max-margin based objectives with beam search during *training* actually may achieve the opposite of our original intent: the objective can be driven down by *introducing* search errors.

The observation that our optimization method led to improvements on *both* the tasks–even on NER for which hard beam search during decoding on a CE trained model hurt the performance–by making the optimization more search aware, indicates the effectiveness of our approach for training seq2seq models.

5   Conclusion

While beam search is a method of choice for performing search in neural sequence models, as our experiments confirm, it is not necessarily guaranteed to improve accuracy when applied to cross-entropy-trained models. In this paper, we propose a novel method for optimizing model parameters that directly takes into account the process of beam search itself through a continuous, end-to-end sub-differentiable relaxation of beam search composed with the final evaluation loss. Experiments demonstrate that our method is able to improve overall test-time results for models using beam search as a test-time inference method, leading to substantial

---

[3]Separately, we also ran experiments with a max-margin objective that used hard beam search to compute loss-augmented decodes. This objective is discontinuous, but we evaluated the performance of gradient optimization nonetheless. While not included in the result tables, we found that this approach was unstable and considerably underperformed both approximate max-margin and direct loss objectives.

improvements in accuracy.

6   Acknowledgement

References

Andor, D.; Alberti, C.; Weiss, D.; Severyn, A.; Presta, A.; Ganchev, K.; Petrov, S.; and Collins, M. 2016. Globally normalized transition-based neural networks. In *Association for Computational Linguistics*.

Bahdanau, D.; Serdyuk, D.; Brakel, P.; Ke, N. R.; Chorowski, J.; Courville, A.; and Bengio, Y. 2016. Task loss estimation for structured prediction.

Bahdanau, D.; Brakel, P.; Xu, K.; Goyal, A.; Lowe, R.; Pineau, J.; Courville, A.; and Bengio, Y. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 1171–1179.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Collins, M., and Roark, B. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 111. Association for Computational Linguistics.

Daumé III, H., and Marcu, D. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, 169–176. ACM.

Daumé, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325.

Dyer, C.; Ballesteros, M.; Ling, W.; Matthews, A.; and Smith, N. A. 2015. Transition-based dependency parsing with stack long short-term memory. In *Association for Computational Linguistics*.

Dyer, C.; Kuncoro, A.; Ballesteros, M.; and Smith, N. A. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, 199–209.

Gormley, M. R.; Dredze, M.; and Eisner, J. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics (TACL)*.

Goyal, K.; Dyer, C.; and Berg-Kirkpatrick, T. 2017. Differentiable scheduled sampling for credit assignment. In *Association for Computational Linguistics*.

Gu, J.; Im, D. J.; and Li, V. O. 2017. Neural machine translation with gumbel-greedy decoding. *arXiv preprint arXiv:1706.07518*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hockenmaier, J., and Steedman, M. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *LREC*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Koehn, P., and Knowles, R. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Neubig, G. 2017. Neural machine translation and sequence-to-sequence models: A tutorial. *arXiv preprint arXiv:1703.01619*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Ross, S.; Gordon, G. J.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, volume 1, 6.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing*.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Taskar, B.; Guestrin, C.; and Koller, D. 2004. Max-margin markov networks. In *Advances in neural information processing systems*, 25–32.

Tjong Kim Sang, E. F., and De Meulder, F. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, 142–147. Association for Computational Linguistics.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of machine learning research* 6(Sep):1453–1484.

Vaswani, A.; Bisk, Y.; Sagae, K.; and Musa, R. 2016. Supertagging with lstms. In *Proceedings of NAACL-HLT*, 232–237.

Wiseman, S., and Rush, A. M. 2016. Sequence-to-sequence learning as beam-search optimization. In *Empirical Methods in Natural Language Processing*.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, 77–81.