Sequence Level Training with Recurrent Neural Networks

Presentation by Alex Caiqi Zhang

Original Paper: [arxiv]

What We Can Learn

- How to combine RNN with REINFORCE in text generation.
    - To solve exposure bias and word level loss function.
- How the authors derive new models from existing ones step by step.
    - XENT → DAD → E2E → MIXER
    - **Beneficial to our own research.**

Backgrounds

Task: **Text Generation**

- Text summarization
- Machine translation
- Image captioning

Popular Models (back to 2016):

- N-grams [1]
- Feed-forward neural networks [2]
- RNN [3]

[1] Improved backing-off for M-gram language modeling, Kneser & Ney, 1995
[2] Hierarchical probabilistic neural network language model, Morin & Bengio, 2005
[3] Recurrent neural network based language model, Mikolov et al., 2010

Drawbacks of the Current Models

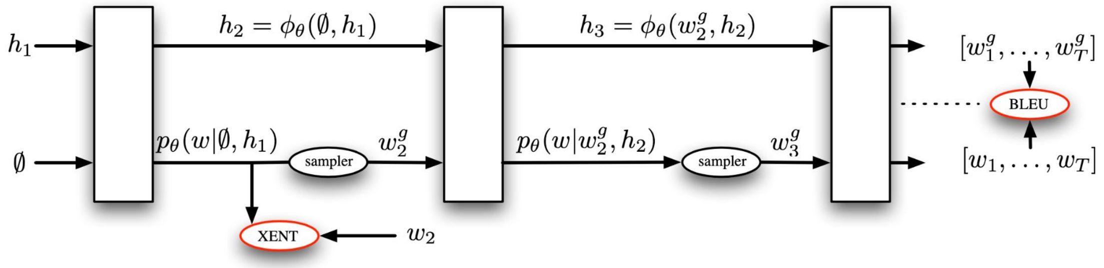
- Exposure bias
  - Models are **trained** to predict the next word **given the previous ground truth words** as input.
  - Models are **tested** to generate an entire sequence by **feeding the generated words** as input.
- Word level training loss function
  - Popular choice: the cross-entropy loss used to maximize the probability of **the next correct word.**
  - But results are evaluated in sequence level.

Consequences of the Drawbacks

- Exposure bias
    - Different distribution of inputs: **words drawn from the data distribution** VS **words drawn from the model distribution.**
    - Errors may accumulate along the way.
- Word level training loss function
    - Hard to optimize the whole sequence.

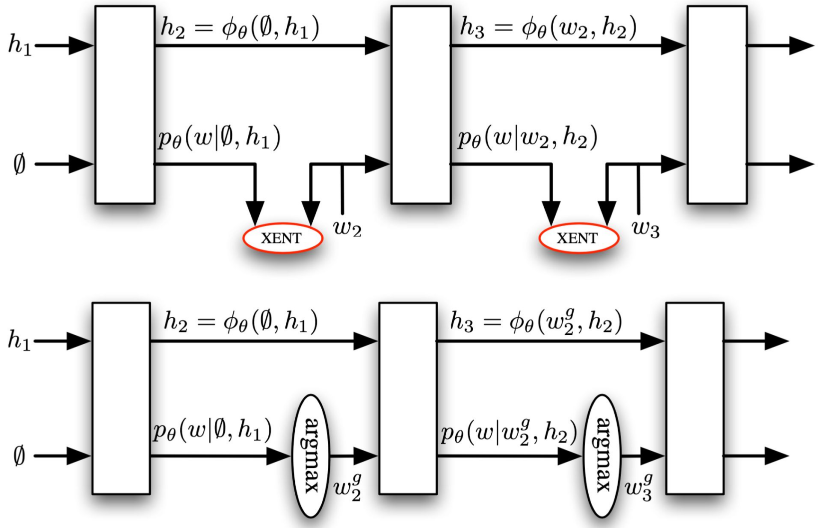
Proposed Solution: MIXER

- Mixed Incremental Cross-Entropy Reinforce
- Two basic ideas:
  - Incremental learning
  - Hybrid loss function which combines both REINFORCE and cross-entropy
- Advantages:
  - ✅Avoids exposure bias
  - ✅Sequence level training
  - ✅End to end



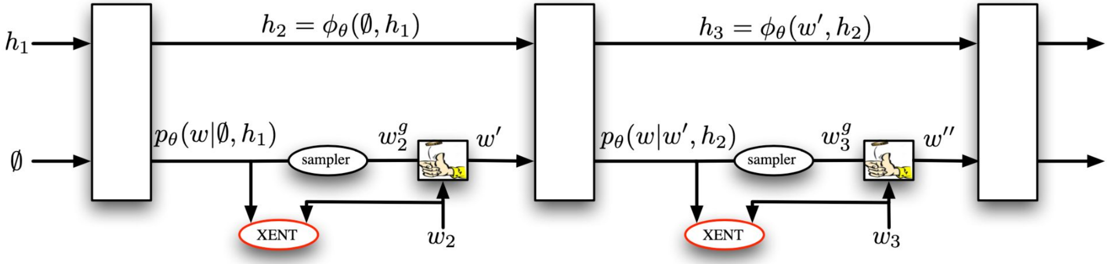
Cross Entropy Training (XENT)

- The model learns to **greedily** predict the the next word at each time step (without considering the whole sequence).
- Predictions are produced by either taking the argmax or by sampling from the distribution over words.
- Properties:
  - ❌Avoids exposure bias
  - ❌Sequence level training
  - ❌End to end

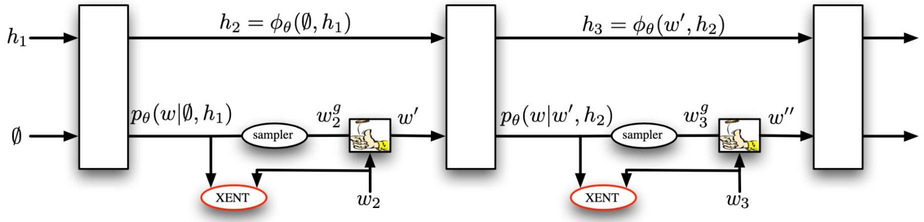


Data As Demonstrator (DAD)

- Addresses *exposure bias* by **mixing the ground truth training data with model predictions.**
- At each time step and with a certain probability takes as input:
  - the prediction from the model at the previous time step
  - the ground truth data
- Properties:
  - ✅Avoids exposure bias
  - ❌Sequence level training
  - ❌End to end



Data As Demonstrator (DAD)



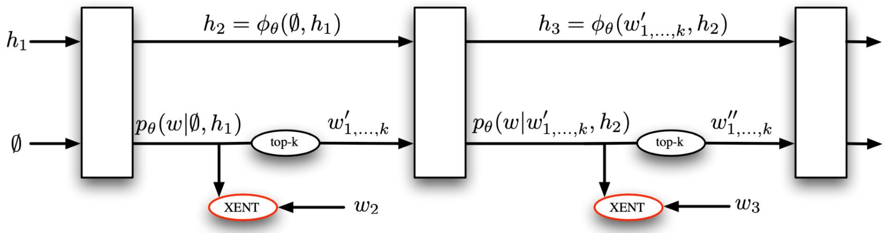
- Annealing schedules:
  - At the beginning, the algorithm **always** chooses the **ground truth words**.
  - As the training progresses the **model predictions** are selected **more often**.
  - This has the effect of making the model somewhat more aware of how it will be used at test time.

Drawbacks of DAD

- At every time step the target labels are always selected from the ground truth data, regardless of how the input was chosen
  - The history of predicted words is not considered
  - E.g., Ground truth is: *I took a long walk*.
    - What we have now: *I took a walk …*
    - DAD will force the model to predict the word “walk” a second time
- Gradients are not back-propagated through the samples drawn by the model
- The XENT loss is still at the word level.

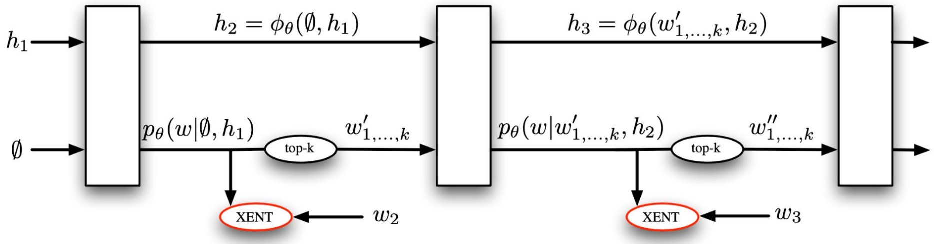
End-to-end Backprop (E2E)

- Perhaps the most natural and naïve approach approximating sequence level training
- Can also be interpreted as a computationally efficient approximation to **beam search**
- Properties:
  - ✅Avoids exposure bias
  - ✅End to end
  - ❌Sequence level training



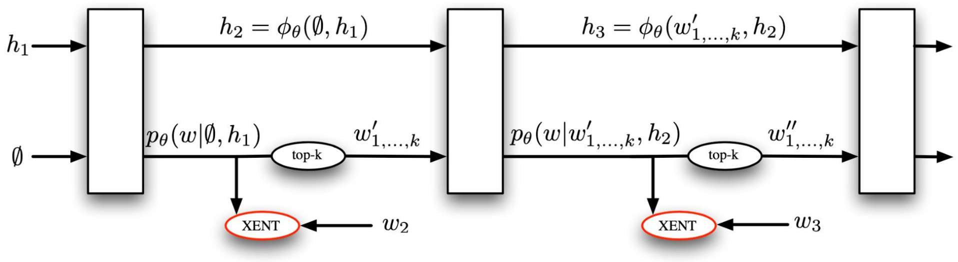
End-to-end Backprop (E2E)

- The key idea: at time step *t+1* we propagate as input the top *k* words predicted at the previous time step (instead of the ground truth word)
    - we take the output distribution over words from the previous time step t
    - pass it through a k-max layer
    - this layer zeros all but the k largest values and re-normalizes them to sum to one
- Compared to beam search, this can be interpreted as **fusing the *k* possible next hypotheses together into a single path.**
    - makes the whole process differentiable and trainable using standard back-propagation.



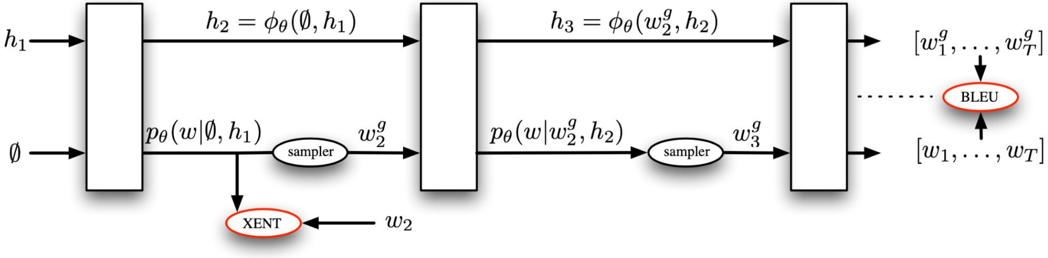
End-to-end Backprop (E2E)

- In practice we also employ a schedule
  - we use only the ground truth words at the beginning.
  - gradually let the model use its own top-k predictions as training proceeds.
- Properties (recap):
  - ✅Avoids exposure bias
  - ✅End to end
  - ❌Sequence level training



Mixed Incremental Cross-Entropy Reinforce (MIXER)

- The proposed method **avoids the exposure bias problem**, and also **directly optimizes for the final evaluation metric**.
- An extension of the REINFORCE algorithm
- Properties:
  - ✅Avoids exposure bias
  - ✅End to end
  - ✅Sequence level training



REINFORCE

- **Agent:** RNN model
- **Environment:** the words and the context vector it sees as input at every time step.
- **Policy:** defined by the parameters of this agent; results in the agent picking an action.
- **Action:** predicting the next word in the sequence at each time step.
- **Reward:** once the agent has reached the end of a sequence, it observes a reward
  - We can choose any reward function (e.g. BLEU/ROUGE-2)

REINFORCE Training

- We have a training set of optimal sequences of actions.
- During training we choose actions according to the current policy and only observe a reward at the end of the sequence (or after maximum sequence length).
  - comparing the sequence of actions from the current policy against the optimal action sequence results in reward
- The goal of training is to find the parameters of the agent that maximize the expected reward
  - we define our loss as the negative expected reward

Drawbacks of REINFORCE

- **Random policy to start with.**
  - This assumption can make the learning for large action spaces very challenging.
  - Text generation is such a setting where the cardinality of the action set is in the order of 10^4 (the number of words in the vocabulary).

From REINFORCE to MIXER

- First, change the initial policy of REINFORCE.
  - MIXER starts from the optimal policy and then slowly deviates from it to let the model explore and make use of its own predictions.
  - **Start off with a much better policy than random!**

Data: a set of sequences with their corresponding context.
Result: RNN optimized for generation.
Initialize RNN at random and set N^{XENT}, $N^{\text{XE+R}}$ and Δ;
for $s = T, 1, -\Delta$ **do**
  if $s == T$ **then**
    train RNN for N^{XENT} epochs using XENT only;
  else
    train RNN for $N^{\text{XE+R}}$ epochs. Use XENT loss in the first s steps, and REINFORCE (sampling from the model) in the remaining $T - s$ steps;
  end
end

Algorithm 1: MIXER pseudo-code.

TASK	N^{XENT}	$N^{\text{XE+R}}$	Δ
summarization	20	5	2
machine translation	25	5	3
image captioning	20	5	2

From REINFORCE to MIXER

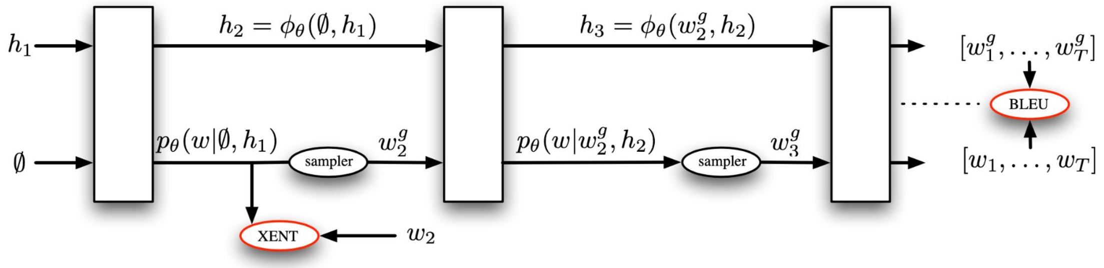
- Second, introduce model predictions during training with an annealing schedule in order to gradually teach the model to produce stable sequences.
- Repeat this process until **only** REINFORCE is used to train the whole sequence.

Data: a set of sequences with their corresponding context.
Result: RNN optimized for generation.
Initialize RNN at random and set N^{XENT}, $N^{\text{XE+R}}$ and Δ;
for $s = T, 1, -\Delta$ **do**
  if $s == T$ **then**
    train RNN for N^{XENT} epochs using XENT only;
  else
    train RNN for $N^{\text{XE+R}}$ epochs. Use XENT loss in the first s steps, and REINFORCE (sampling from the model) in the remaining $T - s$ steps;
  end
end

Algorithm 1: MIXER pseudo-code.

MIXER

- Mixed Incremental Cross-Entropy Reinforce
- Two basic ideas:
  - Incremental learning
  - Hybrid loss function which combines both REINFORCE and cross-entropy
- Advantages:
  - ✅Avoids exposure bias
  - ✅Sequence level training
  - ✅End to end



Experiments

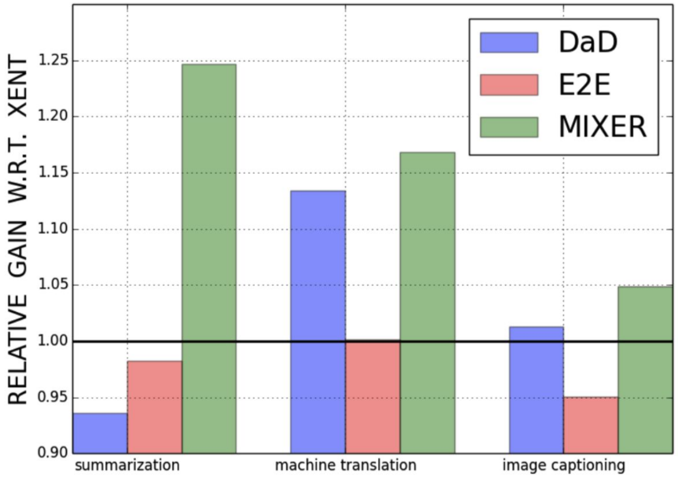
- Text summarization:
    - Model: conditional Elman RNN with 128 hidden units
    - Dataset: a subset of Gigaword corpus [4] as explained in [5]
- Machine translation:
    - Model: an LSTM with 256 hidden units
    - Dataset: German-English (IWSLT 2014)
- Image captioning:
    - Model: LSTM with 512 hidden units
    - Dataset: MSCOCO [6]

[4] English gigaword, Graff et al., Technical report, 2003

[5] A neural attention model for abstractive sentence summarization, Rush et al., EMNLP, 2015

[6] Microsoft coco: Common objects in context, Lin et al., Technical report, 2014

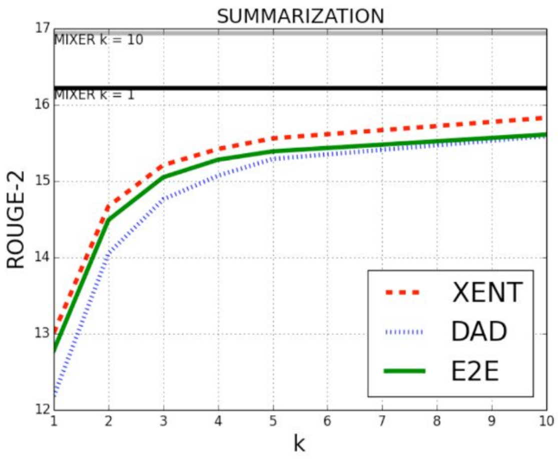
Experiments

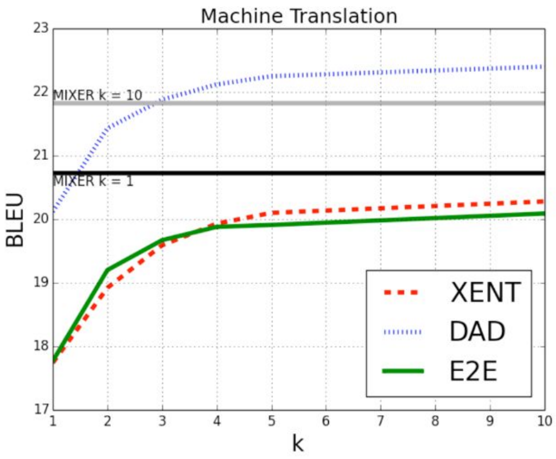


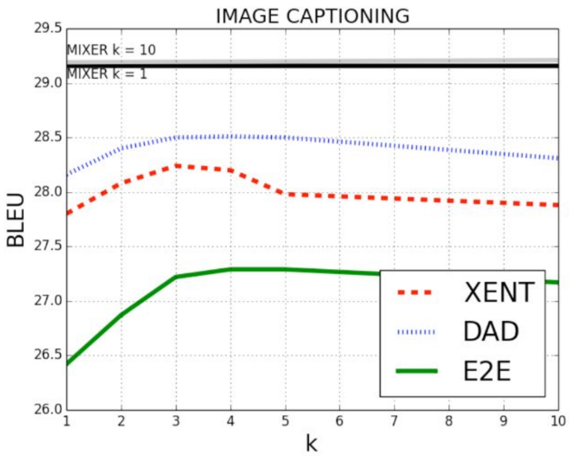
TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**

Beam Search Results

- Beam search **always improves** performance, although the amount depends on the task.
- Greedy performance of MIXER is **competitive** with baselines using beam search.
- MIXER is several times faster since it relies only on greedy search.







Limitations

- The actual performance of the model is highly dependent on the reliability of the test metrics.
  - BLEU often consider the overlap between generated and real sentences, while ignoring attributes such as semantic fluency and diversity of the sentences.
  - **High score → Low quality**
- The expected gradient computed using mini-batches under REINFORCE typically exhibit **high variance**, and without proper context-dependent normalization, is typically unstable [7].

[7] Rennie S J, Marcheret E, Mroueh Y, et al. Self-critical sequence training for image captioning, 2017

Thank you!