Discreteness in Neural Natural Language Processing

Lili Mou\textsuperscript{a} Hao Zhou\textsuperscript{b} Lei Li\textsuperscript{b}

\textsuperscript{a}Alberta Machine Intelligence Institute (Amii), University of Alberta
\textsuperscript{b}ByteDance AI Lab

doublepower.mou@gmail.com
{zhouhao.nlp,lileilab}@bytedance.com

EMNLP-IJCNLP 2019 Tutorial
Part IV: Discrete Output Space
Roadmap

• Examples of discrete output space

• Challenges and Solutions of Discrete Output Space
  - From Continuous Outputs to Discrete Outputs
    ▶ Embedding Matching by Softmax
  - Non-differentiable: Difficult for non-MLE training (e.g., GAN)
    ▶ RL for Generation
    ▶ Gumbel Softmax for Generation
  - Exponential Search Space
    ▶ Hard for Global Inference
    ▶ Hard for Constrained Decoding

• Case Study
  - Kernelized Bayesian Softmax
  - SeqGAN
  - Constrained Sentence Generation with CGMH
Outputs of NLP Tasks

\[ X \xrightarrow{\text{neural networks}} P(Y|X) \rightarrow Y \]

\{ Word, Sentence, Tree, Graph, ... \}
Outputs of NLP Tasks

More complex discrete outputs such as sequence, tree or graph structures exit in NLP.
Output Sentences

Machine Writing

ChatBOT

Question Answering

Machine Translation
Output Trees

Roadmap

• Examples of discrete output space

• Challenges and Solutions of Discrete Output Space
  - From Continuous Outputs to Discrete Outputs
    ▶ Embedding Matching by Softmax
  - Non-differentiable: Difficult for non-MLE training (e.g., GAN)
    ▶ RL for Generation
    ▶ Gumbel Softmax for Generation
  - Exponential Search Space
    ▶ Hard for Global Inference
    ▶ Hard for Constrained Decoding

• Case Study
  - Kernelized Bayesian Softmax
  - SeqGAN
  - Constrained Sentence Generation with CGMH
Challenges of the Discrete Output Space

Discrete outputs, especially the discrete sequence/structure outputs are non-trivial for handling in neural NLP.
Challenges of the Discrete Output Space

Discrete outputs, especially the discrete sequence/structure outputs are non-trivial for handling in neural NLP.

- From Continuous Outputs to Discrete Outputs
- Non-differentiable: fine for MLE but Non-trivial for other Training such as GAN
- Exponential Search Space
  - Hard for Global Inference
  - Hard for Constrained Decoding
Challenges of the Discrete Output Space

Discrete outputs, especially the discrete sequence/structure outputs are non-trivial for handling in neural NLP.

- From Continuous Outputs to Discrete Outputs
- Non-differentiable: fine for MLE but Non-trivial for other Training such as GAN
- Exponential Search Space
  - Hard for Global Inference
  - Hard for Constrained Decoding

In the next part, we will explain these challenges in detail and give some solutions.
Roadmap

• Examples of discrete output space

• Challenges and Solutions of Discrete Output Space
  – From Continuous Outputs to Discrete Outputs
    ▶ Embedding Matching by Softmax
  – Non-differentiable: Difficult for non-MLE training (e.g., GAN)
    ▶ RL for Generation
    ▶ Gumbel Softmax for Generation
  – Exponential Search Space
    ▶ Hard for Global Inference
    ▶ Hard for Constrained Decoding

• Case Study
  – Kernelized Bayesian Softmax
  – SeqGAN
  – Constrained Sentence Generation with CGMH
From Continuous to Discrete Outputs

\[ P(Y|X) \]
From Continuous to Discrete Outputs

How to transform continuous outputs to discrete Y?
Embedding Matching by Softmax

A simple sentiment classification case:
Embedding Matching by Softmax

A simple sentiment classification case:

\[ P(Y|X) = \text{softmax}(o \times W) \]
MLE for Training

Maximum Likelihood Estimation:

\[
\min \mathbb{E}_{<X,Y> \sim p_{\text{data}}} \left[ -\log p_{\theta}(Y | X) \right]
\]

\[
p_{\theta}(Y' | X) = \frac{\sigma(Y' | X)}{\sum_{Y} \sigma(Y | X)}
\]

Partition function: two possibilities, namely, positive or negative.
How about Sequence

\[
X \implies P(Y|X) \implies Y = \text{I like this tutorial}
\]
Exponential Hypothesis Space!

Maximum Likelihood Estimation:

\[
\min \mathbb{E}_{<X,Y> \sim p_{data}} [-\log p_\theta(Y | X)]
\]

\[
p_\theta(Y') = \frac{\sigma(Y' | X)}{\sum_Y \sigma(Y | X)}
\]

Calculating partition function directly requires exponential time!

\[
\mathcal{V} \times \mathcal{V} \times \mathcal{V} \times \mathcal{V} = \mathcal{V}^4
\]

I like this tutorial
Locally Normalized Factorization

• Directed, fully-observed Bayesian network:

Decompose the joint distribution as a product of tractable conditionals:

Given \( Y = [y_1, y_2, y_3 \ldots, y_n] \)

\[
p_\theta(Y) = \prod_{i=1}^{n} p_\theta(y_i | y_1, y_2, \ldots, y_{i-1}) = \prod_{i=1}^{n} p_\theta(y_i | y_{<i})
\]
Exponential Hypothesis Space!

Maximum Likelihood Estimation:

\[
\min \mathbb{E}_{<X,Y> \sim p_{\text{data}}} [-\log p_\theta(Y | X)]
\]

\[
p_\theta(Y') = \frac{\sigma(Y' | X)}{\sum_Y \sigma(Y | X)}
\]

Calculating partition function directly requires exponential time!

But, under certain model structure, it is possible to computing within polynomial time.
Tractable for Computing by Step by Step Factorization

\[ p_{\theta}(y_i' \mid y_{<i}, X) = \frac{\sigma(y_i' \mid y_1 \cdots y_{i-1}, X)}{\sum_{y_i'} \sigma(y_i \mid y_1 \cdots y_{i-1}, X)} \]

Vocabulary Size

Tractable for computing
Parameterization by Neural Networks

\[ p_\theta(y'_i | y_{<i}, X) = \frac{\sigma(y'_i | y_1 \ldots y_{i-1}, X)}{\sum_{y'_i} \sigma(y_i | y_1 \ldots y_{i-1}, X)} \]

Parameterization by RNN
Text Generation as an Example

\[ p_\theta(y'_i \mid y_{<i}) \]
Softmax at Each Time Step

\[ p_\theta(y_i' \mid y_{<i}) \]
Embedding Matching inside the Softmax

\[ p_\theta(y'_i \mid y_{<i}) \]
BackPropagation

Cross Entropy Loss
Structures as Sequence Prediction

John has a dog. →

```
S
 /   
|    |
NP   VP
    / 
  NNP VBZ
     /   
    DT   NN
          
          (S (NP NNP ) NP (VP VBZ (NP DT NN ) NP ) VP . ) S
```

Linearizing the tree structure as a sequence of syntax labels.

Learning and Predicting Trees as a Sequence

Modeling the syntax parsing problem as a sequence to sequence prediction.

Roadmap

• Examples of discrete output space

• Challenges and Solutions of Discrete Output Space
  - From Continuous Outputs to Discrete Outputs
    ▶ Embedding Matching by Softmax
  - Non-differentiable: Difficult for non-MLE training (e.g., GAN)
    ▶ RL for Generation
    ▶ Gumbel Softmax for Generation
  - Exponential Search Space
    ▶ Hard for Global Inference
    ▶ Hard for Constrained Decoding

• Case Study
  - Kernelized Bayesian Softmax
  - SeqGAN
  - Constrained Sentence Generation with CGMH
Non-Differentiable Problem

\[
X \xrightarrow{\text{neural networks}} P(Y | X) \rightarrow Y
\]
Non-Differentiable

- Fine for MLE but Non-trivial for other Training such as GAN.
What's GAN?

Generative Adversarial Networks:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x))]$$
Generator vs. Discriminator

Generative Adversarial Networks:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x))]$$
Objective Revisit

Generative Adversarial Networks:

$$\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

$$= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x))]$$
Objective Revisit

Generative Adversarial Networks:

\[
\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]
\]
BackPropagation Fails

\[
\min_G \max_D L(D, G) = \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x))]
\]

Text is discrete, hard to propagate gradients from D to G!
Using RL or Gumbel Softmax

The same techniques used in dealing with the latent space such as RL or Gumbel softmax could also be adopted for handling the discrete output space.
Roadmap

• Examples of discrete output space

• Challenges and Solutions of Discrete Output Space
  – From Continuous Outputs to Discrete Outputs
    ▶ Embedding Matching by Softmax
  – Non-differentiable: Difficult for non-MLE training (e.g., GAN)
    ▶ RL for Generation
    ▶ Gumbel Softmax for Generation
  – Exponential Search Space
    ▶ Hard for Global Inference
    ▶ Hard for Constrained Decoding

• Case Study
  – Kernelized Bayesian Softmax
  – SeqGAN
  – Constrained Sentence Generation with CGMH
Exponential Hypothesis Space

Figure from Liang Huang’s slides
Hard for Global Inference

- Inference for decoding
  - Hard to yield the best scored output in the exponential space
- Inference in training (globally normalized model)
  - Non-trivial to compute the partition function
Inference for Decoding

\[
\arg\max_Y \log p_\theta(Y | X) = \arg\max_Y \sum_{i=1}^n \log p_\theta(y_i | y_1, y_2, \ldots, y_{i-1}, X) = \arg\max_Y \sum_{i=1}^n \log p_\theta(y_i | y_{<i}, X)
\]

Exponential search space

Figure from Liang Huang’s slides
Beam Search

\[
\arg \max_{Y \in \text{BEAM}} \log p_\theta(Y | X) = \arg \max_{Y \in \text{BEAM}} \sum_{i=1}^{n} \log p_\theta(y_i | y_{<i}, X)
\]

Heuristic search by beam search

Figure from Liang Huang’s slides
Inference in Training

Maximum Likelihood Estimation:

\[ \min \mathbb{E}_{<X,Y> \sim p_{data}} \left[ -\log p_\theta(Y|X) \right] \]

\[ p_\theta(Y') = \frac{\sigma(Y'|X)}{\sum_Y \sigma(Y|X)} \]

Calculating partition function directly requires exponential time!
Approximated Globally Normalized Model

Maximum Likelihood Estimation:

\[ \min \ \mathbb{E}_{<X,Y> \sim p_{data}} \left[ -\log p_\theta(Y|X) \right] \]

\[ p_\theta(Y') = \frac{\sigma(Y'|X)}{\sum_Y \sigma(Y|X)} \]

\[ \sum_{Y \in BEAM} \sigma(Y|X) \]
Inference in Training

Maximum Likelihood Estimation:

\[ \min \mathbb{E}_{<X,Y> \sim p_{data}} \left[ -\log p_\theta(Y|X) \right] \]

\[ p_\theta(Y') = \frac{\sigma(Y'|X)}{\sum_Y \sigma(Y|X)} \]

Contrastive divergence using beam search as sampling
Global Normalized Structured Prediction:

\[
\min \mathbb{E}_{<X,Y> \sim p_{\text{data}}} [-\log p_{\theta}(Y \mid X)]
\]

\[
p_{\theta}(Y') = \frac{\sigma(Y' \mid X)}{\sum_{Y \in \text{BEAM}} \sigma(Y \mid X)}
\]

Contrastive divergence using beam search as sampling


Challenges of Discrete Output Structures

- From Continuous Outputs to Discrete Outputs
- Non-differentiable: fine for MLE but Non-trivial for other Training such as GAN
- Exponential Search Space
  ▶ Hard for Global Inference
  ▶ Hard for Constrained Decoding
Constrained Decoding:

\[
\arg \max_{Y} \quad p_{\theta}(Y | X), \\
\text{s.t.} \quad Y \text{ satisfy } C = \{C_1, C_2, \ldots, C_n\}
\]

The decoding outputs should satisfy a set of constraints.
Constraints Definition

• Generating sentence satisfying constraints:
  • Hard constrains: **Keyword must occur in sentences**
    – E.g. Juice -> Brand natural juice, specially made for you
Constraints Definition

• Generating sentence satisfying constraints:
  • Hard constrains: Keyword must occur in sentences
    – E.g. Juice -> Brand natural juice, specially made for you
  • Soft constrains: Semantically similar to a given sentence (paraphrase)
    – E.g. The movie is a great success -> It is one of my favorite movies
Beam search over the Search Space
Vanilla Beam Search Fails

Desired outputs satisfying constraints
Vanilla Beam Search Fails

Vanilla beam search may hardly find the desired outputs under specific constraints.
Advanced Approaches
Targets of Constrained Decoding

Target Distribution of Constrained Decoding:

$$\pi(Y) = \prod_i p_{\theta}(y_i \mid y_{<i}) \times \prod_{C \in \mathcal{C}} p_C(Y)$$

Density of the original model

Indicator functions for constraints
No Direct Sampling Method

Target Distribution of Constrained Decoding:

$$\pi(Y) = \prod_i p_{\theta}(y_i | y_{<i}) \times \prod_{C \in C} p_C(Y)$$

Density of the original model

Indicator functions for constraints

However, $\pi(Y)$ is quite high dimensional, and no direct sampling method.
The constrained decoding problem turns to be sampling instances from a high dimensional distribution.
Generation by Sampling

The constrained decoding problem turns to be sampling instances from a high dimensional distribution.

Main Idea of CGMH

• Instead of sampling from $\pi(x)$ directly, generating samples iteratively:
  – Starting with initial keywords
  – Next sentence based on modification of previous
  – Action proposals to modify the sentences

• Metropolis-Hastings Algorithm

Generation by Local Changes

- Suppose we have a blueprint

The book is interesting <EOS>
Generation by Local Changes

• Suppose we have a blueprint

The book is interesting <EOS>

This
Generation by Local Changes

• Suppose we have a blueprint

   The book is interesting <EOS>

   This book is quite interesting <EOS>
Generation by Local Changes

• Suppose we have a blueprint

The book is interesting <EOS>

This book is interesting <EOS>

fascinating

quite
Metropolis Hastings Sampling

Metropolis-Hastings (MH) perform sampling by first proposes a transition, and then accepts or rejects the transition.

$$A(x' | x_{t-1}) = \min(1, \frac{\pi(x') \cdot g(x_{t-1} | x')}{{\pi(x_{t-1}) \cdot g(x' | x_{t-1})}})$$

$g$ is proposal distribution
Metropolis—Hastings Sampler

- **Algorithm**
  - Start from an *arbitrary* initial state $x^{(0)}$
  - For every step $t$
    - Propose a new state $x' \sim g(x' | x^{(t)})$
    - Accept $x'$ w.p. $A(x' | x) = \min \left\{ 1, \frac{p(x')g(x^{(t)} | x')}{p(x)g(x' | x^{(t)})} \right\}$, i.e.,
      $$x^{(t+1)} = x'$$
    - Reject $x'$ otherwise, i.e., $x^{(t+1)} = x^{(t)}$
  - Return $x^{(t)}$ with a large $t$
CGMH performs constrained generation by:
1. Pretrain Language Model prob;
2. Start from a initial sentence;
3. Propose a new action and accept/reject the action.
We use MH algorithm to sample from $\pi(x)$

- From a sentence $x_{t-1}$, we propose an action on one word of $x_{t-1}$.
- Actions include:
  1. **Replacement**: change a word to another one
  2. **Insertion**: add a word
  3. **Deletion**: remove a word
CGMH: Acceptance Ratio

- Calculate the acceptance rate:
  \[ A(x'|x_{t-1}) = \min(1, \frac{\pi(x') \cdot g(x_{t-1}|x')}{\pi(x_{t-1}) \cdot g(x'|x_{t-1})}) \]

- Accept \( x' \) with probability \( A(x'|x_{t-1}) \)
Proof Sketch (Cont.)

- MH Sampler satisfies detailed balance

\[ \forall x, y, \text{ if } x \neq y, \quad p(x) \cdot \mathcal{T}_{x \rightarrow y} = p(x) \cdot g(y \mid x) \cdot \min \left\{ 1, \frac{p(y)g(x \mid y)}{p(x)g(y \mid x)} \right\} \quad (1) \]

\[ p(y) \cdot \mathcal{T}_{y \rightarrow x} = p(y) \cdot g(x \mid y) \cdot \min \left\{ 1, \frac{p(x)g(y \mid x)}{p(y)g(x \mid y)} \right\} \quad (2) \]

- W.L.O.G., we assume \( p(x)g(y \mid x) \geq p(y)g(x \mid y) \)

\[ (1) = p(y) \cdot g(x \mid y) \]

\[ (2) = p(y) \cdot g(x \mid y) \]

- \( \forall x, y, \text{ if } x = y, \quad p(x)\mathcal{T}_{x \rightarrow y} = p(y)\mathcal{T}_{y \rightarrow x} \) also holds
Case Study

- Embedding Matching by softmax
  - Kernelized Bayesian Softmax
- RL for Generation
  - SeqGAN
- Generation by Sampling
  - Constrained Sentence Generation with CGMH
  - Generating Adversarial Examples for Natural Languages
Case Study 1

• Embedding Matching by Softmax
  – Kernelized Bayesian Softmax

• RL for Generation
  – SeqGAN

• Generation by Sampling
  – Constrained Sentence Generation with CGMH
  – Generating Adversarial Examples for Natural Languages
Kernelized Bayesian Softmax
Kernelized Bayesian Softmax

KerBS: Kernelized Bayesian Softmax

\[ P(x_t = i) = \sum_{j \in 0,1,\ldots,N_i} P(x_t = s^j_i) \]

where \( P(x_t = s^j_i) = \frac{\exp(\mathcal{K}_{\theta_i}(h_t, w^j_i))}{\sum_k \sum_{r \in 0,1,\ldots,N_k} \exp(\mathcal{K}_{\theta_k}(h_t, w^r_k))} \)

\[ \mathcal{K}_\theta(h, e) = |h| \cdot |e| \cdot (a \exp(-\theta \cos(h, e)) - a) \]

Here \( h \) is hidden state, \( e \) is embedding, \( \theta \) is a parameter controlling the embedding variances of each sense and \( a = \frac{-\theta}{2(\exp(-\theta) + \theta - 1)} \) is a normalization factor.

Why KerBS?

Model capacity of softmax is not OK

Motivated by BERT, we may need context dependent embedding for text generation!
Text Generation as Matching

Text Generation is Embedding Matching

Context Independent Embedding

Context Dependent Embedding
Bottleneck of Text Generation

Bottleneck of text generation is the softmax

Embedding matrix in softmax should have larger capacity.
Visualization of BERT

- Multi-Sense & Varying Variances

Softmax can handle this situation
Visualization of BERT

- Multi-Sense & Varying Variances

Softmax can’t handle multisense.
Visualization of BERT

- **Multi-Sense & Varying Variances**

Softmax can’t handle multisense and varying variances.
KerBS - Multisense

Each word may have several senses. KerBS allocates a vector for each sense.
KerBS - Multisense

After getting the probabilities of each sense, KerBS sums up all sense probabilities of same word.

\[ P(x_t = i) = \sum_{j \in 0, 1, \ldots, N_i} P(x_t = s_i^j) \]
KerBS - Varying Variances

The distribution of each word’s output vectors have different variances. We use a variable kernel to represent varying variances.

\[
P(x_t = s_i^j) = \frac{\exp(\mathcal{K}_{\theta_i}(h_t, w_i^j))}{\sum_k \sum_{r \in \{0, 1, \ldots, N_k\}} \exp(\mathcal{K}_{\theta_k}(h_t, w_k^r))}
\]

\[
\mathcal{K}_\theta(h, e) = |h| |e| (a \exp(-\theta \cos(h, e)) - a)
\]

Note that when \(\theta \to 0\), \(\mathcal{K}_\theta(h, e) \to |h| |e| \cos(h, e)\), which is regular Euclidean norm!
KerBS - Varying Variances

The distribution of each word’s output vectors have different variances. We use a variable kernel to represent varying variances.

Figure 2: Kernel shapes of different $\theta$. 

(a) $\theta = -2$

(b) $\theta = 2$
How to decide the sense number of each word?

Dynamically change each word’s sense number while training. Delete senses that are less used. Add senses to words which are not well fitted.
Dynamic Allocation

**Distillation**

![Diagram showing dynamic allocation in distillation]

**KerBS**

- **Kernel**
- **Copy**
- **Softmax**
- **Aggregate**

**P_sense**

- apple: 0.6
- amazon: 0.02
- tiger: 0.0

**P_word**

- apple: 0.6
- amazon: 0.02
- tiger: 0.0

**Tuning**

- **Query embedding**
- **Get “apple” embedding**
- **Weighted sum**

![Diagram showing dynamic allocation in tuning]

- **To**
- **Eat**
- **An**
- **Apple**
- **Weighted sum**
Theoretical Guarantee

Lemma 1
KerBS has the ability to learn the multi-sense property. If the real distribution of context vectors consists of several disconnected clusters, KerBS will learn to represent as many as these clusters.

KerBS can capture the multi-sense property.

Lemma 2
KerBS has the ability to learn model variances. For distributions with larger variances, KerBS learns larger $\theta$.

KerBS can learn varying variances.
Experiments-Setting

We test KerBS on 3 text generation tasks:

1. **Machine Translation (MT)** is conducted on IWSLT’16 De-En, which contains 196k pairs of sentences for training.

2. **Language modeling (LM)** is included. Following previous work, we use a 300k, 10k and 30k subset of One-Billion-Word Corpus for training, validating and testing.

3. **Dialog generation (Dialog)** is also included. We employ the DailyDialog dataset for experiment.
# Main Results

## Table 1: Performance of KerBS on Seq2Seq.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Metrics</th>
<th>Seq2Seq</th>
<th>Seq2Seq+ MoS [Yang et al. 2018]</th>
<th>SeqSeq + KerBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>BLEU-4</td>
<td>25.91</td>
<td>26.45</td>
<td>27.28</td>
</tr>
<tr>
<td>LM</td>
<td>PPL</td>
<td>103.12</td>
<td>102.72</td>
<td>102.17</td>
</tr>
<tr>
<td>Dialog</td>
<td>BLEU-1</td>
<td>16.56</td>
<td>13.73</td>
<td>17.85</td>
</tr>
<tr>
<td></td>
<td>Human Eval.</td>
<td>1.24</td>
<td>1.04</td>
<td>1.40</td>
</tr>
</tbody>
</table>

## Table 2: Performance of KerBS on Transformer.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Metrics</th>
<th>Transformer</th>
<th>Transformer + MoS [Yang et al. 2018]</th>
<th>Transformer + KerBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT</td>
<td>BLEU-4</td>
<td>29.61</td>
<td>28.54</td>
<td>30.90</td>
</tr>
<tr>
<td>Dialog</td>
<td>BLEU-1</td>
<td>10.61</td>
<td>9.81</td>
<td>10.90</td>
</tr>
</tbody>
</table>
Case Study 2

- Greedy Embedding Matching
  - Kernelized Bayesian Softmax

- RL for Generation
  - SeqGAN

- Generation by Sampling
  - Constrained Sentence Generation with CGMH
  - Generating Adversarial Examples for Natural Languages
SeqGAN

Directly applying RL to use Discriminator outputs as reward for updating Generator.
BackPropagation Fails

- Sentence is discrete, BP fails in such case
  - RL
  - Gumbel Softmax

Variance of gradient is very large!
Hard for training :(
RL for Text Generation

GANs for text generation

Strategies to deal with discontinuity

Policy Gradient

Gumbel Softmax

GAN models

SeqGAN: First GAN on discrete sentence space.

RankGAN: Use rank information to mitigate gradient vanishing.

LeakGAN: Use feature extracted by D to guide G.

GumbelGAN: Use Gumbel-trick to handle discontinuity.

TextGAN: Use feature matching for training.

RelGAN: Build stronger D and G. The first practical Gumbel GAN.

LATEX-GAN: Combines Gumbel GAN and AAE

AAE (Adversarial Autoencoder)

ARAЕ: Perform GAN on embedding space.

LATEX-GAN: Combines Gumbel GAN and AAE
MLE Outperforms different GAN Variants

<table>
<thead>
<tr>
<th>Model</th>
<th>NLL$_{oracle}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeqGAN (Yu et al., 2017)</td>
<td>8.74</td>
</tr>
<tr>
<td>RankGAN (Lin et al., 2017)</td>
<td>8.25</td>
</tr>
<tr>
<td>LeakGAN (Guo et al., 2017)</td>
<td>7.04</td>
</tr>
<tr>
<td>IRL (Shi et al., 2018)</td>
<td>6.91</td>
</tr>
<tr>
<td>MLE ($\alpha = 1.0$)</td>
<td>9.40</td>
</tr>
<tr>
<td>MLE ($\alpha = 0.4$)</td>
<td>5.50</td>
</tr>
<tr>
<td>MLE ($\alpha = 0.001$)</td>
<td>4.58</td>
</tr>
</tbody>
</table>

Table 2: NLL$_{oracle}$ measured on the synthetic task (lower is better). All results are taken from their respective papers. An MLE-trained model with reduced temperature easily improves upon these GAN variants, producing the highest quality sample.

Figure 3: Effect of temperature tuning on the global metrics (lower is better for both metrics) for the synthetic task. The GAN cross-validated on quality only lies outside the figure because of severe mode collapse.

Case Study 3

- Greedy Embedding Matching
  - Kernelized Bayesian Softmax
- RL for Generation
  - SeqGAN
- Generation by Sampling
  - Constrained Sentence Generation with CGMH
  - Generating Adversarial Examples for Natural Languages
Advertisement Slogan by Constrained Generation

Keywords from Advertiser

Rin clothes bright

Advertisement Slogan

New Rin Keeps clothes bright like new
Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW, the sports car of future life</td>
</tr>
<tr>
<td>8</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>9</td>
<td>Delete</td>
<td>Reject</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>10</td>
<td>Delete</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>11</td>
<td>[Output]</td>
<td></td>
<td>BMW, the sports car of the future</td>
</tr>
</tbody>
</table>

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
</tbody>
</table>
Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
</tbody>
</table>

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
</tbody>
</table>
CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW , the sports car of daily life</td>
</tr>
</tbody>
</table>
CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW , the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW , the sports car of future life</td>
</tr>
</tbody>
</table>

Sampling in Sentence Space

CGMH performs Metropolis-Hastings **sampling directly in sentence space**:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW , the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW , the sports car of future life</td>
</tr>
<tr>
<td>8</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW , the sports car of the future life</td>
</tr>
</tbody>
</table>
## Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW, the sports car of future life</td>
</tr>
<tr>
<td>8</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>9</td>
<td>Delete</td>
<td>Reject</td>
<td>BMW, the sports car of the future life</td>
</tr>
</tbody>
</table>

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW, the sports car of future life</td>
</tr>
<tr>
<td>8</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>9</td>
<td>Delete</td>
<td>Reject</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>10</td>
<td>Delete</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
</tbody>
</table>

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Acc/Rej</th>
<th>Sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>[Input]</td>
<td></td>
<td>BMW sports</td>
</tr>
<tr>
<td>1</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW sports car</td>
</tr>
<tr>
<td>2</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW the sports car</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of daily life</td>
</tr>
<tr>
<td>7</td>
<td>Replace</td>
<td>Accept</td>
<td>BMW, the sports car of future life</td>
</tr>
<tr>
<td>8</td>
<td>Insert</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>9</td>
<td>Delete</td>
<td>Reject</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>10</td>
<td>Delete</td>
<td>Accept</td>
<td>BMW, the sports car of the future life</td>
</tr>
<tr>
<td>11</td>
<td>[Output]</td>
<td></td>
<td>BMW, the sports car of the future</td>
</tr>
</tbody>
</table>

## Cases of Keyword to Sentences

<table>
<thead>
<tr>
<th><strong>Keyword(s)</strong></th>
<th><strong>Generated Sentences</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>friends</td>
<td>My good <strong>friends</strong> were in danger.</td>
</tr>
<tr>
<td>project</td>
<td>The first <strong>project</strong> of the scheme.</td>
</tr>
<tr>
<td>have, trip</td>
<td>But many people <strong>have</strong> never made the <strong>trip</strong>.</td>
</tr>
<tr>
<td>lottery, scholarships</td>
<td>But the <strong>lottery</strong> has provided <strong>scholarships</strong>.</td>
</tr>
<tr>
<td>decision, build, home</td>
<td>The <strong>decision</strong> is to <strong>build</strong> a new <strong>home</strong>.</td>
</tr>
<tr>
<td>attempt, copy, painting, denounced</td>
<td>The first <strong>attempt</strong> to <strong>copy</strong> the <strong>painting</strong> was <strong>denounced</strong>.</td>
</tr>
</tbody>
</table>
## Paraphrase Generation

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-ref</th>
<th>BLEU-ori</th>
<th>NLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin Sentence</td>
<td>30.49</td>
<td>100.00</td>
<td>7.73</td>
</tr>
<tr>
<td>VAE-SVG (100k)</td>
<td>22.50</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VAE-SVG (100k)</td>
<td>22.90</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VAE-SVG (50k)</td>
<td>17.10</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VAE-SVG (50k)</td>
<td>17.40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Seq2seq (100k)</td>
<td>22.79</td>
<td>33.83</td>
<td>6.37</td>
</tr>
<tr>
<td>Seq2seq (50k)</td>
<td>20.18</td>
<td>27.59</td>
<td><strong>6.71</strong></td>
</tr>
<tr>
<td>Seq2seq (20k)</td>
<td>16.77</td>
<td>22.44</td>
<td>6.67</td>
</tr>
<tr>
<td>VAE (unsupervised)</td>
<td>9.25</td>
<td>27.23</td>
<td>7.74</td>
</tr>
<tr>
<td>CGMH w/o matching</td>
<td>18.85</td>
<td>50.28</td>
<td>7.52</td>
</tr>
<tr>
<td>w/ KW</td>
<td>20.17</td>
<td>53.15</td>
<td>7.57</td>
</tr>
<tr>
<td>w/ KW + WVA</td>
<td>20.41</td>
<td>53.64</td>
<td>7.57</td>
</tr>
<tr>
<td>w/ KW + WVM</td>
<td>20.89</td>
<td>54.96</td>
<td>7.46</td>
</tr>
<tr>
<td>w/ KW + ST</td>
<td>20.70</td>
<td>54.50</td>
<td>7.78</td>
</tr>
</tbody>
</table>

### Examples:

<table>
<thead>
<tr>
<th>Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ori</td>
<td>what’s the best plan to lose weight</td>
</tr>
<tr>
<td>Ref</td>
<td>what is a good diet to lose weight</td>
</tr>
<tr>
<td>Gen</td>
<td>what’s the best way to slim down quickly</td>
</tr>
<tr>
<td>Ori</td>
<td>how should i control my emotion</td>
</tr>
<tr>
<td>Ref</td>
<td>how do i control anger and impulsive emotions</td>
</tr>
<tr>
<td>Gen</td>
<td>how do i control my anger</td>
</tr>
<tr>
<td>Ori</td>
<td>why do my dogs love to eat tuna fish</td>
</tr>
<tr>
<td>Ref</td>
<td>why do my dogs love eating tuna fish</td>
</tr>
<tr>
<td>Gen</td>
<td>why do some dogs like to eat raw tuna and raw fish</td>
</tr>
</tbody>
</table>
Generating adversarial example for text is hard!
Because the text space is discrete, which is non-trivial to apply adversarial gradients!
CGMH for Generating Fluent Adversarial Examples

CGMH for Generating Fluent Adversarial Examples


(a) IMDB

(b) SNLI

CGMH for Generating Fluent Adversarial Examples

<table>
<thead>
<tr>
<th>Task</th>
<th>Approach</th>
<th>Succ(%)</th>
<th>Invok#</th>
<th>PPL</th>
<th>(\alpha(%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>Genetic</td>
<td>98.7</td>
<td>1427.5</td>
<td>421.1</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>(b)-MHA</td>
<td>98.7</td>
<td>1372.1</td>
<td>385.6</td>
<td>17.9</td>
</tr>
<tr>
<td></td>
<td>(w)-MHA</td>
<td>99.9</td>
<td>748.2</td>
<td>375.3</td>
<td>34.4</td>
</tr>
<tr>
<td>SNLI</td>
<td>Genetic</td>
<td>76.8</td>
<td>971.9</td>
<td>834.1</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>(b)-MHA</td>
<td>86.6</td>
<td>681.7</td>
<td>358.8</td>
<td>9.7</td>
</tr>
<tr>
<td></td>
<td>(w)-MHA</td>
<td>88.6</td>
<td>525.0</td>
<td>332.4</td>
<td>13.3</td>
</tr>
</tbody>
</table>

Conclusion of the Tutorial
Conclusion of the Tutorial

- Neural networks are good
- Natural language is discrete (Input, latent, output spaces)
  - Representation learning
  - Non-differentiability
  - Exponential search space
Thank You