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
Slides available at the instructors’ homepage

https://lili-mou.github.io/
1. Why do we need this tutorial?

2. What can you learn from this tutorial?
Why this tutorial?

- Deep learning has almost dominated NLP these years.
Why this tutorial?

- Deep learning has almost dominated NLP these years

- Different from speech and images, natural language units (word, sentence, paragraph, etc.) are **discrete**

- May cause problems in neural NLP
What could we learn from this tutorial?

- We will give examples of discreteness in neural NLP, including input, latent and output spaces.
What could we learn from this tutorial?

• We will give examples of discreteness in neural NLP, including input, latent and output spaces.

• We will introduce advanced techniques to address the discreteness problem.
What could we learn from this tutorial?

- We will give examples of discreteness in neural NLP, including input, latent and output spaces.
- We will introduce advanced techniques to address the discreteness problem.
- Cases will be finally studied to show how we can use these techniques to solve practical problems.
Outline

• Tutorial Introduction
  - Ubiquitous discreteness in natural language processing
  - Challenges of dealing with discreteness in neural NLP
• Discrete Input Space
  - Mapping discrete symbols to distributed representation
• Discrete Latent Space
  - Addressing the non-differential problem in back-propagation of discrete variables
• Discrete Output Space
  - Learning and inference in exponential hypothesis space
  - Training without maximum likelihood estimation
• Take Away
Part I: Introduction
Roadmap

• The role of distributed representation in deep learning
• Ubiquitous discreteness in natural language processing
• Challenges of dealing with discreteness in deep learning-based NLP
  - Continuous/distributed representation
  - Non-differentiability
  - Exponential search space
Neural Networks

Input layers take data signals

Hidden layers perform non-linear transformation

Output layers give task oriented predictions
Forward

Obtaining predictions by forward propagation
Backward Updating parameters by backward propagation
DL is suitable for continuous variables

- For speech and images, the input and output spaces are always **continuous**, which are straightforward for forward and backward propagations in neural networks.
Ubiquitous Discreteness in NLP

• Natural Language is discrete
  – input space, latent space, output space

I like this tutorial

sentence

parse tree
From Input to Output

X \Rightarrow \text{Hidden} \Rightarrow Y
From Input to Output

\[ X \xrightarrow{\text{neural networks}} Y \]
From Input to Output

\[ \text{Word} \quad \text{Sentence} \quad \text{Tree} \quad \text{Graph} \quad \ldots \]

Input and Output Spaces may be discrete in NLP!
$P(Y|X)$ with Latent Variable $Z$

$X \xrightarrow{} Z \xrightarrow{} Y$

(w/ probabilistic modeling)

\[
P(Y|X) = \sum_{Z} P(Z|X)P(Y|Z, X)
\]
$P(Y|X)$ with Latent Variable $Z$

Latent space may also be discrete in NLP!
Non-Trivial to Deal with Discreteness in Neural Networks

Input, latent and output of NLP tasks are oftentimes discrete symbols or structures.
Challenges of Discreteness

• Input Space
  – How to get good distributed representation?
Challenges of Discreteness

• Input Space
  – How to get good distributed representation?

• Latent Space
  – Difficult for Backpropagation
Challenges of Discreteness

- **Input Space**
  - How to get good distributed representation?

- **Latent Space**
  - Hard for Backpropagation

- **Output Space**
  - Exponential Search Space, hard for learning and inference.
  - Besides MLE, hard for training.
Challenges of Discreteness

- Input Space
  - How to get good distributed representation?
- Latent Space
  - Hard for Backpropagation
- Output Space
  - Exponential Search Space, hard for learning and inference.
  - Besides MLE, hard for training.

More challenges will be introduced in following parts.
What’s Next?

In following parts, we will introduce techniques to alleviate above problems for **input**, **latent** and **output** spaces, respectively.
Outline

• Tutorial Introduction
  - Ubiquitous discreteness in natural language processing
  - Challenges of dealing with discreteness in neural NLP
• Discrete Input Space
  - Mapping discrete symbols to distributed representation
• Discrete Latent Space
  - Addressing the non-differential problems in BP for discrete variables
• Discrete Output Space
  - Learning and inference in exponential hypothesis space
  - Training without maximum likelihood estimation
• Take Away
Part II: Discrete Input Space
Outline

• Tutorial Introduction
  – Ubiquitous discreteness in natural language processing
  – Challenges of dealing with discreteness in neural NLP

• Discrete Input Space
  – Mapping discrete symbols to distributed representation

• Discrete Latent Space
  – Addressing the non-differential problems in BP for discrete variables

• Discrete Output Space
  – Learning and inference in exponential hypothesis space
  – Training without maximum likelihood estimation

• Take Away
Roadmap

• Examples of discrete input space
  - Sentence, graph, tree, relation, etc.

• Embedding discrete input as distributed vectors
  - From one-hot to distributed representations
  - From context-independent to context dependent representations

• Incorporating discrete structures into neural architectures.
Images are continuous signals which can be fed into neural networks directly as distributed representations.
How about Text?

Word
Sentence
Tree
Graph
Relation
...

\[
X \xrightarrow{\text{neural networks}} Y
\]
Dealing with Discrete Input

• For example, for sentence classification task, we can input the **sentences**, **syntax tree** of the sentence or even extra **knowledge graph** into neural networks to get final predictions.

• Inputs can be **discrete** symbols/structures, which can not be fed to neural networks directly.

“I am visiting Hong Kong”
ID/One-Hot Representation

Index representation

\[ V = \begin{cases} 
0 & \text{Hong } \\
1 & \text{I } \\
2 & \text{visiting } \\
3 & \text{am } \\
4 & \text{Kong} 
\end{cases} \]

One-hot representation

\[ \begin{align*}
I & \text{ am visiting Hong Kong} \\
& \begin{pmatrix}
0 \\
1 \\
0 \\
0 \\
0
\end{pmatrix} \\
& \begin{pmatrix}
0 \\
0 \\
0 \\
1 \\
0
\end{pmatrix} \\
& \begin{pmatrix}
1 \\
0 \\
0 \\
0 \\
0
\end{pmatrix} \\
& \begin{pmatrix}
0 \\
0 \\
0 \\
0 \\
1
\end{pmatrix}
\end{align*} \]

Issue with the two representations:
closeness in values does not reflect the semantic relevance
Embeddings

- Map a word to a low-dimensional space
  - Not as low as one-dimensional ID representation
  - Not as high as $|\mathcal{V}|$-dimensional one-hot representation

- Word vector representation (a.k.a., word embeddings)
  - Mapping a word to a vector
  - Equivalent to linear transformation of one-hot vector

\[
\begin{pmatrix}
\text{Embedding of word } i \\
\text{retrieved by matrix-vector multiplication}
\end{pmatrix} \cdot \begin{pmatrix}
0 \\
1 \\
. \\
. \\
. \\
0
\end{pmatrix}
\]

One-hot representation of word $i$ (sparse)
Embedding by Table Lookup

Transforming discrete symbols to distributed representations by table lookup.
The embedding matrix will be updating during the whole neural network training.
Question

Is the learned embedding in a specific task generalizable to other tasks?
Pretraining Embedding

Pretraining word embedding in large-scale corpora, and then fine tuning in downstream tasks.
Word2Vec

Distributed semantics: similar context leads to similar semantics.

Word2Vec

distributed semantics: similar context leads to similar semantics.

Context Independent!
Context Dependent Representation

Word Embedding are related to its context of observed sentences.

One word has different embeddings

I have a dog. [SEP] He likes **play**-ing. [SEP]

Key Ideas

1. Transformer block (multi-head attention, positional embedding, layer norm)
2. Masked Language Model
3. Next sentence prediction
# Context Dependent VS. Context Independent Embeddings

<table>
<thead>
<tr>
<th></th>
<th>Word2Vec</th>
<th>BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
<td>Context Independent</td>
<td>Context Dependent</td>
</tr>
<tr>
<td><strong>Capacity</strong></td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Bad</td>
<td>Good</td>
</tr>
</tbody>
</table>
Advanced Representations

- Input
  - rich and rightful feature (context, order, etc.) [Roberta, XLNet, K-Bert]

- Model
  - feature aggregation
  - utilize more context [XLNet]

- Pre-training Objective [MASS, ERINE, SpanBert, T5, etc.]
  - Bert style, Mass style, …
Rich context

• The context of a word can be viewed as the feature for the word in concern.

• The more the valid context/feature, the better it can represent the word in concern.
Rich context

- In pre-training, we need sufficient number of masks to be computationally efficient.

- The mask in Bert is actually the introduced noise feature for representing a real word.

- There is a trade-off between Quantity and Quality.

- Recent papers Roberta, XLNet and K-Bert can be thought of as having enriched context and getting rid of noise.
Roberta

• Change wrt. Bert
  • Removed NSP task, (SpanBert did the same thing)
    • two segment from different document is noise feature to each other
  • All words in a training example came from the same document
  • Dynamic masking with large batch (up to 8K) and some lr increment and momentum change($\text{beta}_2: 0.999 \rightarrow 0.98$)
    • More training data (up to 160G, 10 times compared to data used in Bert)

XLNet

- Change wrt. Bert

- two-stream self-attention, latter words will have more words to observe.

\[ J_{\text{BERT}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{is a city}), \]

\[ J_{\text{XLNet}} = \log p(\text{New} \mid \text{is a city}) + \log p(\text{York} \mid \text{New}, \text{is a city}). \]

- Caching mechanism, extend observable sequence up to 512+384, means even more feature.

Model

- XLNet tries to utilize more context by devising a two-stream attention mechanism.
- XLNet also devised caching mechanism to utilizing even more context.
- Not much work on feature aggregation.
Pre-training objective: MASS

- Mass:

Pre-training objective: T5

- **T5**: Text-To-Text Transfer Transformer

[Original text]
Thank you for inviting me to your party last week.

[Inputs]
Thank you <X> me to your party <Y> week.

[Targets]
<X> for inviting <Y> last <Z>

Pre-training objective: SpanBert

- SpanBert:

\[
\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(x_7) + \mathcal{L}_{\text{SBO}}(x_4, x_9, p_7)
\]

Figure 1: An illustration of SpanBERT. In this example, the span *an American football game* is masked. The span boundary objective then uses the boundary tokens *was* and *to* to predict each token in the masked span.

Pre-training objective: ERNIE

Besides Sentences
• Syntax tree structures are widely used in NLP, offering informative syntax information inside the tree structure, which is helpful to downstream task performance.

• Many related works study how to encode such tree structures.
Recursive NNs like tree (instead of sequential) structure Recurrent NNs.

From leafs to root, encoding the whole tree from bottom to up.

Tree LSTM

Tree LSTM likes tree (instead of sequential) structure LSTM.

On Tree Based Neural Sentence Modeling

However, tree structured NNs have been less useful in recent days.

- This paper studies to which extend tree-based encoders help downstream tasks.
On Tree Based Neural Sentence Modeling

However, tree structured NNs have been less useful in recent days.

- This paper studies to which extend tree-based encoders help downstream tasks.
Candidates include 2 Trivial Trees without Syntax

(a) Parsing tree.  (b) Balanced tree.  (c) Gumbel tree.
(d) Left-branching tree.  (e) Right-branching tree.

Trivial Trees
Input Tree Representations into Different Tasks

(a) Encoder-decoder framework for sentence generation.

(b) Encoder-classifier framework for sentence classification.

(c) Siamese encoder-classifier framework for sentence relation classification.
# Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>AGN</th>
<th>ARP</th>
<th>ARF</th>
<th>DBpedia</th>
<th>WSR</th>
<th>NLI</th>
<th>Conj</th>
<th>Para</th>
<th>MT</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Latent Trees</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gumbel</td>
<td>91.8</td>
<td>87.1</td>
<td>48.4</td>
<td>98.6</td>
<td>66.7</td>
<td>80.4</td>
<td>51.2</td>
<td>20.4</td>
<td>17.4</td>
<td>39.5</td>
</tr>
<tr>
<td>+bi-leaf-RNN</td>
<td>91.8</td>
<td><strong>88.1</strong></td>
<td><strong>49.7</strong></td>
<td>98.7</td>
<td>69.2</td>
<td><strong>82.9</strong></td>
<td>53.7</td>
<td>20.5</td>
<td>22.3</td>
<td>75.3</td>
</tr>
<tr>
<td><strong>(Constituency) Parsing Trees</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parsing</td>
<td>91.9</td>
<td>87.5</td>
<td>49.4</td>
<td>98.8</td>
<td>66.6</td>
<td>81.3</td>
<td>52.4</td>
<td>19.9</td>
<td>19.1</td>
<td>44.3</td>
</tr>
<tr>
<td>+bi-leaf-RNN</td>
<td>92.0</td>
<td>88.0</td>
<td>49.6</td>
<td>98.8</td>
<td>68.6</td>
<td>82.8</td>
<td>53.4</td>
<td>20.4</td>
<td>22.2</td>
<td>72.9</td>
</tr>
<tr>
<td><strong>Trivial Trees</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Balanced</td>
<td>92.0</td>
<td>87.7</td>
<td>49.1</td>
<td>98.7</td>
<td>66.2</td>
<td>81.1</td>
<td>52.1</td>
<td>19.7</td>
<td>19.0</td>
<td>49.4</td>
</tr>
<tr>
<td>+bi-leaf-RNN</td>
<td><strong>92.1</strong></td>
<td>87.8</td>
<td><strong>49.7</strong></td>
<td><strong>98.8</strong></td>
<td><strong>69.6</strong></td>
<td>82.6</td>
<td><strong>54.0</strong></td>
<td>20.5</td>
<td>22.3</td>
<td>76.0</td>
</tr>
<tr>
<td>Left-branching</td>
<td>91.9</td>
<td>87.6</td>
<td>48.5</td>
<td>98.7</td>
<td>67.8</td>
<td>81.3</td>
<td>50.9</td>
<td>19.9</td>
<td>19.2</td>
<td>48.0</td>
</tr>
<tr>
<td>+bi-leaf-RNN</td>
<td>91.2</td>
<td>87.6</td>
<td>48.9</td>
<td>98.6</td>
<td>67.7</td>
<td>82.8</td>
<td>53.3</td>
<td>20.6</td>
<td>21.6</td>
<td>72.9</td>
</tr>
<tr>
<td>Right-branching</td>
<td>91.9</td>
<td>87.7</td>
<td>49.0</td>
<td><strong>98.8</strong></td>
<td>68.6</td>
<td>81.0</td>
<td>51.3</td>
<td>20.4</td>
<td>19.7</td>
<td>54.7</td>
</tr>
<tr>
<td>+bi-leaf-RNN</td>
<td>91.9</td>
<td>87.9</td>
<td>49.4</td>
<td>98.7</td>
<td>68.7</td>
<td>82.8</td>
<td>53.5</td>
<td><strong>20.9</strong></td>
<td><strong>23.1</strong></td>
<td><strong>80.4</strong></td>
</tr>
<tr>
<td><strong>Linear Structures</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM</td>
<td>91.7</td>
<td>87.8</td>
<td>48.8</td>
<td>98.6</td>
<td>66.1</td>
<td>82.6</td>
<td>52.8</td>
<td>20.3</td>
<td>19.1</td>
<td>46.9</td>
</tr>
<tr>
<td>+bidirectional</td>
<td>91.7</td>
<td>87.8</td>
<td>49.2</td>
<td>98.7</td>
<td>67.4</td>
<td>82.8</td>
<td>53.3</td>
<td>20.2</td>
<td>21.3</td>
<td>67.0</td>
</tr>
<tr>
<td><strong>Avg. Length</strong></td>
<td>31.5</td>
<td>33.7</td>
<td>33.8</td>
<td>20.1</td>
<td>23.1</td>
<td>11.2</td>
<td>23.3</td>
<td>10.2</td>
<td>34.1</td>
<td>34.1</td>
</tr>
</tbody>
</table>
Experimental Results

All experiments are conducted 5 times to get average results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Sentence Classification</th>
<th></th>
<th></th>
<th></th>
<th>Sentence Relation</th>
<th></th>
<th></th>
<th>Sentence Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AGN</td>
<td>ARP</td>
<td>ARF</td>
<td>DBpedia</td>
<td>WSR</td>
<td>NLI</td>
<td>Conj</td>
<td>Para</td>
</tr>
<tr>
<td>Latent Trees</td>
<td>Gumbel</td>
<td>91.8</td>
<td>87.1</td>
<td>48.4</td>
<td>98.6</td>
<td>66.7</td>
<td>80.4</td>
<td>51.2</td>
</tr>
<tr>
<td></td>
<td>+bi-leaf-RNN</td>
<td>91.8</td>
<td>88.1</td>
<td>49.7</td>
<td>98.7</td>
<td>69.2</td>
<td><strong>82.9</strong></td>
<td>53.7</td>
</tr>
<tr>
<td>(Constituency) Parsing Trees</td>
<td>Parsing</td>
<td>91.9</td>
<td>87.5</td>
<td>49.4</td>
<td><strong>98.8</strong></td>
<td>66.6</td>
<td>81.3</td>
<td>52.4</td>
</tr>
<tr>
<td></td>
<td>+bi-leaf-RNN</td>
<td>92.0</td>
<td>88.0</td>
<td>49.6</td>
<td><strong>98.8</strong></td>
<td>68.6</td>
<td>82.8</td>
<td>53.4</td>
</tr>
<tr>
<td>Trivial Trees</td>
<td>Balanced</td>
<td>92.0</td>
<td>87.7</td>
<td>49.1</td>
<td>98.7</td>
<td>66.2</td>
<td>81.1</td>
<td>52.1</td>
</tr>
<tr>
<td></td>
<td>+bi-leaf-RNN</td>
<td><strong>92.1</strong></td>
<td>87.8</td>
<td>49.7</td>
<td><strong>98.8</strong></td>
<td><strong>69.6</strong></td>
<td>82.6</td>
<td><strong>54.0</strong></td>
</tr>
<tr>
<td></td>
<td>Left-branching</td>
<td>91.9</td>
<td>87.6</td>
<td>48.5</td>
<td>98.7</td>
<td>67.8</td>
<td>81.3</td>
<td>50.9</td>
</tr>
<tr>
<td></td>
<td>+bi-leaf-RNN</td>
<td>91.2</td>
<td>87.6</td>
<td>48.9</td>
<td>98.6</td>
<td>67.7</td>
<td>82.8</td>
<td>53.3</td>
</tr>
<tr>
<td></td>
<td>Right-branching</td>
<td>91.9</td>
<td>87.7</td>
<td>49.0</td>
<td><strong>98.8</strong></td>
<td>68.6</td>
<td>81.0</td>
<td>51.3</td>
</tr>
<tr>
<td></td>
<td>+bi-leaf-RNN</td>
<td>91.9</td>
<td>87.9</td>
<td>49.4</td>
<td>98.7</td>
<td>68.7</td>
<td>82.8</td>
<td>53.5</td>
</tr>
<tr>
<td>Linear Structures</td>
<td>LSTM</td>
<td>91.7</td>
<td>87.8</td>
<td>48.8</td>
<td>98.6</td>
<td>66.1</td>
<td>82.6</td>
<td>52.8</td>
</tr>
<tr>
<td></td>
<td>+bidirectional</td>
<td>91.7</td>
<td>87.8</td>
<td>49.2</td>
<td>98.7</td>
<td>67.4</td>
<td>82.8</td>
<td>53.3</td>
</tr>
<tr>
<td>Avg. Length</td>
<td>31.5</td>
<td>33.7</td>
<td>33.8</td>
<td>20.1</td>
<td>23.1</td>
<td>11.2</td>
<td>23.3</td>
<td>10.2</td>
</tr>
</tbody>
</table>
Left-branching trees pay more attention to left words, but balanced trees treat all words fairly, and learns the weights by model.
Shallow Trees work Better

Constructing balanced trees with varying depth.

Shallow trees leads to better performances.

Figure 5: \( \rho \)-depth and \( \rho \)-performance lines for three tasks. There is a trend that the depth drops and the performance raises with the growth of \( \rho \).
Graph convolution neural networks can encode the graph structures as distributed representations.

**Spectral convolution**  
[Kipf & Welling, 2016]

**Spatial convolution**  
[Duvenaud et al., 2015]

**Other graph operations**  
E.g., attention  
[Veličković et al., 2018]
Summary for Discrete Input Space

- Representing discrete tokens
  - Pretrained word embeddings by table lookup
  - Pretrained word embeddings within context
- Representing discrete structures
  - Trees, graphs, etc.
  - Structured CNN, RNN, attention, etc.
References

• Welch G, Bishop G. An introduction to the Kalman filter.