Neural Encoding with Structured Decoding

Pushpendre Rastogi
3rd year CS Phd. Student
pushpendre@jhu.edu

Johns Hopkins University

CLSP Student Seminar, Spring 2016
1. Introduction

2. Best of Both Worlds: Neural Encoding with Structured Decoding

3. Acknowledgements and References
Introduction: Two Themes

1. Improving Neural Network Architectures.
Outline

1. Introduction

2. Best of Both Worlds: Neural Encoding with Structured Decoding

3. Acknowledgements and References
String transduction: Convert an input string to an output string.

Example

- Morphological Transduction:
  - Convert an imperative word in german to its past participle form. a b r e i b t $\mapsto$ a b g e r i e b e n
- Lemmatization:
  - Lemmatize a word in tagalog. b i n a w a l a n $\mapsto$ b a w a l
- Annotate a string:
  - Bob is a builder $\mapsto$ Noun Verb Det Noun
What do we offer?

**Task = 13SIA**
- BiLSTM: 80
- WFST: 85
- Seq2Seq: 90
- Attention: 95

**Task = 2PIE**
- BiLSTM: 100
- WFST: 80
- Seq2Seq: 95
- Attention: 85

**Task = 2PKE**
- BiLSTM: 95
- Seq2Seq: 80

**Task = rP**
- BiLSTM: 90
- Seq2Seq: 85
The Idea

Use a Neural Sequence Encoder to weight the arcs of a Weighted FST.
Weighted Finite State Transducers: Deterministic

What is a State?
The States of an FST/WFST are its Memory.

Previous Work weights this transducer.

Pushpendre Rastogi (CLSP, JHU)
Weighted Finite State Transducers: Deterministic

0 \rightarrow 1 \rightarrow 2 \rightarrow 3

s:s \rightarrow a:a \rightarrow y:y

What is a State?
The States of an FST/WFST are its Memory.

Previous Work weights this transducer.

Pushpendre Rastogi (CLSP, JHU)
Weighted Finite State Transducers: Deterministic

What is a State?
The States of an FST/WFST are its Memory.

Previous Work weights this transducer.

Pushpendre Rastogi (CLSP, JHU)
Background

Weighted Finite State Transducers: Deterministic

What is a State?
The States of an FST/WFST are its Memory.

Previous Work weights this transducer.
Weighted Finite State Transducers: Non-Deterministic

What's in a Path?

A Path is an alignment.

\[
\begin{align*}
\epsilon:s & \mapsto \rightarrow say:sassy \\
\epsilon&s & \mapsto \rightarrow say:say \\
\epsilon:s & \mapsto \rightarrow say:sasy \\
\end{align*}
\]

Previous Work weights this transducer.

Pushpendre Rastogi (CLSP, JHU)
Weighted Finite State Transducers: Non-Deterministic

What's in a Path?
A Path is an alignment.

$(\epsilon:s \ s:a \ a:s \ y:s) \mapsto \rightarrow \text{say:sass}$

$(\epsilon:s \ s:a \ a:\epsilon \ y:y) \mapsto \rightarrow \text{say:say}$

$(\epsilon: \epsilon \ s:s \ a:a \ y:y) \mapsto \rightarrow \text{say:say}$

$(\epsilon:s \ s:a \ a:s \ y:y) \mapsto \rightarrow \text{say:sasy}$

Previous Work weights this transducer.
Weighted Finite State Transducers: Non-Deterministic

What's in a Path?
A Path is an alignment.

Previous Work weights this transducer.

Pushpendre Rastogi (CLSP, JHU)
Weighted Finite State Transducers: Non-Deterministic

What’s in a Path?

A Path is an alignment.

- \((\varepsilon:s \ s:a \ a:s \ y:s) \mapsto \text{say: sass}\)
- \((\varepsilon:s \ s:a \ a:\varepsilon \ y:y) \mapsto \text{say: say}\)
- \((\varepsilon:\varepsilon \ s:s \ a:a \ y:y) \mapsto \text{say: say}\)
- \((\varepsilon:s \ s:a \ a:s \ y:y) \mapsto \text{say: sasy}\)

Previous Work weights this transducer.
Background

Neural Bi-Directional Sequence Encoder

\[ \alpha_0 \rightarrow e_s \rightarrow e_a \rightarrow e_y \]

\( e_s \rightarrow \) \( e_a \rightarrow \) \( e_y \rightarrow \)
Background

Neural Bi-Directional Sequence Encoder

\[ f(\alpha_0, e_s) \]

\[ \alpha_0 \]

\[ e_s \]

\[ e_a \]

\[ e_y \]
Neural Bi-Directional Sequence Encoder

\[ \alpha_0 \rightarrow \alpha_1 \rightarrow e_y \]

\[ e_s \rightarrow e_a \]

Pushpendre Rastogi (CLSP, JHU)
Neural Bi-Directional Sequence Encoder

\[
\begin{align*}
\alpha_0 & \rightarrow \alpha_1 \rightarrow \alpha_2 \\
e_s & \rightarrow e_a \rightarrow e_y \\
\end{align*}
\]
Neural Bi-Directional Sequence Encoder

\[ \alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \beta_3 \]

\[ e_s \rightarrow e_a \rightarrow e_y \]
Neural Bi-Directional Sequence Encoder

\[ \alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \text{Output} \]

- \( e_s \) to \( \alpha_0 \)
- \( e_a \) to \( \alpha_1 \)
- \( e_y \) to \( \alpha_2 \)

\[ \beta_2 \rightarrow \beta_3 \]
Background

Neural Bi-Directional Sequence Encoder

\[ \alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \text{Output} \]

\[ e_s \rightarrow \beta_1 \rightarrow \beta_2 \rightarrow \beta_3 \]

\[ e_a \rightarrow \]
### Weighted Finite State Transducers [Moh97, Eis02]

**Pros**

- The states in an FST can be tailored for the task.
- Can compute the probability of a string.

**Cons**

- Traditionally arcs weights are linear functionals of arc features.
- ROI on feature engineering may be low.
- The model may become slow if there are too many features.
- The local features may not be expressive enough.

### Neural Encoders and Decoders [SVL14]

**Pros**

- Produce reasonable results with zero feature engineering.

**Cons**

- Require a lot of training data for performance.
- Cannot return the probability of a string.
Background: Existing models.

### Weighted Finite State Transducers [Moh97, Eis02]

**Pros**
- The states in an FST can be tailored for the task.
- Can compute the probability of a string.

**Cons**

### Neural Encoders and Decoders [SVL14]

**Pros**

**Cons**
Background: Existing models.

**Weighted Finite State Transducers [Moh97, Eis02]**

**Pros** The states in an FST can be tailored for the task. Can compute the probability of a string.

**Cons** Traditionally arcs weights are linear functionals of arc features.
- ROI on feature engineering may be low.
- The model may become slow if there are too many features.
- The local features may not be expressive enough.

**Neural Encoders and Decoders [SVL14]**

**Pros**

**Cons**
Background: Existing models.

Weighted Finite State Transducers [Moh97, Eis02]

Pros  The states in an FST can be tailored for the task. Can compute the probability of a string.

Cons  Traditionally arcs weights are linear functionals of arc features.
       - ROI on feature engineering may be low.
       - The model may become slow if there are too many features.
       - The local features may not be expressive enough.

Neural Encoders and Decoders [SVL14]

Pros  Produce reasonable results with zero feature engineering.

Cons
### Background: Existing models.

#### Weighted Finite State Transducers [Moh97, Eis02]

**Pros** The states in an FST can be tailored for the task. Can compute the probability of a string.

**Cons** Traditionally arcs weights are linear functionals of arc features.
- ROI on feature engineering may be low.
- The model may become slow if there are too many features.
- The local features may not be expressive enough.

#### Neural Encoders and Decoders [SVL14]

**Pros** Produce reasonable results with zero feature engineering.

**Cons** Require a lot of training data for performance. Cannot return the probability of a string.
Neural Encoding with Structured Decoding

Figure: The automaton \( I \) encoding say.

Figure: Transducer \( F \). Only a few of the possible states and edit arcs are shown.

Previous Work weights these transducers
Neural Encoding with Structured Decoding

Figure: The automaton $I$ encoding say.

Figure: Transducer $F$. Only a few of the possible states and edit arcs are shown. Previous Work weights these transducers.

Figure: $G = I \circ F$. Only a few states, but all arcs between them are shown. Our Work weights this transducer.
Neural Encoding with Structured Decoding

Figure: The automaton $I$ encoding say.

Figure: Transducer $F$. Only a few of the possible states and edit arcs are shown. Previous Work weights these transducers.

Figure: $G = I \circ F$. Only a few states, but all arcs between them are shown. Our Work weights this transducer.

Why do we do this?

<table>
<thead>
<tr>
<th>Weighting $F$</th>
<th>Weighting edits per type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighting $G$</td>
<td>Weighting edits per token.</td>
</tr>
<tr>
<td></td>
<td>Neural features encode entire sentence.</td>
</tr>
<tr>
<td></td>
<td>We get a context dependent output side language model.</td>
</tr>
</tbody>
</table>
Neural Encoding with Structured Decoding

Figure: $G = I \circ F$. Only a few states, but all arcs between them are shown. Our Work weights this transducer.
Neural Encoding with Structured Decoding

Idea: Use a BiLSTM to weight the arcs of $G$.

Figure: $G = I \circ F$. Only a few states, but all arcs between them are shown. Our Work weights this transducer.
Let $w((1, a) \mapsto (2, s), a, s) \triangleq \langle v_{a,s}, (\alpha_2, \beta_1, e_a) \rangle$

$v_{a,s}$ represents $(a, s)$

$h$ may be the Identity or Relu, ...

**Idea:** Use a BiLSTM to weight the arcs of $G$.
Neural Encoding with Structured Decoding

Let \( w((1, a) \rightarrow (2, s), a, s) \) 
\[ \Delta \langle v_{a,s}, (\alpha_2, \beta_1, e_a) \rangle \]

\( v_{a,s} \) represents \((a, s)\)

\( h \) may be the Identity or Relu, ...

Idea: Use a stack of BiLSTM to weight the arcs of \( G \).

Figure: \( G = I \circ F \). Only a few states, but all arcs between them are shown. Our Work weights this transducer.
Neural Encoding with Structured Decoding

Let \( w((1, a) \mapsto (2, s), a, s) \)\n
\[ \triangleq \langle v_{a,s}, (\alpha_2, \beta_1, e_a) \rangle \]

\( v_{a,s} \) represents \((a, s)\)

\( h \) may be the Identity or Relu, ...

Idea: Use a stack of BiLSTM to weight the arcs of \( G \).

Training: SGD of the negative penalized conditional log-likelihood.

Figure: \( G = I \circ F \). Only a few states, but all arcs between them are shown.
Our Work weights this transducer.
We conducted experiments on two datasets:

- Morphological Reinflection of German Verbs.
- Lemmatization
We conducted experiments on two datasets:

- **Morphological Reinflection of German Verbs.**

<table>
<thead>
<tr>
<th>Task</th>
<th>Input</th>
<th>Output</th>
<th>Training Size</th>
<th>Dev Size</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>13SIA $\mapsto$ 13SKE</td>
<td>abrieb</td>
<td>abreibe</td>
<td>500</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>2PIE $\mapsto$ 13PKE</td>
<td>abreibt</td>
<td>abreiben</td>
<td>500</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>2PKE $\mapsto$ z</td>
<td>abreiben</td>
<td>abzurieben</td>
<td>500</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>rP $\mapsto$ pA</td>
<td>abreibt</td>
<td>abgerieben</td>
<td>500</td>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

- **Lemmatization**

<table>
<thead>
<tr>
<th>Task</th>
<th>Input</th>
<th>Output</th>
<th>Training Size</th>
<th>Dev Size</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basque</td>
<td>abestean</td>
<td>abestu</td>
<td>4674</td>
<td>584</td>
<td>584</td>
</tr>
<tr>
<td>English</td>
<td>activated</td>
<td>activate</td>
<td>3932</td>
<td>492</td>
<td>492</td>
</tr>
<tr>
<td>Irish</td>
<td>beathach</td>
<td>beathaigh</td>
<td>1101</td>
<td>138</td>
<td>138</td>
</tr>
<tr>
<td>Tagalog</td>
<td>binawalan</td>
<td>bawal</td>
<td>7636</td>
<td>954</td>
<td>954</td>
</tr>
</tbody>
</table>
Experiments

We conducted experiments on two datasets:

- Morphological Reinflection of German Verbs.
- Lemmatization

<table>
<thead>
<tr>
<th>Model</th>
<th>13SIA</th>
<th>2PIE</th>
<th>2PKE</th>
<th>rP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moses15</td>
<td>85.3</td>
<td>94.0</td>
<td>82.8</td>
<td>70.8</td>
</tr>
<tr>
<td>Dreyer (Backoff)</td>
<td>82.8</td>
<td>88.7</td>
<td>74.7</td>
<td>69.9</td>
</tr>
<tr>
<td>Dreyer (Lat-Class)</td>
<td>84.8</td>
<td>93.6</td>
<td>75.7</td>
<td>81.8</td>
</tr>
<tr>
<td>Dreyer (Lat-Region)</td>
<td>87.5</td>
<td>93.4</td>
<td>88.0</td>
<td>83.7</td>
</tr>
<tr>
<td>BiLSTM-WFST</td>
<td>85.1</td>
<td>94.4</td>
<td>85.5</td>
<td>83.0</td>
</tr>
<tr>
<td>Model Ensemble</td>
<td>85.8</td>
<td>94.6</td>
<td>86.0</td>
<td>83.8</td>
</tr>
</tbody>
</table>

Table: Exact match accuracy on Morphological Reinflection.

<table>
<thead>
<tr>
<th>Model</th>
<th>Basque</th>
<th>English</th>
<th>Irish</th>
<th>Tagalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base (W)</td>
<td>85.3</td>
<td>91.0</td>
<td>43.3</td>
<td>0.3</td>
</tr>
<tr>
<td>WFAffix (W)</td>
<td>80.1</td>
<td>93.1</td>
<td>70.8</td>
<td>81.7</td>
</tr>
<tr>
<td>ngrams (D)</td>
<td>91.0</td>
<td>92.4</td>
<td>96.8</td>
<td>80.5</td>
</tr>
<tr>
<td>ngrams + x (D)</td>
<td>91.1</td>
<td>93.4</td>
<td>97.0</td>
<td>83.0</td>
</tr>
<tr>
<td>ngrams + x + l (D)</td>
<td>93.6</td>
<td>96.9</td>
<td>97.9</td>
<td>88.6</td>
</tr>
<tr>
<td>BiLSTM-WFST</td>
<td>91.5</td>
<td>94.5</td>
<td>97.9</td>
<td>97.4</td>
</tr>
</tbody>
</table>

Table: Exact match accuracy on Lemmatization.
Experiments: The Learning Curve

Figure: Best match accuracy on test data Vs. Number of training samples.
Experiments: Comparison with Seq-to-Seq

Comparison between Sequence-to-sequence based models and the proposed model, on the validation set of morphological re-inflection tasks.
Experiments: Comparison with Seq-to-Seq

Comparison between Sequence-to-sequence based models and the proposed model, on the validation set of morphological re-inflection tasks.

Task = 13SIA

Task = 2PIE

Task = 2PKE

Task = rP

Accuracy

BiLSTM
WFST
Seq2Seq
Attention
1. Introduction

2. Best of Both Worlds: Neural Encoding with Structured Decoding

3. Acknowledgements and References
I collaborated with Ryan Cotterell and Jason Eisner for the work on neural-transducer hybrids. It is the culmination of a lot of earlier unpublished work done with Mo Yu, Dingquan Wang, Nanyun Peng and Elan Hourticolon-Retzler.

During this project I was sponsored by DARPA under the DEFT Program (Agreement FA8750-13-2-0017).
References

Jason Eisner.
Parameter estimation for probabilistic finite-state transducers.

Mehryar Mohri.
Finite-state transducers in language and speech processing.

Ilya Sutskever, Oriol Vinyals, and Quoc Le.
Sequence to sequence learning with neural networks.