# Neural Transition-based String Transduction for Limited-Resource Setting in Morphology

Peter Makarov and *Simon Clematide*

Institute of Computational Linguistics
University of Zurich, Switzerland

August 21, 2018

# String Transduction Tasks in Morphology

## String transduction with (optional) features

input string $\mathbf{x}$ + features $\mathbf{f} \rightarrow$ output string $\mathbf{y}$

# String Transduction Tasks in Morphology

## String transduction with (optional) features

input string **x** + features **f** → output string **y**

| **Task:** | **Input x** | + **Features f** | → **Output y** |
|---|---|---|---|
| Lemmatization | seeing | {} | see |
| Inflection | see | {Verb, Past} | seen |
| Reinflection | seen | {Verb/In, Past/In, Verb/Out, PartPres/Out} | seeing |
| Reinflection (fixed task) | geht *(go!)* | {} | gegangen *(went)* |

# String Transduction Tasks in Morphology

**String transduction with (optional) features**

input string **x** + features **f** → output string **y**

| **Task:** | **Input x** | + **Features f** | → **Output y** |
|---|---|---|---|
| Lemmatization | seeing | {} | see |
| Inflection | see | {Verb, Past} | seen |
| Reinflection | seen | {Verb/In, Past/In, Verb/Out, PartPres/Out} | seeing |
| Reinflection (fixed task) | geht *(go!)* | {} | gegangen *(went)* |

or normalization of historical texts, grapheme-to-phoneme, . . .

# General Modeling Objectives of Our Approach

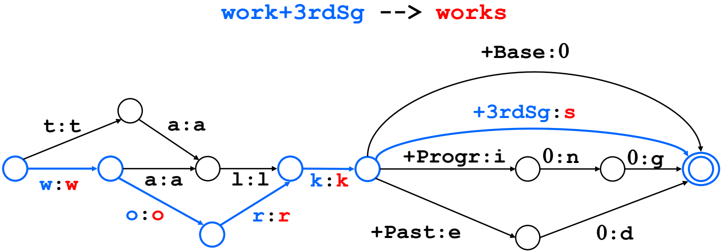
- Maximize performance for limited-resource training sets!
- Maximize language independence!
- Minimize feature engineering!
- Minimize hyperparameter tuning and training setup!

# General Modeling Objectives of Our Approach

- Maximize performance for limited-resource training sets!
- Maximize language independence!
- Minimize feature engineering!
- Minimize hyperparameter tuning and training setup!

In short, keep it as simple as possible...

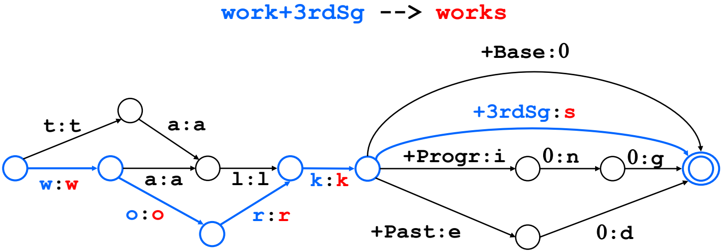
# Classical Finite-State Transducers (FST)




Source: Lauri Karttunen 2005

- Transduction of string/features **x**/**f** to **y** by edit operations on character level

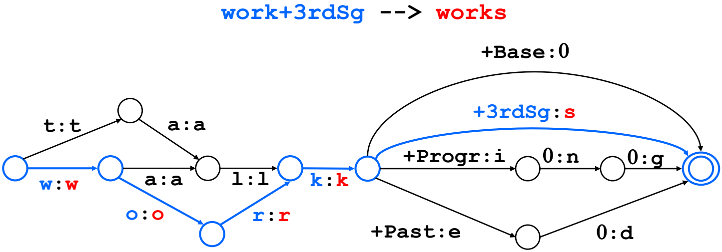
# Classical Finite-State Transducers (FST)




Source: Lauri Karttunen 2005

- Transduction of string/features **x**/**f** to **y** by edit operations on character level
- Transitions = edit operations: COPY (t:t), SUBSTITUTE (+3rdSg:s), INSERT (0:d), DELETE (+Base:0) Note: $0 = \epsilon$

# Classical Finite-State Transducers (FST)

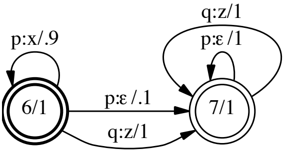



Source: Lauri Karttunen 2005

- Transduction of string/features **x**/**f** to **y** by edit operations on character level
- Transitions = edit operations: COPY (t:t), SUBSTITUTE (+3rdSg:s), INSERT (0:d), DELETE (+Base:0) Note: $0 = \epsilon$
- Fixed transition structure (topology)
- Transitions with 0/1 weights ("Boolean" semiring)
- Rational string relations

# Weighted Finite-State Transducers (WFST) [Mohri, 1997]

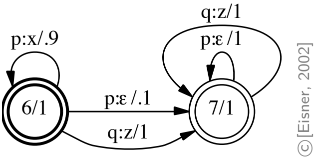
- Attach weights to transitions
- Weighted conditional probabilistic relations: $P(\mathbf{y} \mid \mathbf{x}) = \sum_{k \in Paths_{\mathbf{x}:\mathbf{y}}} P(k)$




©[Eisner, 2002]

# Weighted Finite-State Transducers (WFST) [Mohri, 1997]

- Attach weights to transitions
- Weighted conditional probabilistic relations: $P(\mathbf{y} \mid \mathbf{x}) = \sum_{k \in Paths_{\mathbf{x}:\mathbf{y}}} P(k)$

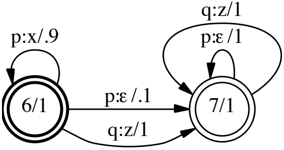



## Weight Learning Problem [Eisner, 2002]

Learning weights given a fixed topology: log-linear parametrization

# Weighted Finite-State Transducers (WFST) [Mohri, 1997]

- Attach weights to transitions
- Weighted conditional probabilistic relations: $P(\mathbf{y} \mid \mathbf{x}) = \sum_{k \in Paths_{\mathbf{x}:\mathbf{y}}} P(k)$




©[Eisner, 2002]

## Weight Learning Problem [Eisner, 2002]

Learning weights given a fixed topology: log-linear parametrization

## Topology Learning Problem

There are many possibilities for character-level string alignment . . .

```
f l i e g e n    f l i e g e n    f l i e g e n    f l i e g e n
| | | | | | |    | | | | | | |    | | | | | | |    | | | | | | |
f l   o g        f l o   g        f l o g          f l       o g
```

# LAT: "Latent-Variable Modeling of String Transductions with Finite-State Methods" [Dreyer et al., 2008]

- WFST-based approach
- Considers (almost) all possible character alignments

# LAT: "Latent-Variable Modeling of String Transductions with Finite-State Methods" [Dreyer et al., 2008]

- WFST-based approach
- Considers (almost) all possible character alignments
- Adds (engineered!) latent contextual features that learn lexical classes, modification patterns, etc.
- Uses a complex staged training regime

# LAT: "Latent-Variable Modeling of String Transductions with Finite-State Methods" [Dreyer et al., 2008]

- WFST-based approach
- Considers (almost) all possible character alignments
- Adds (engineered!) latent contextual features that learn lexical classes, modification patterns, etc.
- Uses a complex staged training regime
- Works great on low-resource settings

# NWST: Weighting Finite-State Transductions with Neural Context [Rastogi et al., 2016]

- Hybrid approach: Classical FST topology + Recurrent Neural Networks for context-sensitive scoring of the transitions

# NWST: Weighting Finite-State Transductions with Neural Context [Rastogi et al., 2016]

- Hybrid approach: Classical FST topology + Recurrent Neural Networks for context-sensitive scoring of the transitions
- BiLSTM reads input $\mathbf{x}$
- Training goal maximizes conditional sequence (!) probability

$$\sum_{(x,y^*)\in dataset} \log P(\mathbf{y}^* \mid \mathbf{x})$$

(computed over all possible alignments)

# NWST: Weighting Finite-State Transductions with Neural Context [Rastogi et al., 2016]

- Hybrid approach: Classical FST topology + Recurrent Neural Networks for context-sensitive scoring of the transitions
- BiLSTM reads input **x**
- Training goal maximizes conditional sequence (!) probability

$$\sum_{(x,y^*)\in dataset} \log P(\mathbf{y}^* \mid \mathbf{x})$$

  (computed over all possible alignments)
- Topology is defined by classical edit actions:
  1:1 (substitute/copy), 1:0 (delete), 0:1 (insert)

# NWST: Weighting Finite-State Transductions with Neural Context [Rastogi et al., 2016]

- Hybrid approach: Classical FST topology + Recurrent Neural Networks for context-sensitive scoring of the transitions
- BiLSTM reads input **x**
- Training goal maximizes conditional sequence (!) probability

$$\sum_{(x,y^*)\in dataset} \log P(\mathbf{y}^* \mid \mathbf{x})$$

  (computed over all possible alignments)
- Topology is defined by classical edit actions:
  1:1 (substitute/copy), 1:0 (delete), 0:1 (insert)
- Works great on low-resource settings

# MED: Neural Machine Translation on Character-Level [Kann and Schütze, 2016]

- Standard attentional seq2seq encoder/decoder RNN model [Bahdanau et al., 2015]
- Features $\mathbf{f}$ are serialized along input $\mathbf{x}$: translates $\mathbf{x}/\mathbf{f}$ into $\mathbf{y}$

# MED: Neural Machine Translation on Character-Level [Kann and Schütze, 2016]

- Standard attentional seq2seq encoder/decoder RNN model [Bahdanau et al., 2015]
- Features **f** are serialized along input **x**: translates **x/f** into **y**
- BiLSTM encoder creates fixed-length vector representation of input
- Greedy recursive decoding outputs most probable character at each time step

# MED: Neural Machine Translation on Character-Level [Kann and Schütze, 2016]

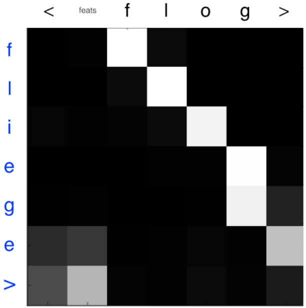
- Standard attentional seq2seq encoder/decoder RNN model [Bahdanau et al., 2015]
- Features **f** are serialized along input **x**: translates **x/f** into **y**
- BiLSTM encoder creates fixed-length vector representation of input
- Greedy recursive decoding outputs most probable character at each time step
- Works great on medium and large datasets
- Suffers on small datasets

# MED: Neural Machine Translation on Character-Level [Kann and Schütze, 2016]

- Standard attentional seq2seq encoder/decoder RNN model [Bahdanau et al., 2015]
- Features **f** are serialized along input **x**: translates **x/f** into **y**
- BiLSTM encoder creates fixed-length vector representation of input
- Greedy recursive decoding outputs most probable character at each time step
- Works great on medium and large datasets
- Suffers on small datasets
- Attention is crucial for performance

# Soft Attention in Morphological Reinflection

## Where does soft attention concentrate?



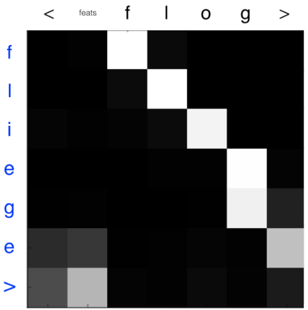

Source: [Aharoni and Goldberg, 2017]

flog → fliege
*(flew)* → *(fly)*

- Obvious pattern: monotonic diagonal character alignment

# Soft Attention in Morphological Reinflection

## Where does soft attention concentrate?



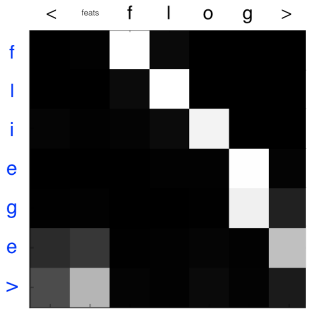

Source: [Aharoni and Goldberg, 2017]

flog → fliege
*(flew)* → *(fly)*

- Obvious pattern: monotonic diagonal character alignment
- Why not directly start with fixed monotonic character alignment?

# Soft Attention in Morphological Reinflection

## Where does soft attention concentrate?




Source: [Aharoni and Goldberg, 2017]

flog → fliege

*(flew)* → *(fly)*

- Obvious pattern: monotonic diagonal character alignment
- Why not directly start with fixed monotonic character alignment?

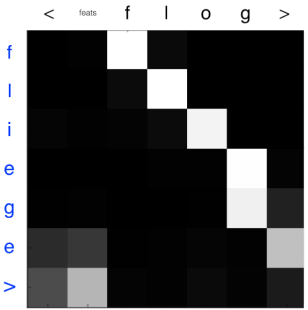
## Character alignment strategies

CRP Mans Hulden's stochastic Chinese Restaurant Process aligner

```
f l i e g e
f l o   g
```

# Soft Attention in Morphological Reinflection

## Where does soft attention concentrate?




Source: [Aharoni and Goldberg, 2017]

flog → fliege

*(flew) → (fly)*

- Obvious pattern: monotonic diagonal character alignment
- Why not directly start with fixed monotonic character alignment?

## Character alignment strategies

CRP Mans Hulden's stochastic Chinese Restaurant Process aligner

```
f l i e g e
f l o   g
```

LCS Simple deterministic Longest Common Substring + pre-/suffixation

```
f l i e g e
f l o g
```

# HA: "Morphological Inflection Generation with Hard Monotonic Attention" [Aharoni and Goldberg, 2017]

## Inflection example

```
f l o   g
| | | | | | |
f l i e g e n
```

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | $t$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ⟨s⟩ | | f | | l | | o | | | g | | | | ⟨/s⟩ | $y$ |
| ⟨s⟩ | STEP | f | STEP | l | STEP | o | STEP | STEP | g | STEP | STEP | STEP | ⟨/s⟩ | $a_t$ |
| ⟨s⟩ | ⟨s⟩ | f | f | l | l | i | i | e | g | g | e | n | ⟨/s⟩ | $x_i$ |
| 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | $i$ |

## Key ideas of transduction machine

- Given input **x** and **f**, predict action sequence **a** of a transition machine with read and write tapes!

# HA: "Morphological Inflection Generation with Hard Monotonic Attention" [Aharoni and Goldberg, 2017]

## Inflection example

```
f l o   g
| | | | | | |
f l i e g e n
```

| $t$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $y$ | ⟨s⟩ | | f | | l | | o | | | g | | | | ⟨/s⟩ |
| $a_t$ | ⟨s⟩ | STEP | f | STEP | l | STEP | o | STEP | STEP | g | STEP | STEP | STEP | ⟨/s⟩ |
| $x_i$ | ⟨s⟩ | ⟨s⟩ | f | f | l | l | i | i | e | g | g | e | n | ⟨/s⟩ |
| $i$ | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 |

## Key ideas of transduction machine

- Given input **x** and **f**, predict action sequence **a** of a transition machine with read and write tapes!
- Minimalistic set of actions (SUBSTITUTE = STEP, INSERT):
  - STEP: increment read tape position pointer $i$

# HA: "Morphological Inflection Generation with Hard Monotonic Attention" [Aharoni and Goldberg, 2017]

## Inflection example

```
f l o   g
| | | | | | |
f l i e g e n
```

| $t$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $y$ | ⟨s⟩ | | f | | l | | o | | | g | | | | ⟨/s⟩ |
| $a_t$ | ⟨s⟩ | STEP | f | STEP | l | STEP | o | STEP | STEP | g | STEP | STEP | STEP | ⟨/s⟩ |
| $x_i$ | ⟨s⟩ | ⟨s⟩ | f | f | l | l | i | i | e | g | g | e | n | ⟨/s⟩ |
| $i$ | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 |

## Key ideas of transduction machine

- Given input **x** and **f**, predict action sequence **a** of a transition machine with read and write tapes!
- Minimalistic set of actions (SUBSTITUTE = STEP, INSERT):
  - STEP: increment read tape position pointer *i*
  - INSERT character *c* on write tape

# HA: "Morphological Inflection Generation with Hard Monotonic Attention" [Aharoni and Goldberg, 2017]

## Inflection example

```
f l o   g
| | | | | | |
f l i e g e n
```

| $t$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $y$ | ⟨s⟩ | | f | | l | | o | | | g | | | | ⟨/s⟩ |
| $a_t$ | ⟨s⟩ | STEP | f | STEP | l | STEP | o | STEP | STEP | g | STEP | STEP | STEP | ⟨/s⟩ |
| $x_i$ | ⟨s⟩ | ⟨s⟩ | f | f | l | l | i | i | e | g | g | e | n | ⟨/s⟩ |
| $i$ | 0 | 0 | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 |

## Key ideas of transduction machine

- Given input **x** and **f**, predict action sequence **a** of a transition machine with read and write tapes!
- Minimalistic set of actions (SUBSTITUTE = STEP, INSERT):
  - STEP: increment read tape position pointer *i*
  - INSERT character *c* on write tape
- Gold oracle action sequence **a** for training set is determined by an external monotonic character aligner (CRP)

# HA: Encoding, Decoding and Hard Attention

**BiLSTM encoding of input $\mathbf{x} = x_1 \dots x_n$** $E(\cdot)$=embedding

$$\mathbf{h}_1, \dots, \mathbf{h}_n = \mathrm{BiLSTM}(E(x_1), \dots, E(x_n))$$

# HA: Encoding, Decoding and Hard Attention

**BiLSTM encoding of input $\mathbf{x} = x_1 \dots x_n$** $E(\cdot)$=embedding

$$\mathbf{h}_1, \dots, \mathbf{h}_n = \mathrm{BiLSTM}(E(x_1), \dots, E(x_n))$$

**LSTM decoding of character $i$ at timestep $t$ given features $\mathbf{f}$**

$$\mathbf{s}_t = \mathrm{LSTM}([E(a_{t-1});\ \mathbf{h}_i;\ \mathbf{f}]),$$

$E(a_{t-1})$ is the embedding of the most probable action of timestep $t-1$

# HA: Encoding, Decoding and Hard Attention

**BiLSTM encoding of input $\mathbf{x} = x_1 \dots x_n$** $E(\cdot)$=embedding

$$\mathbf{h}_1, \dots, \mathbf{h}_n = \mathrm{BiLSTM}(E(x_1), \dots, E(x_n))$$

**LSTM decoding of character $i$ at timestep $t$ given features $\mathbf{f}$**

$$\mathbf{s}_t = \mathrm{LSTM}([E(a_{t-1});\ \ \mathbf{h}_i;\ \ \mathbf{f}]),$$

$E(a_{t-1})$ is the embedding of the most probable action of timestep $t-1$

**Probability distribution for actions at timestep t**

$$P(a_t = k \mid \mathbf{a}_{<t}, \mathbf{x}, \mathbf{f}, \Theta) = \mathrm{softmax}_k(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b})$$

# HA: Encoding, Decoding and Hard Attention

**BiLSTM encoding of input $\mathbf{x} = x_1 \ldots x_n$** $E(\cdot)$=embedding

$$\mathbf{h}_1, \ldots, \mathbf{h}_n = \mathrm{BiLSTM}(E(x_1), \ldots, E(x_n))$$

**LSTM decoding of character $i$ at timestep $t$ given features $\mathbf{f}$**

$$\mathbf{s}_t = \mathrm{LSTM}([E(a_{t-1});\ \ \mathbf{h}_i;\ \ \mathbf{f}]),$$

$E(a_{t-1})$ is the embedding of the most probable action of timestep $t-1$

**Probability distribution for actions at timestep t**

$$P(a_t = k \mid \mathbf{a}_{<t}, \mathbf{x}, \mathbf{f}, \Theta) = \mathrm{softmax}_k(\mathbf{W} \cdot \mathbf{s}_t + \mathbf{b})$$

**Hard attention of decoder**

"Neuralese" for looking only at a single (BiLSTM encoded) character $\mathbf{h}_i$

# CA: Neural State Transition Model with Copy Action

## Key additions of our approach to HA

- Conceptualization of HA approach as a neural state transition machine with traditional character edit actions
- Inspired by transition-based dependency parsing method

# CA: Neural State Transition Model with Copy Action

## Key additions of our approach to HA

- Conceptualization of HA approach as a neural state transition machine with traditional character edit actions
- Inspired by transition-based dependency parsing method
- Adds explicit COPY action that shortens action sequences and captures the character-preserving bias of morphological processes

# CA: Neural State Transition Model with Copy Action

## Key additions of our approach to HA

- Conceptualization of HA approach as a neural state transition machine with traditional character edit actions
- Inspired by transition-based dependency parsing method
- Adds explicit COPY action that shortens action sequences and captures the character-preserving bias of morphological processes
- Simplification of our earlier HAEM approach [Makarov and Ruszics, 2017] (best low-resource architecture in SIGMORPHON 2017)

# CA: Neural State Transition Model with Copy Action

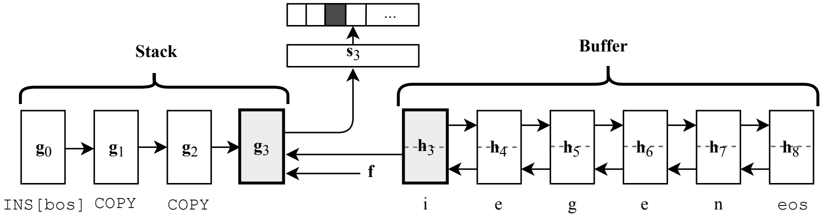
## Key additions of our approach to HA

- Conceptualization of HA approach as a neural state transition machine with traditional character edit actions
- Inspired by transition-based dependency parsing method
- Adds explicit COPY action that shortens action sequences and captures the character-preserving bias of morphological processes
- Simplification of our earlier HAEM approach [Makarov and Ruszics, 2017] (best low-resource architecture in SIGMORPHON 2017)

## Edit actions as state transition operations

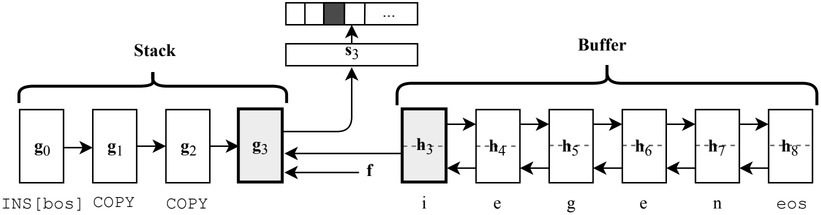
- DELETE: pop next character from input buffer
- COPY: pop next character from input buffer and append it to output
- INSERT $c$: append character $c$ to output

# Example Transduction “fliegen” → “flog”



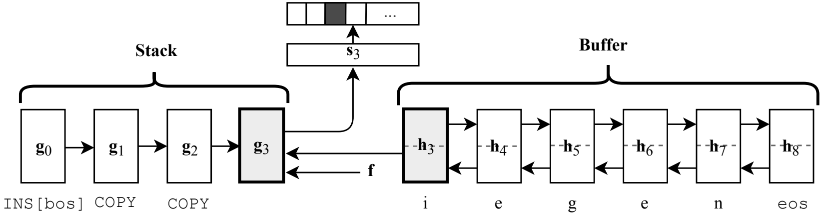

| $t$ | Transition $a_t$ | Output y | Stack | Buffer |
|---|---|---|---|---|
| 0 | INSERT(BOS) | [] | [INSERT(BOS)] | [f,l,i,e,g,e,n,EOS] |

# Example Transduction "fliegen" → "flog"



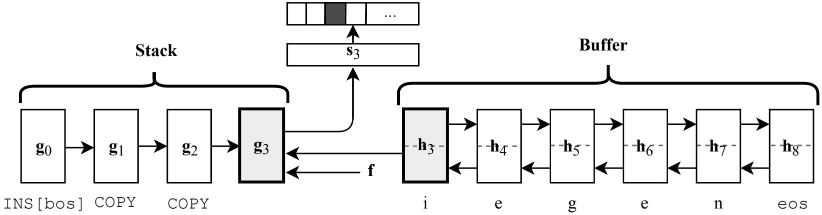

| $t$ | **Transition** $a_t$ | **Output y** | **Stack** | **Buffer** |
|---|---|---|---|---|
| 0 | INSERT(BOS) | [] | [INSERT(BOS)] | [f,l,i,e,g,e,n,EOS] |
| 1 | COPY | [f] | [COPY, INSERT(BOS)] | [l,i,e,g,e,n,EOS] |

# Example Transduction "fliegen" → "flog"




| $t$ | Transition $a_t$ | Output y | Stack | Buffer |
|---|---|---|---|---|
| 0 | INSERT(BOS) | [] | [INSERT(BOS)] | [f,l,i,e,g,e,n,EOS] |
| 1 | COPY | [f] | [COPY, INSERT(BOS)] | [l,i,e,g,e,n,EOS] |
| 2 | COPY | [f, l] | [COPY, COPY, ...] | [i,e,g,e,n, EOS] |
| 3 | DELETE | [f, l] | [DELETE,COPY, ...] | [e,g,e,n,EOS] |

# Example Transduction "fliegen" → "flog"




| $t$ | Transition $a_t$ | Output y | Stack | Buffer |
|---|---|---|---|---|
| 0 | INSERT(BOS) | [] | [INSERT(BOS)] | [f,l,i,e,g,e,n,EOS] |
| 1 | COPY | [f] | [COPY, INSERT(BOS)] | [l,i,e,g,e,n,EOS] |
| 2 | COPY | [f, l] | [COPY, COPY, . . . ] | [i,e,g,e,n, EOS] |
| 3 | DELETE | [f, l] | [DELETE,COPY, . . . ] | [e,g,e,n,EOS] |
| 4 | DELETE | [f, l] | [DELETE, DELETE, . . . ] | [g,e,n,EOS] |
| 5 | INSERT($o$) | [f, l, o] | [INSERT($o$), DELETE, . . . ] | [g,e,n,EOS] |
| 6 | COPY | [f, l, o, g] | [COPY, INSERT($o$), . . . ] | [e,n,EOS] |
| 7 | DELETE | [f, l, o, g] | [DELETE, COPY, . . . ] | [n,EOS] |
| 8 | DELETE | [f, l, o, g] | [DELETE, DELETE, . . . ] | [EOS] |
| 9 | INSERT(EOS) | [f, l, o, g] | | |

# CA and HA: Considerations

## Important architectural properties

- FST topology is replaced by probabilistic neural action predictions

# CA and HA: Considerations

## Important architectural properties

- FST topology is replaced by probabilistic neural action predictions
- Local action decisions at time $t$ have access to action history through decoder LSTM ...
- and the full input context through BiLSTM input encoding

# CA and HA: Considerations

## Important architectural properties

- FST topology is replaced by probabilistic neural action predictions
- Local action decisions at time $t$ have access to action history through decoder LSTM . . .
- and the full input context through BiLSTM input encoding
- Beam search alleviates issues from greedy decoding

# MLE Training of State-Transition System

Training data $D$ are pairs of input strings **x** and *transition sequences* **a**

(ignoring features for simplicity)

| Alignment | Input and alignment-derived actions |
|---|---|
| fliegen | (f, l, i, e, g, e, n) |
| fl og | (COPY, COPY, DEL, DEL, INS(o), COPY, DEL, DEL) |

# MLE Training of State-Transition System

Training data $D$ are pairs of input strings $\mathbf{x}$ and *transition sequences* $\mathbf{a}$

(ignoring features for simplicity)

**Alignment**

```
fliegen
fl  og
```

**Input and alignment-derived actions**

```
(f,    l,    i,    e,            g,    e,    n)
(COPY, COPY, DEL,  DEL,  INS(o), COPY, DEL,  DEL)
```

**Maximize conditional log-likelihood of training data $D$**

$$\mathcal{L}(D, \Theta) = \sum_{(\mathbf{x},\mathbf{a}) \in D} \log P(\mathbf{a} \mid \mathbf{x}; \Theta) = \sum_{(\mathbf{x},\mathbf{a}) \in D} \sum_{i=1}^{|a|} \log P(a_i \mid \mathbf{a}_{<i}, \mathbf{x}; \Theta)$$

Our parameter $\Theta$ maximizes the probability of per-character edits!

# Training Problems

## Loss-evaluation mismatch

We need character sequence accuracy, not edit action probability!

# Training Problems

## Loss-evaluation mismatch

We need character sequence accuracy, not edit action probability!

## Pipeline architecture with external aligner

- Unlike supervised parsing where it is simple to get gold actions **a**, simple algorithms do generally not produce optimal **a**.
- First, external aligner derives **a** from **x** and **y**. NN trains on **a** and **x**.

# Training Problems

## Loss-evaluation mismatch

We need character sequence accuracy, not edit action probability!

## Pipeline architecture with external aligner

- Unlike supervised parsing where it is simple to get gold actions **a**, simple algorithms do generally not produce optimal **a**.
- First, external aligner derives **a** from **x** and **y**. NN trains on **a** and **x**.
- The procedure to generate **a** is uninformed of the downstream task!

# Training Problems

## Loss-evaluation mismatch

We need character sequence accuracy, not edit action probability!

## Pipeline architecture with external aligner

- Unlike supervised parsing where it is simple to get gold actions $\mathbf{a}$, simple algorithms do generally not produce optimal $\mathbf{a}$.
- First, external aligner derives $\mathbf{a}$ from $\mathbf{x}$ and $\mathbf{y}$. NN trains on $\mathbf{a}$ and $\mathbf{x}$.
- The procedure to generate $\mathbf{a}$ is uninformed of the downstream task!

## Exposure bias

During training, system sees only gold action history to condition on.

# Training Problems

## Loss-evaluation mismatch

We need character sequence accuracy, not edit action probability!

## Pipeline architecture with external aligner

- Unlike supervised parsing where it is simple to get gold actions **a**, simple algorithms do generally not produce optimal **a**.
- First, external aligner derives **a** from **x** and **y**. NN trains on **a** and **x**.
- The procedure to generate **a** is uninformed of the downstream task!

## Exposure bias

During training, system sees only gold action history to condition on.

## Better Training

How can we train the entire system end-to-end?

# One Possible Approach: Minimum Risk Training (MRT)

**Minimize expected risk of training data $T$ consisting of (x,y) pairs**

$$\mathcal{R}(T, \Theta) = \sum_{(\mathbf{x},\mathbf{y}) \in T} \mathbb{E}_{\mathbf{a}|\mathbf{x};\Theta} \left[ \delta(\hat{y}, y) \right]$$

where the risk is a combination ($[-1, 1]$)
of normalized Levenshtein distance $\mathrm{NLD} \in [0, 1]$ and accuracy $\in \{0, 1\}$:

$$\delta(\hat{y}, y) = \mathrm{NLD}(\widehat{\mathrm{y}}, \mathrm{y}) - \mathbb{1}\{\widehat{\mathrm{y}} = \mathrm{y}\}$$

# One Possible Approach: Minimum Risk Training (MRT)

## Minimize expected risk of training data $T$ consisting of (x,y) pairs

$$\mathcal{R}(T,\Theta) = \sum_{(\mathbf{x},\mathbf{y})\in T} \mathbb{E}_{\mathbf{a}|\mathbf{x};\Theta}\left[\delta(\hat{y}, y)\right]$$

where the risk is a combination ($[-1, 1]$)
of normalized Levenshtein distance $\mathrm{NLD} \in [0, 1]$ and accuracy $\in \{0, 1\}$:

$$\delta(\hat{y}, y) = \mathrm{NLD}(\widehat{\mathrm{y}}, \mathrm{y}) - \mathbb{1}\{\widehat{\mathrm{y}} = \mathrm{y}\}$$

## Approximation of expectation under posterior $\mathbb{E}_{\mathbf{a}|\mathbf{x};\Theta}[\delta(\hat{y}, y)]$

Sampling from $P(\mathbf{a} \mid \mathbf{x}; \Theta)$ and re-normalizing allows normal gradient training [Shen et al., 2016]

# Expected Benefits from MRT and Some Issues

## Benefits

- Against loss/evaluation mismatch: Optimize for sequence accuracy

# Expected Benefits from MRT and Some Issues

## Benefits

- Against loss/evaluation mismatch: Optimize for sequence accuracy
- Against exposure bias: Sampling exposes model to predicted action histories

# Expected Benefits from MRT and Some Issues

## Benefits

- Against loss/evaluation mismatch: Optimize for sequence accuracy
- Against exposure bias: Sampling exposes model to predicted action histories
- Against MLE gold oracle action "micromanagement": Give the model freedom to prefer alternative action sequences if they lead to perfect results!

# Expected Benefits from MRT and Some Issues

## Benefits

- Against loss/evaluation mismatch: Optimize for sequence accuracy
- Against exposure bias: Sampling exposes model to predicted action histories
- Against MLE gold oracle action “micromanagement”: Give the model freedom to prefer alternative action sequences if they lead to perfect results!

## Practical issues

- Cold start training with MRT is not practical!
- Two stage training regime necessary: First MLE, second MRT!

# Expected Benefits from MRT and Some Issues

## Benefits

- Against loss/evaluation mismatch: Optimize for sequence accuracy
- Against exposure bias: Sampling exposes model to predicted action histories
- Against MLE gold oracle action "micromanagement": Give the model freedom to prefer alternative action sequences if they lead to perfect results!

## Practical issues

- Cold start training with MRT is not practical!
- Two stage training regime necessary: First MLE, second MRT!
- MRT training oscillates and cannot always improve over MLE for all datasets

# Experiments

- For all (!) dataset sizes, our approach uses character and action embeddings of size 100, single-layer LSTMs with hidden size 200
- We apply parameter tying for characters and their insert action as [Aharoni and Goldberg, 2017]
- Beam search with size 4 for our approach CA and our reimplementation HA* of HA

# Experiments

- For all (!) dataset sizes, our approach uses character and action embeddings of size 100, single-layer LSTMs with hidden size 200
- We apply parameter tying for characters and their insert action as [Aharoni and Goldberg, 2017]
- Beam search with size 4 for our approach CA and our reimplementation HA* of HA
- We use DyNet framework
- Without mini-batching except for MRT sampling (MRT samples 20 sequences)

# Experiments

- For all (!) dataset sizes, our approach uses character and action embeddings of size 100, single-layer LSTMs with hidden size 200
- We apply parameter tying for characters and their insert action as [Aharoni and Goldberg, 2017]
- Beam search with size 4 for our approach CA and our reimplementation HA* of HA
- We use DyNet framework
- Without mini-batching except for MRT sampling (MRT samples 20 sequences)
- Rough estimate of MLE training times on 1 CPU core: 30 minutes for low setting (100 training examples), 3 hours for medium settings (1,000), 30 hours for high setting (10,000)

# Experiments

- For all (!) dataset sizes, our approach uses character and action embeddings of size 100, single-layer LSTMs with hidden size 200
- We apply parameter tying for characters and their insert action as [Aharoni and Goldberg, 2017]
- Beam search with size 4 for our approach CA and our reimplementation HA* of HA
- We use DyNet framework
- Without mini-batching except for MRT sampling (MRT samples 20 sequences)
- Rough estimate of MLE training times on 1 CPU core: 30 minutes for low setting (100 training examples), 3 hours for medium settings (1,000), 30 hours for high setting (10,000)
- MRT training times: a bit longer than MLE if converging, otherwise as long as early stopping patience allows...

# Results on SIGMORPHON 2017 inflection task

| **Model** (averages) | | low | medium | **Model** (ensembles) | | low | medium |
|---|---|---|---|---|---|---|---|
| baseline | | 37.9 | 64.7 | | | 37.9 | 64.7 |
| HA* | LCS | 29.1 | 78.5 | HA*-E | LCS | 31.5 | 80.2 |
| CA | LCS | 47.3 | 79.5 | CA-E | LCS | 48.8 | 81.0 |
| HA* | CRP | 23.9 | 75.4 | HA*-E | CRP | 26.1 | 77.8 |
| CA | CRP | 42.5 | 78.9 | CA-E | CRP | 44.0 | 80.6 |
| HA*-MRT | LCS | 30.2 | 79.6 | HA*-MRT-E | LCS | 33.1 | 81.5 |
| CA-MRT | LCS | **48.1** | 80.3 | CA-MRT-E | LCS | 49.9 | 81.9 |
| HA*-MRT | CRP | 25.3 | 78.1 | HA*-MRT-E | CRP | 28.1 | 80.5 |
| CA-MRT | CRP | 43.6 | **81.1** | CA-MRT-E | CRP | 45.7 | **82.9** |
| | | | | HACM-E7 | | 46.8 | 81.8 |
| | | | | HAEM-E7 | | 48.5 | 80.3 |
| | | | | HA[EC]M-E15 | | **50.6** | 82.8 |

- 52 languages; low setting (100 training examples); medium (1000)
- Alignment strategies: Stochastic CRP and longest common substring (LCS)
- Averages and ensembles (-E) over 5 models reported

# Results on SIGMORPHON 2017 inflection task

| **Model** (averages) | | low | medium | **Model** (ensembles) | | low | medium |
|---|---|---|---|---|---|---|---|
| baseline | | 37.9 | 64.7 | | | 37.9 | 64.7 |
| HA* | LCS | 29.1 | 78.5 | HA*-E | LCS | 31.5 | 80.2 |
| CA | LCS | 47.3 | 79.5 | CA-E | LCS | 48.8 | 81.0 |
| HA* | CRP | 23.9 | 75.4 | HA*-E | CRP | 26.1 | 77.8 |
| CA | CRP | 42.5 | 78.9 | CA-E | CRP | 44.0 | 80.6 |
| HA*-MRT | LCS | 30.2 | 79.6 | HA*-MRT-E | LCS | 33.1 | 81.5 |
| CA-MRT | LCS | **48.1** | 80.3 | CA-MRT-E | LCS | 49.9 | 81.9 |
| HA*-MRT | CRP | 25.3 | 78.1 | HA*-MRT-E | CRP | 28.1 | 80.5 |
| CA-MRT | CRP | 43.6 | **81.1** | CA-MRT-E | CRP | 45.7 | **82.9** |
| | | | | HACM-E7 | | 46.8 | 81.8 |
| | | | | HAEM-E7 | | 48.5 | 80.3 |
| | | | | HA[EC]M-E15 | | **50.6** | 82.8 |

- CA with COPY mechanism heavily outperforms HA* on low setting
- CA is consistently better than HA* on medium

# Results on SIGMORPHON 2017 inflection task

| **Model** (averages) | | **low** | **medium** | **Model** (ensembles) | | **low** | **medium** |
|---|---|---|---|---|---|---|---|
| baseline | | 37.9 | 64.7 | | | 37.9 | 64.7 |
| HA* | LCS | 29.1 | 78.5 | HA*-E | LCS | 31.5 | 80.2 |
| CA | LCS | 47.3 | 79.5 | CA-E | LCS | 48.8 | 81.0 |
| HA* | CRP | 23.9 | 75.4 | HA*-E | CRP | 26.1 | 77.8 |
| CA | CRP | 42.5 | 78.9 | CA-E | CRP | 44.0 | 80.6 |
| HA*-MRT | LCS | 30.2 | 79.6 | HA*-MRT-E | LCS | 33.1 | 81.5 |
| CA-MRT | LCS | **48.1** | 80.3 | CA-MRT-E | LCS | 49.9 | 81.9 |
| HA*-MRT | CRP | 25.3 | 78.1 | HA*-MRT-E | CRP | 28.1 | 80.5 |
| CA-MRT | CRP | 43.6 | **81.1** | CA-MRT-E | CRP | 45.7 | **82.9** |
| | | | | HACM-E7 | | 46.8 | 81.8 |
| | | | | HAEM-E7 | | 48.5 | 80.3 |
| | | | | HA[EC]M-E15 | | **50.6** | 82.8 |

- MRT training improves results on low and medium

# Results on SIGMORPHON 2017 inflection task

| **Model** (averages) | | low | medium | **Model** (ensembles) | | low | medium |
|---|---|---|---|---|---|---|---|
| baseline | | 37.9 | 64.7 | | | 37.9 | 64.7 |
| HA* | LCS | 29.1 | 78.5 | HA*-E | LCS | 31.5 | 80.2 |
| CA | LCS | 47.3 | 79.5 | CA-E | LCS | 48.8 | 81.0 |
| HA* | CRP | 23.9 | 75.4 | HA*-E | CRP | 26.1 | 77.8 |
| CA | CRP | 42.5 | 78.9 | CA-E | CRP | 44.0 | 80.6 |
| HA*-MRT | LCS | 30.2 | 79.6 | HA*-MRT-E | LCS | 33.1 | 81.5 |
| CA-MRT | LCS | **48.1** | 80.3 | CA-MRT-E | LCS | 49.9 | 81.9 |
| HA*-MRT | CRP | 25.3 | 78.1 | HA*-MRT-E | CRP | 28.1 | 80.5 |
| CA-MRT | CRP | 43.6 | **81.1** | CA-MRT-E | CRP | 45.7 | **82.9** |
| | | | | HACM-E7 | | 46.8 | 81.8 |
| | | | | HAEM-E7 | | 48.5 | 80.3 |
| | | | | HA[EC]M-E15 | | **50.6** | 82.8 |

- Every system, setting and training profits from ensembling by 1-3 points
- Only our large complex winning ensemble HA[EC]M-E15 from SIGMORPHON is still a bit stronger on low

# Results on CELEX reinflection dataset

| Model | 13SIA | 2PIE | 2PKE | rP | Avg. |
|---|---|---|---|---|---|
| CELEX-BY-TASK | | | | | |
| LAT | **87.5** | 93.4 | 87.4 | 84.9 | 88.3 |
| NWFST | 85.1 | 94.4 | 85.5 | 83.0 | 87.0 |
| HA* | 84.6 | 93.9 | **88.1** | 85.1 | 87.9 |
| CA | 85.0 | 94.5 | 88.0 | 84.9 | 88.1 |
| HA*-MRT | 84.8 | 94.0 | 88.1 | 85.2 | 88.0 |
| CA-MRT | 85.6 | **94.6** | 88.0 | **85.3** | **88.4** |
| CELEX-ALL (ensembles) | | | | | |
| MED | 83.9 | 95.0 | 87.6 | 84.0 | 87.2 |
| HA | 85.8 | **95.1** | **89.5** | 87.2 | 89.5 |
| HA*-E | 85.3 | 94.8 | 88.9 | 87.4 | 89.1 |
| CA-E | 85.8 | 94.9 | 88.8 | 86.7 | 89.1 |
| HA*-MRT-E | 85.8 | 95.0 | 89.2 | **87.7** | 89.4 |
| CA-MRT-E | **86.7** | 94.9 | 89.3 | 87.1 | **89.5** |

CA-MRT on average stronger than all other strong competitors in this medium-sized task

# Results on Lemmatization Dataset [Wicentowski, 2002]

| Model | | basque | english | irish | tagalog | Avg. |
|---|---|---|---|---|---|---|
| Size | | 4.7K | 3.9K | 1.1K | 7.6K | 4.3K |
| LAT | | 93.6 | 96.9 | **97.9** | 88.6 | 94.2 |
| NWFST | | 91.5 | 94.5 | **97.9** | 97.4 | 95.3 |
| HA* | lcs | **97.0** | 97.5 | **97.9** | **98.3** | **97.7** |
| CA | lcs | 96.3 | 96.9 | 97.7 | **98.3** | 97.3 |
| HA* | crp | 96.2 | **97.7** | 97.3 | 97.9 | 97.3 |
| CA | crp | 96.1 | 96.7 | 96.8 | 97.6 | 96.8 |

HA* and CA outperform other approaches by quite some margin in this medium/high resource setting

# Connecting System Performance with Linguistic Properties

## An issue and a proposal

- Datasets sets generally lack explizit per item characterization of covered morphological phenomena, lexical properties (e.g. strong vs weak verbs)
- Automatic assessment of advantages/disadvantages of approaches wrt linguistic properties is difficult

# Connecting System Performance with Linguistic Properties

## An issue and a proposal

- Datasets sets generally lack explizit per item characterization of covered morphological phenomena, lexical properties (e.g. strong vs weak verbs)
- Automatic assessment of advantages/disadvantages of approaches wrt linguistic properties is difficult
- Challenge test sets got popular in Neural Machine Translation [Sennrich, 2017, Avramidis et al., 2018].
- Computational morphology could profit from analog initiatives

# Connecting System Performance with Linguistic Properties

## An issue and a proposal

- Datasets sets generally lack explizit per item characterization of covered morphological phenomena, lexical properties (e.g. strong vs weak verbs)
- Automatic assessment of advantages/disadvantages of approaches wrt linguistic properties is difficult
- Challenge test sets got popular in Neural Machine Translation [Sennrich, 2017, Avramidis et al., 2018].
- Computational morphology could profit from analog initiatives
- We added and released explicit information to all CELEX rP task datasets as an appetizer[a]:
  regular/irregular verbs, "ge" affixation specification

[a] https://gitlab.cl.uzh.ch/morphology-datasets/sigmorphon-2017-format/celex-by-task

# Summary

- Our neural state-transition system that predicts character-wise edit actions performs extremely strong in limited-resource settings and competitive in high settings for typical morphological tasks

# Summary

- Our neural state-transition system that predicts character-wise edit actions performs extremely strong in limited-resource settings and competitive in high settings for typical morphological tasks
- We show that in limited-resource settings, MRT training consistently improves results

# Summary

- Our neural state-transition system that predicts character-wise edit actions performs extremely strong in limited-resource settings and competitive in high settings for typical morphological tasks
- We show that in limited-resource settings, MRT training consistently improves results
- General and robust: no language-specific modeling, no feature engineering, almost no hyperparameter tuning

# The End

Thank you for your attention.

Comments? Questions?

Source code

`https://github.com/ZurichNLP/coling2018-neural-trans-based-morphology`

All test set outputs of our system (re)-implementations:

`https://github.com/ZurichNLP/ZurichNLP/`

`coling2018-neural-transition-based-morphology-test-data`


## Acknowledgments

Tatyana Ruzsics, Tanja Samardžić, Mathias Müller, Roee Aharoni, Pushpendre Rastogi, Katharina Kann, and the anonymous reviewers.

# Outlook: Imitation Learning

## Another solution for end-to-end learning

- Dynamic oracle for gold actions
- Imitation learning approach (forthcoming EMNLP 2018 paper)
- No coldstart/warmstart training!
- Best results on all 3 settings in CoNLL-SIGMORPHON 2018 inflection task!

# Bibliography


Aharoni, R. and Goldberg, Y. (2017).

Morphological inflection generation with hard monotonic attention.

In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Avramidis, E., Macketanz, V., Lommel, A., and Uszkoreit, H. (2018).

Fine-grained evaluation of quality estimation for machine translation based on a linguistically-motivated test suite.

In *Proceedings of the 1st Workshop on Translation Quality Estimation and Automatic Post-Editing*. AMTA.

Bahdanau, D., Cho, K., and Bengio, Y. (2015).

Neural machine translation by jointly learning to align and translate.

In *International Conference on Learning Representations*.

# Bibliography (cont.)

Dreyer, M., Smith, J., and Eisner, J. (2008).

Latent-variable modeling of string transductions with finite-state methods.

In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Eisner, J. (2002).

Parameter estimation for probabilistic finite-state transducers.

In *ACL*.

Kann, K. and Schütze, H. (2016).

Single-model encoder-decoder with explicit morphological representation for reinflection.

In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

# Bibliography (cont.)


Makarov, P. and Ruszics (2017).

Align and copy: Uzh at sigmorphon 2017 shared task for morphological reinflection.

Mohri, M. (1997).

Finite-state transducers in language and speech processing.

*Computational Linguistics*, 23(2):269–311.

Rastogi, P., Cotterell, R., and Eisner, J. (2016).

Weighting finite-state transductions with neural context.

In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California. Association for Computational Linguistics.

# Bibliography (cont.)

Sennrich, R. (2017).

How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs.

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016).

Minimum risk training for neural machine translation.

In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany.

Wicentowski, R. (2002).

*Modeling and learning multilingual inflectional morphology in a minimally supervised framework*.

PhD thesis, Johns Hopkins University.