Copy That! Editing Sequences by Copying Spans

Sheena Panthaplackel*  Miltiadis Allamanis  Marc Brockschmidt
The University of Texas at Austin  Microsoft Research, Cambridge, UK

* Work done during internship at Microsoft Research, Cambridge, UK
Intelligent Editors

Bug Fixing

```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
        return ListManager.getFirst(myList);
    }
    return 0;
}
```

```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 1) {
        return ListManager.min(myList);
    }
    return null;
}
```

(Tufano et al., 2019)

Code Refactoring

```java
foo(x => {return 4;}) → foo(x=>4) (Yin et al., 2019)
```

Grammatical Error Correction

Neither of the two traffic lights are working → Neither of the two traffic lights is working (Park et al., 2020)

Style Transfer

Gotta see both sides of the story. → You have to consider both sides of the story. (Rao and Tetreault, 2018)

Text Simplification

He came back home and played piano. → He came back home. He played piano. (Sulem et al., 2018)

Learning to edit sequences
Intelligent Editors

Bug Fixing

```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
        return ListManager.getFirst(myList);
    }
    return 0;
}
```

```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 1) {
        return ListManager.min(myList);
    }
    return null;
}
```

(Tufano et al., 2019)

Code Refactoring

```java
foo(x =>{return 4 ;}) → foo(x=>4) (Yin et al., 2019)
```

Grammatical Error Correction

Neither of the two traffic lights are working → Neither of the two traffic lights is working (Park et al., 2020)

Style Transfer

Gotta see both sides of the story. → You have to consider both sides of the story. (Rao and Tetreault, 2018)

Text Simplification

He came back home and played piano. → He came back home. He played piano. (Sulem et al., 2018)

Learning to edit sequences by copying spans
Seq2Seq Generation

\( \text{in: input sequence} \)
\( \text{out: output sequence } (o_0 \ldots o_m) \)

\[ p(o_k \mid \text{in, } o_0 \ldots o_{k-1}) = \text{Gen}(o_k) \]

\[ p(o_0 \ldots o_m \mid \text{in}) = \prod_{0 \leq k \leq m} p(o_k \mid \text{in, } o_0 \ldots o_{k-1}) \]

Probability for generating \( o_k \) from vocabulary

\( a b c d e \rightarrow a b f d e \)
Seq2Seq Generation

**in**: input sequence

**out**: output sequence \((o_0 \ldots o_m)\)

\[
p(o_k | \text{in}, o_0 \ldots o_{k-1}) = \text{Gen}(o_k) + \text{Copy}(o_k, \text{pos})
\]

\[
= \sum_{\alpha \in A_k} q(\alpha | \text{in}, o_0 \ldots o_{k-1})
\]

Probability for generating \(o_k\) from vocabulary OR copying \(o_j\) from \(\text{in}[\text{pos}]\)

**Pointer Networks (Vinyals et al., 2015)**

\[
p(o_0 \ldots o_m | \text{in}) = \prod_{1 \leq k \leq m} p(o_k | \text{in}, o_0 \ldots o_{k-1})
\]

Sum of all correct actions \(\alpha \in A_k\) when decoding \(k^{th}\) token

Kinds of actions \((\alpha)\):

\[
\text{Gen}(\text{token}) \quad \text{Copy}(\text{token}, \text{pos})
\]

\[
a b c d e \rightarrow a b f d e
\]
Seq2Seq Generation

\textbf{in}: input sequence \\
\textbf{out}: output sequence \((o_0 \ldots o_m)\)

\[
p(o_k | in, o_1 \ldots o_{k-1}) = \text{Gen}(o_k) + \text{Copy}(o_k, \text{pos})
\]

\[
= \sum_{\alpha \in A_k} q(\alpha | in, o_1 \ldots o_{k-1})
\]

\[
p(o_0 \ldots o_m | in) = \prod_{1 \leq k \leq m} p(o_k | in, o_0 \ldots o_{k-1})
\]

Probability for generating \(o_k\) from vocabulary
OR copying \(o_j\) from \(in[\text{pos}]\)

\text{Pointer Networks (Vinyals et al., 2015)}

Sum of all correct actions \(\alpha \in A_k\) when decoding \(k^{th}\) token

Kinds of actions \(\alpha\):
- \(\text{Gen}(\text{token})\)
- \(\text{Copy}(\text{token}, \text{pos})\)
- \(\text{Copy}(\text{span of tokens, pos range})\)

\(a\ b\ c\ d\ e \rightarrow a\ b\ f\ d\ e\)

- \(\text{Gen}(a)\)
- \(\text{Copy}(a, 0:1)\)
- \(\text{Gen}(b)\)
- \(\text{Copy}(b, 1:2)\)
- \(\text{Gen}(f)\)
- \(\text{Gen}(d)\)
- \(\text{Copy}(d, 3:4)\)
- \(\text{Gen}(e)\)
- \(\text{Copy}(e, 4:5)\)
- \(\text{Gen}(\text{EOS})\)
Seq2Seq Generation

**in**: input sequence

**out**: output sequence \((o_0 \ldots o_m)\)

\[
p(o_k | \text{in}, o_1 \ldots o_{k-1}) = \text{Gen}(o_k) + \text{Copy}(o_k, \text{pos})
\]

\[
= \sum_{\alpha \in A_k} q(\alpha | \text{in}, o_1 \ldots o_{k-1})
\]

Probability for generating \(o_k\) from vocabulary

OR copying \(o_j\) from \(\text{in}[\text{pos}]\)

Pointer Networks (Vinyals et al., 2015)

Sum of all correct actions \(\alpha \in A_k\) when decoding \(k^{th}\) token

Kinds of actions (\(\alpha\)):

- \(\text{Gen(token)}\)
- \(\text{Copy(token, pos)}\)
- \(\text{Copy(span of tokens, pos range)}\)

\[
p(o_0 \ldots o_m | \text{in}) = \prod_{1 \leq k \leq m} p(o_k | \text{in}, o_0 \ldots o_{k-1})
\]

\(a \ b \ c \ d \ e \rightarrow a \ b \ f \ d \ e\)
Seq2Seq + Span Copying

\[ \alpha_k \rightarrow o_k \]

Action likelihoods

Scores for generating each \( v \in V \)

Encoder

Decoder

Projection Layer

Softmax Layer
Seq2Seq + Span Copying

Scores for copying each span $\text{in}[i:j]$; $0 \leq i < n, i < j \leq n$

Scores for generating each $v \in V$

Action likelihoods

$$s_{k,[i:j]} = (W \cdot (r_i \| r_{j-1}) \cdot h_k^T)$$

Softmax Layer

Projection Layer

Encoder

Decoder
Seq2Seq + Span Copying

Scores for copying each span $\text{in}[i:j]; 0 \leq i < n, i < j \leq n$

$\begin{bmatrix}
S_{k,0:1} & S_{k,0:2} & \ldots & S_{k,0:n} \\
S_{k,1:2} & S_{k,1:n} & \ldots & \ldots \\
\vdots & \vdots & \ddots & \vdots \\
S_{k,n-1:n} & \end{bmatrix}$

$s_{k,[i:j]} = (W \cdot (r_i || r_{j-1}) \cdot h_k^T)$

Yields a token sequence of length $l$

$\alpha_k \rightarrow o_k \ldots o_{k+l}$

Scores for generating each $v \in \mathcal{V}$

Projection Layer

Softmax Layer

Action likelihoods
Seq2Seq + Span Copying

Scores for copying each span \( \text{in}[i:j]; \ 0 \leq i < n, \ i < j \leq n \)

\[
\begin{bmatrix}
S_{k,0:1} & S_{k,0:2} & \cdots & S_{k,0:n} \\
S_{k,1:2} & \cdots & S_{k,1:n} \\
\vdots & \ddots & \vdots \\
S_{k,n-1:n} & \cdots & \cdots & \cdots
\end{bmatrix}
\]

\[s_{k,[i:j]} = (W \cdot (r_i \| r_{j-1}) \cdot h_k^T)\]

\(\alpha_k \rightarrow o_k \ldots o_{k+l}\)

Yields a token sequence of length \(l\)

Scores for generating each \(v \in V\)

Action likelihoods

\[\text{Copy}_k(\text{in}[i:j], i:j) \quad \ldots \quad \text{Gen}_k(v) \quad \ldots\]
Seq2Seq + Span Copying

\[ \text{a b c d e} \rightarrow \text{a b f d e} \]
Seq2Seq + Span Copying

\[ a b c d e \rightarrow a b f d e \]

Decoder state is agnostic to predicted action
Seq2Seq + Span Copying – Training

Objective: Maximize \( p(o_0 \ldots o_m \mid \text{in}) \)

Zhou et al. (2018)
Predict any one of the correct actions at each step
\[
\prod_{0 \leq k \leq m} \sum_{\alpha \in A_k} q(\alpha \mid \text{in}, o_0 \ldots o_{k-1})
\]

- Local correctness ≠ global correctness
- Not explicitly encouraging the model to rely on fewer actions

\[a b c d e \rightarrow a b f d e\]
Seq2Seq + Span Copying – Training

**Objective**: Maximize $p(o_0 \ldots o_m | in)$

Zhou et al. (2018)
Predict any one of the correct actions at each step

$\prod_{0 \leq k \leq m} \sum_{\alpha \in A_k} q(\alpha | \text{in}, o_0 \ldots o_{k-1})$

- Local correctness ≠ global correctness
- Not explicitly encouraging the model to rely on fewer actions

We propose...
Marginalization over all possible correct action sequences that yield $o_0 \ldots o_m$

$a\ b\ c\ d\ e \rightarrow a\ b\ f\ d\ e$

**Diagram**

- **Gen(a)**
- **Copy(a, 0:1)**
- **Copy(a b, 0:2)**
- **Gen(b)**
- **Copy(b, 1:2)**
- **Gen(f)**
- **Gen(d)**
- **Copy(d, 3:4)**
- **Copy(d e, 3:5)**
- **Gen(e)**
- **Copy(e, 4:5)**
- **Gen(EOS)**
Seq2Seq + Span Copying – Training

**Objective:** Maximize $p(o_0 \ldots o_m | \text{in})$

```
abcede → abfde
```

- **Gen(a)**
  - Copy($a$, 0:1)
  - Copy($a b$, 0:2)

- **Gen(b)**
  - Copy($b$, 1:2)

- **Gen(f)**

- **Gen(d)**
  - Copy($d$, 3:4)
  - Copy($d e$, 3:5)

- **Gen(e)**
  - Copy($e$, 4:5)

- **Gen(EOS)**
Seq2Seq + Span Copying – Training

Objective: Maximize \( p(o_0 \ldots o_m \mid \text{in}) \)

\[
p(o_k \ldots o_m \mid \text{in}, o_0 \ldots o_{k-1}) = \sum_{\alpha \in A_k} q(\alpha \mid \text{in}, o_0 \ldots o_{k-1}) \times p(o_{k+l} \ldots o_m \mid o_0 \ldots o_{k+l-1})
\]

\[a b c d e \rightarrow a b f d e\]

Enumerating all possible correct action sequences that yield \(a b f d e\)

- Probability of generating correct suffix conditioned only on subsequence generated so far, not the concrete actions
- Encourages copying longer spans through fewer # of actions
Seq2Seq + Span Copying – Inference

Generate likely action sequences through beam search
- Action sequences of the same length could yield token sequences of varying length
  → Explicitly maintain token sequence length and “pause” expansion when needed
- Different action sequences could yield identical token sequences
  → “Merge” rays yielding identical token sequences

**Iteration 1:**

Beam = $[R_0: (SOS, 1.0)]$

Compute top 2 actions from $q(\alpha | in, SOS)$

- $Copy(a, 0:1) : 0.4$
  - $(SOS \ a, 1.0 \times 0.4)$
- $Copy(a \ b, 0:2) : 0.2$
  - $(SOS \ a \ b, 1.0 \times 0.2)$

Create new rays

$(SOS \ a, 0.4)$
$(SOS \ a \ b, 0.2)$
**Seq2Seq + Span Copying – Inference**

Generate likely action sequences through **beam search**
- Action sequences of the same length could yield token sequences of varying length
  → Explicitly maintain token sequence length and “pause” expansion when needed
- Different action sequences could yield identical token sequences
  → “**Merge**” rays yielding identical token sequences

**Iteration 2**: Beam = \([R_0: (\text{SOS } a, 0.4), R_1: (\text{SOS } a \ b, 0.2)]\)  \(l = 2\)  \(\text{in: a b c d e}\)  Beam width = 2

- Compute top 2 actions from \(q(a \mid \text{in, SOS } a)\)
  - \(Gen(b): 0.3\)
  - \(Copy(d, 3: 4): 0.1\)

- \(R_0: (\text{SOS } a, 0.4)\)
- \(R_1: (\text{SOS } a \ b, 0.2)\)

- Create new rays
  - \((\text{SOS } a \ b, 0.4 \times 0.3)\)
  - \((\text{SOS } a \ d, 0.4 \times 0.1)\)

- Merge rays
  - \((\text{SOS } a \ b, 0.32)\)
  - \((\text{SOS } a \ d, 0.04)\)

- Copy ray
  - \((\text{SOS } a \ b, 0.2)\)
Experimental Evaluation: Bug Fixing

**Bug fixing:** Given a faulty version of the code, generate the corrected version.

```
public boolean equals(Object obj) {
    return this.equals(obj);
}
```

```
public boolean equals(Object obj) {
    if (obj == null)
        return false;
    return this.equals(obj);
}
```

Bug-fix pair (BFP) datasets (*Tufano et al., 2019*) consisting of Java code snippets:
- **BFP**<sub>small</sub>: ≤ 50 code tokens
- **BFP**<sub>medium</sub>: 50-150 code tokens
Experimental Evaluation: Bug Fixing

**Baselines:**
- Tufano et al. (2019): Seq2Seq w/ no copy mechanism
- Seq2Seq + Token Copying: Seq2Seq w/ copying single tokens

Seq2Seq + Span Copying outperforms baselines, achieving new state-of-the-art on BFP datasets
Experimental Evaluation: Bug Fixing

Lengths of spans corresponding to *Copy* actions during beam decoding in log-y scale

**BFP\textsubscript{small}**:
- $\mu = 9.6$, median = 7
- 11.2% of *Copy* actions correspond to spans of length 1

**BFP\textsubscript{medium}**:
- $\mu = 29.5$, median = 19
- 27.1% of *Copy* actions correspond to spans of length 1

*Copy* actions tend to yield long copy spans
Experimental Evaluation: Bug Fixing

- Heuristic *Copy* action selection fails to capture the entire spectrum of correct actions.
Experimental Evaluation: Bug Fixing

- Heuristic *Copy* action selection fails to capture the entire spectrum of correct actions.
- Marginalization incentivizes the model to use as few actions as possible.
Experimental Evaluation: Learning Edit Representations

Learn to embed similar edit patterns nearby in a vector space

Alice Rumph was a painter, etcher, and teacher.

Alice Edith Rumph was a painter, etcher, and teacher.

… had a heart attack after he was pushed by a mugger.

… To the west of the Reiner crater.

… To the west of the Reiner crater on the moon.

Mark Taufa is an Australian professional rugby league player.

Mark Larry Taufa is an Australian professional rugby league player.

… had a heart attack after he was pushed by a mugger.

… had a heart attack after he was pushed by a mugger in the market.
Experimental Evaluation: Learning Edit Representations

Autoencoder-like model structure (*Yin et al., 2019*)
Experimental Evaluation: Learning Edit Representations

Datasets:
- WikiAtomicEdits (Faruqui et al., 2018)
- GitHubEdits (Yin et al., 2019)
- C# Fixers (Yin et al., 2019) (eval only)

Accuracy on Edit Representation Tasks

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Yin et al. (2019)</th>
<th>Seq2Seq + Token Copying</th>
<th>Seq2Seq + Span Copying</th>
</tr>
</thead>
<tbody>
<tr>
<td>WikiAtomicEdits</td>
<td>72.9</td>
<td>67.8</td>
<td>78.1</td>
</tr>
<tr>
<td>GitHubEdits</td>
<td>59.6</td>
<td>64.4</td>
<td>67.4</td>
</tr>
</tbody>
</table>

Span copying…
- Allows a model to more accurately predict edited text and code
Experimental Evaluation: Learning Edit Representations

Datasets:
- WikiAtomicEdits (Faruqui et al., 2018)
- GitHubEdits (Yin et al., 2019)
- C# Fixers (Yin et al., 2019) (eval only)

Span copying…
- Allows a model to more accurately predict edited text and code
- Facilitates learning more generalizable edit representations
Summary

• Span-copying mechanism which can be integrated with common encoder-decoder architectures.
• Marginalization for training encourages decoder to copy long spans.
• Beam search variant which is better suited for this setting.
• Approach leads to improved performance for editing tasks.