Copy That! Editing Sequences by Copying Spans

Sheena Panthaplackel*
The University of Texas at Austin

Miltiadis Allamanis   Marc Brockschmidt
Microsoft Research, Cambridge, UK

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

Intelligent Editors

Bug Fixing

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

→

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

(Tufano et al., 2019)

Code Refactoring

`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

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

→

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

(Tufano et al., 2019)

Code Refactoring

`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

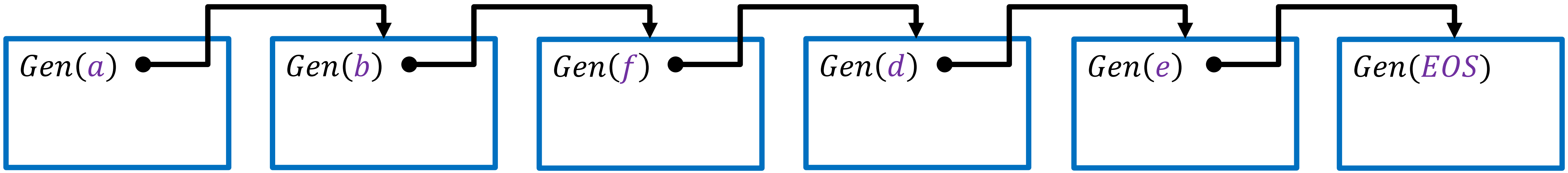
$\boldsymbol{in}$: input sequence
$\boldsymbol{out}$: output sequence ($o_0 \ldots o_m$)

$$p(o_0 \ldots o_m \mid \boldsymbol{in}) = \prod_{0 \le k \le m} p(o_k \mid \boldsymbol{in}, o_0 ... o_{k-1})$$

$$p(o_k \mid \boldsymbol{in}, o_0 ... o_{k-1}) = Gen(o_k)$$

Probability for generating o_k from vocabulary

a b c d e → a b f d e



Seq2Seq Generation

in: input sequence
out: output sequence ($o_0 \ldots o_m$)

$$p(o_0 \ldots o_m \mid \textbf{in}) = \prod_{1 \le k \le m} p(o_k \mid \textbf{in}, o_0 \ldots o_{k-1})$$

$$p(o_k \mid \textbf{in}, o_0 \ldots o_{k-1}) = Gen(o_k) + Copy(o_k, pos)$$

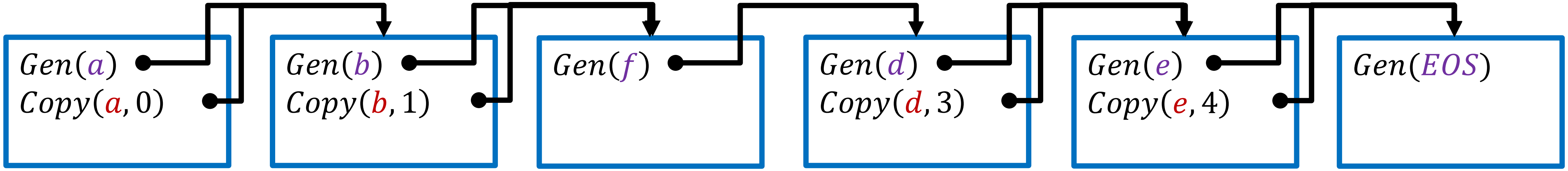
Probability for generating o_k from vocabulary
OR copying o_j from $\textbf{in}[pos]$
Pointer Networks (Vinyals et al., 2015)

$$= \sum_{\alpha \in A_k} q(\alpha \mid \textbf{in}, o_0 \ldots o_{k-1})$$

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

- Gen(token)
- $Copy$(token, pos)

a b c d e → a b f d e



Seq2Seq Generation

$\boldsymbol{in}$: input sequence
$\boldsymbol{out}$: output sequence ($o_0 \dots o_m$)

$$p(o_0 \dots o_m \mid \boldsymbol{in}) = \prod_{1 \le k \le m} p(o_k \mid \boldsymbol{in}, o_0 \dots o_{k-1})$$

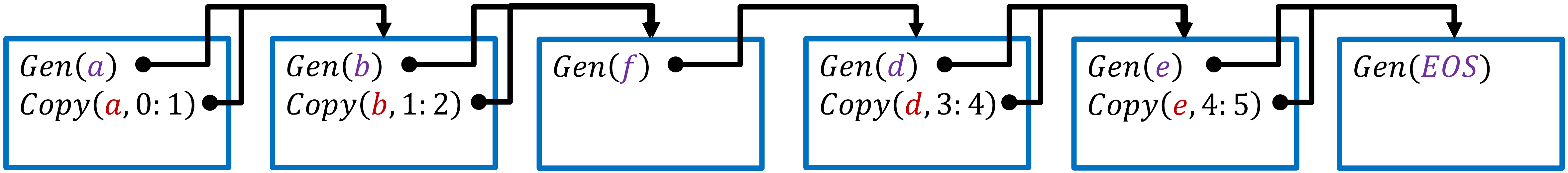
$$p(o_k \mid \boldsymbol{in}, o_1 \dots o_{k-1}) = Gen(o_k) + Copy(o_k, pos)$$

Probability for generating o_k from vocabulary
OR copying o_j from $\boldsymbol{in}[pos]$
Pointer Networks (Vinyals et al., 2015)

$$= \sum_{\alpha \in A_k} q(\alpha \mid \boldsymbol{in}, o_1 \dots o_{k-1})$$

Sum of all correct actions $\alpha \in A_k$ when decoding k^{th} token
Kinds of actions (α):
- Gen(token)
- ~~$Copy$(token, pos)~~ $Copy$(span of tokens, pos range)

a b c d e → a b f d e



Seq2Seq Generation

in: input sequence
out: output sequence ($o_0 \dots o_m$)

~~$p(o_0 \dots o_m \mid \boldsymbol{in}) = \prod_{1 \le k \le m} p(o_k \mid \boldsymbol{in}, o_0 \dots o_{k-1})$~~

$p(o_k \mid \boldsymbol{in}, o_1 \dots o_{k-1}) = Gen(o_k) + Copy(o_k, pos)$

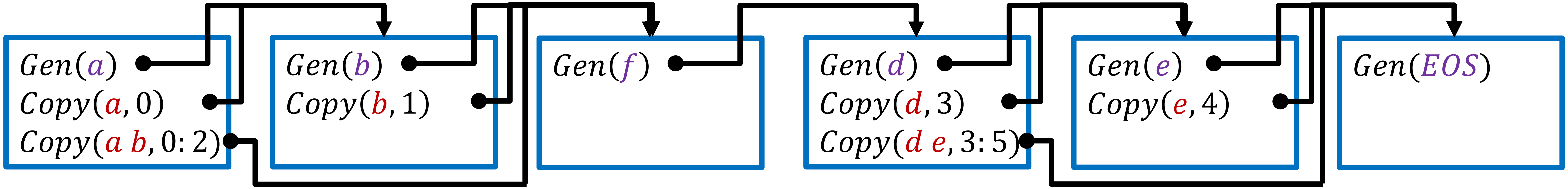
Probability for <u>generating</u> o_k from vocabulary
OR <u>copying</u> o_j from $\boldsymbol{in}[pos]$
Pointer Networks (Vinyals et al., 2015)

$= \sum_{\alpha \in A_k} q(\alpha \mid \boldsymbol{in}, o_1 \dots o_{k-1})$

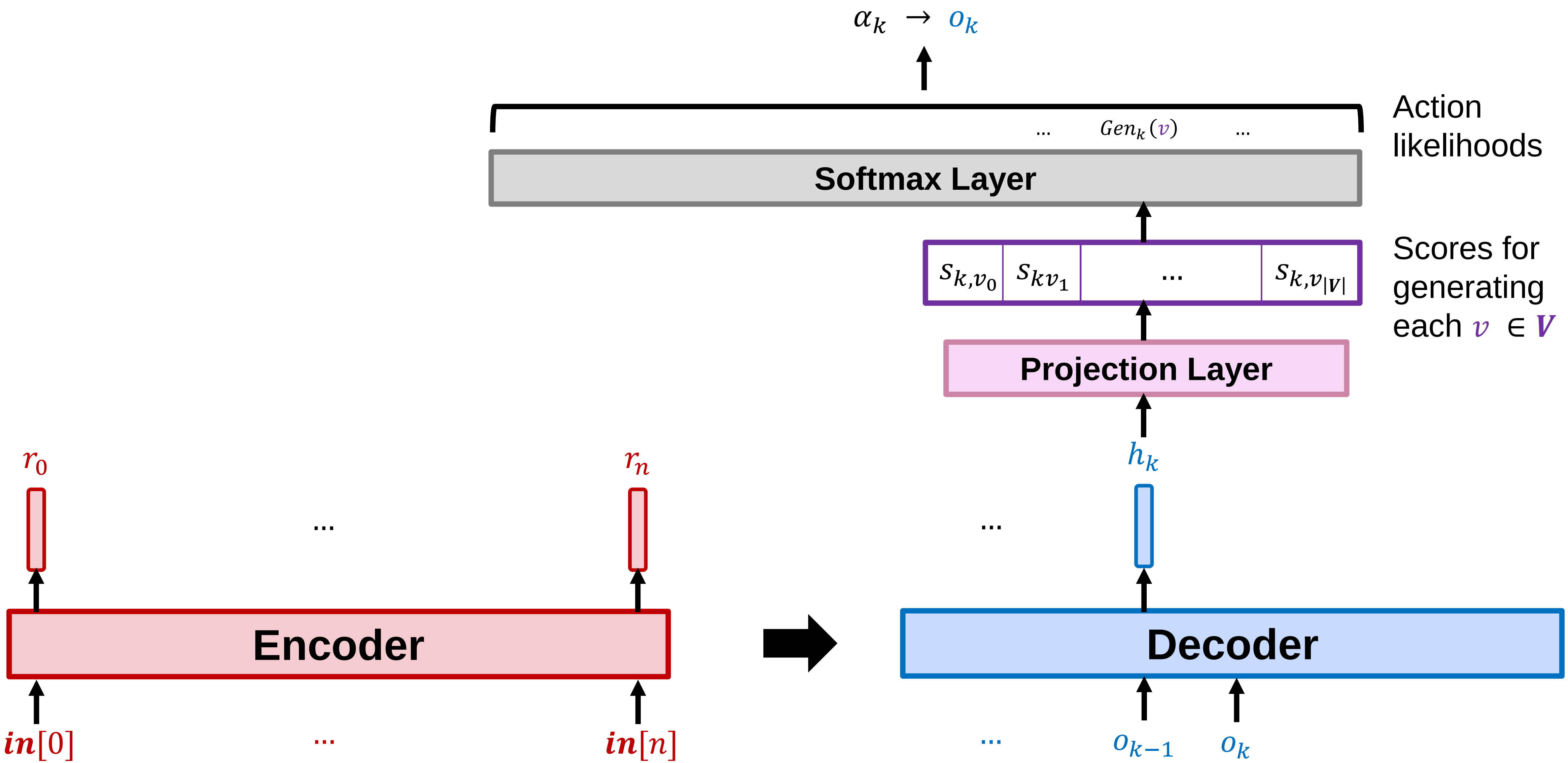
Sum of all correct actions $\alpha \in A_k$ when decoding k^{th} token
Kinds of actions (α):

- Gen(token)
- ~~$Copy$(token, pos)~~ $Copy$(span of tokens, pos range)

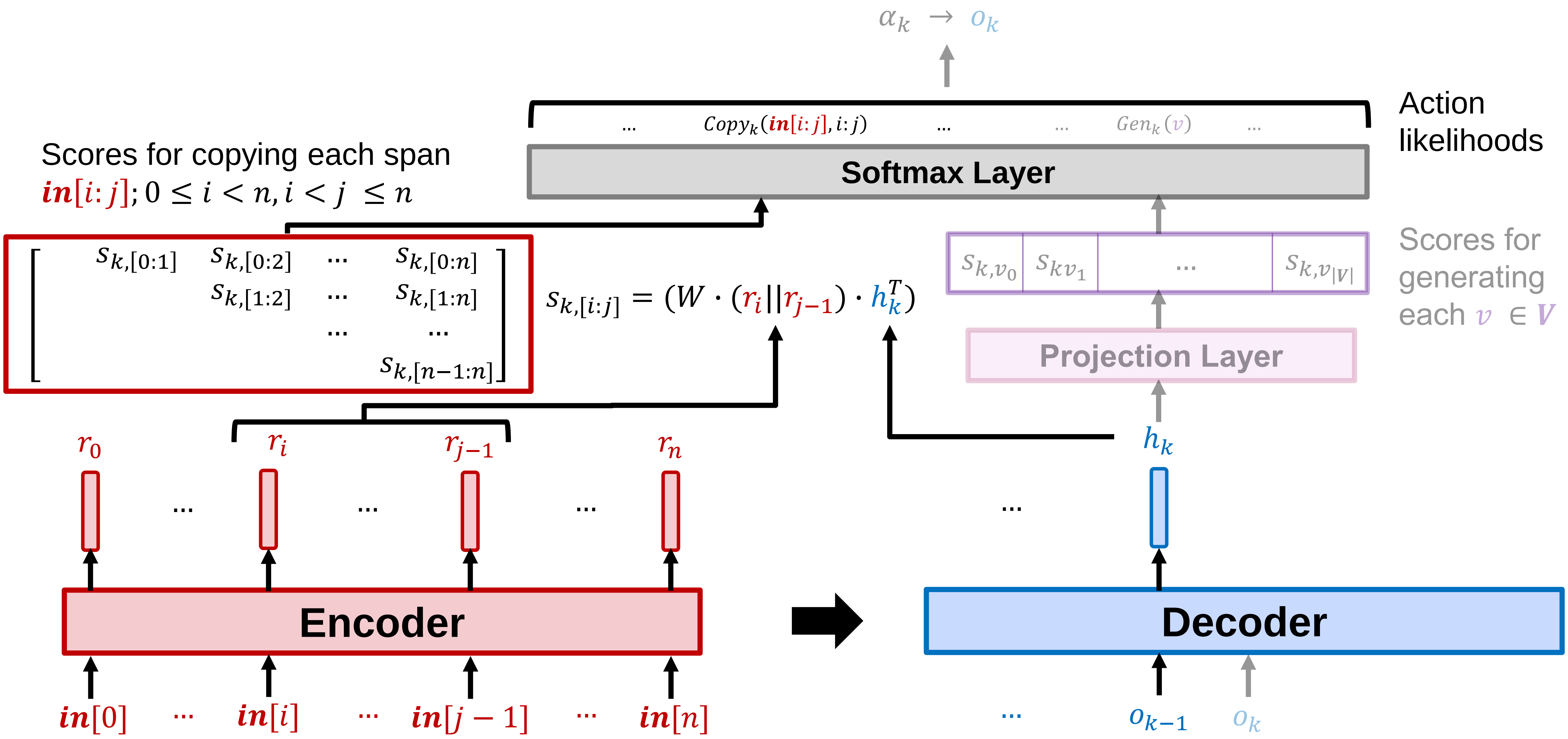
a b c d e → a b f d e



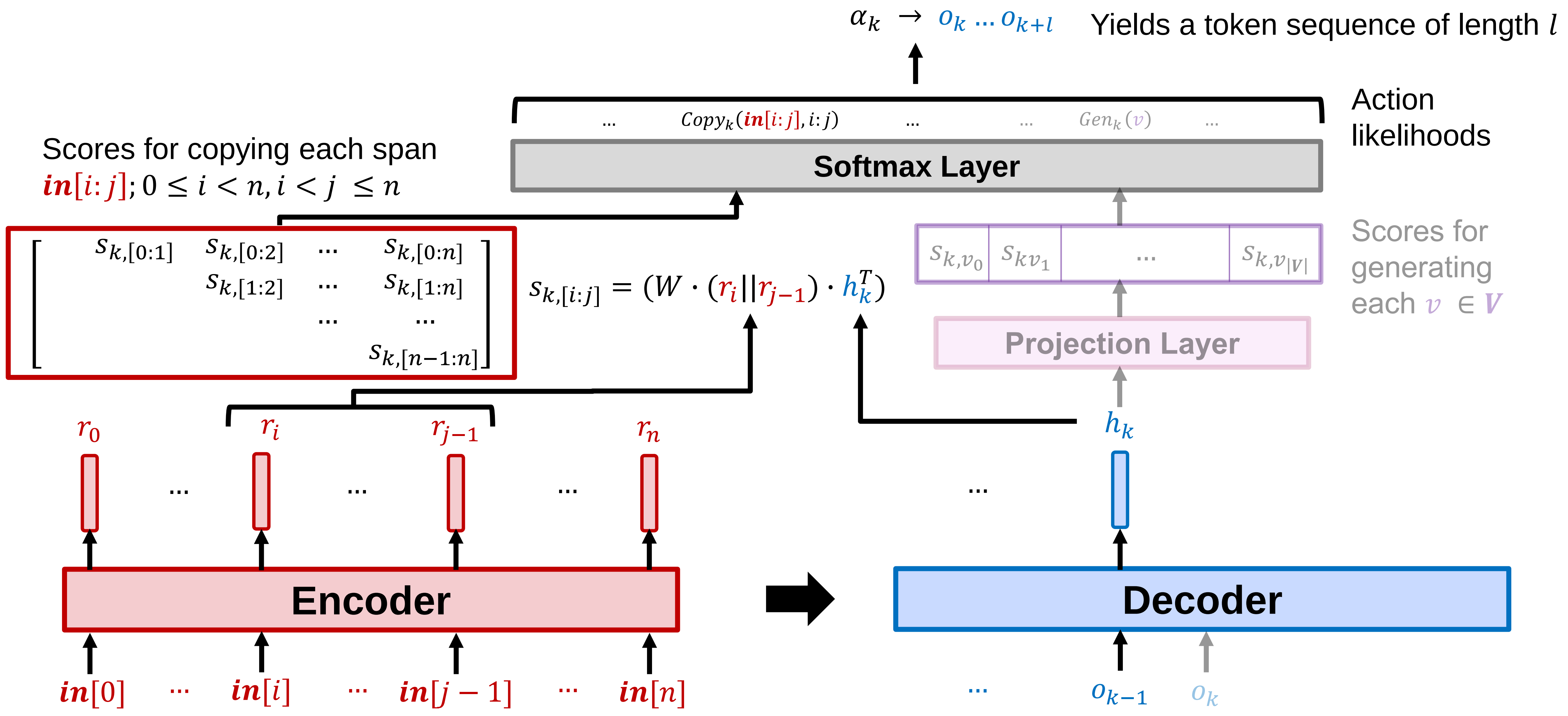
Seq2Seq + Span Copying



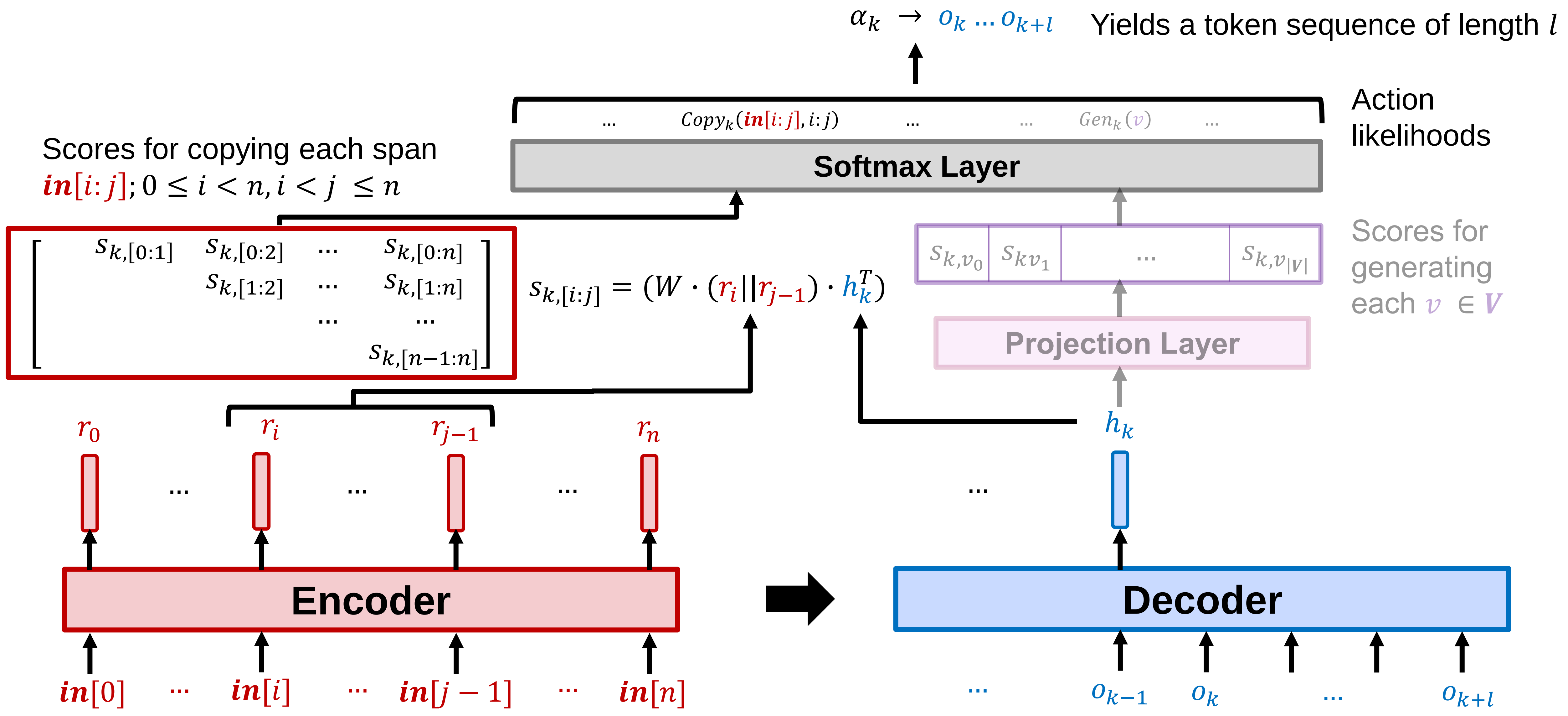
Seq2Seq + Span Copying



Seq2Seq + Span Copying

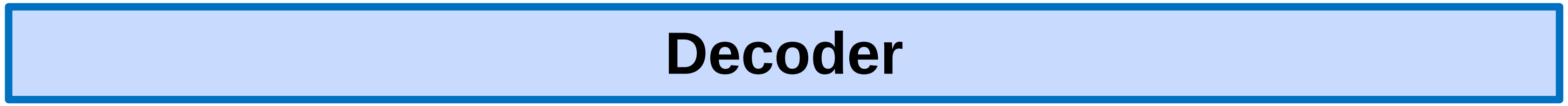


Seq2Seq + Span Copying



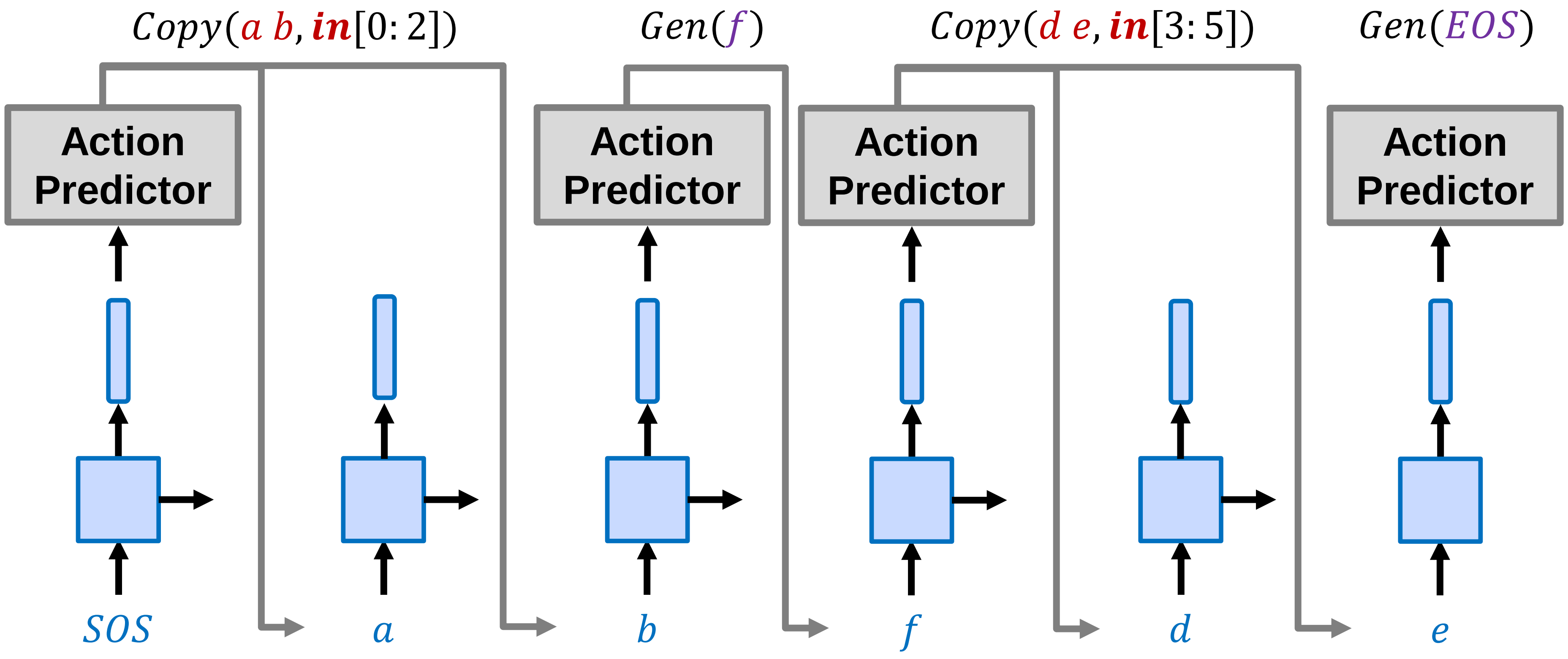
Seq2Seq + Span Copying

a b c d e → a b f d e



Seq2Seq + Span Copying

a b c d e → a b f d e



Decoder state is agnostic to predicted action

Seq2Seq + Span Copying – Training

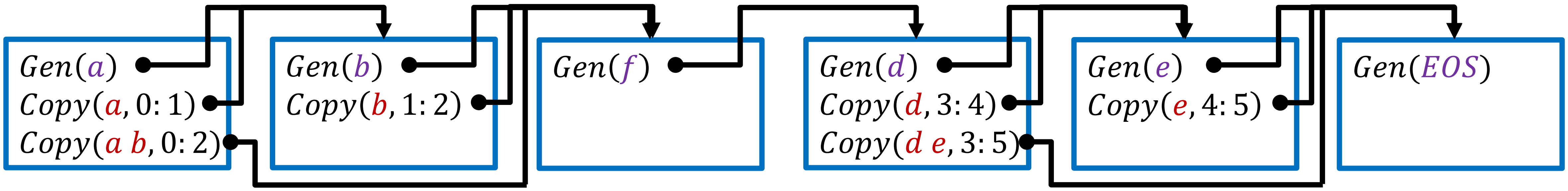
Objective: Maximize $p(o_0 \ldots o_m \mid \boldsymbol{in})$

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

$$\prod_{0 \le k \le m} \sum_{\alpha \in A_k} q(\alpha \mid \boldsymbol{in}, o_0 \ldots o_{k-1})$$

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

a b c d e → a b f d e



Seq2Seq + Span Copying – Training

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

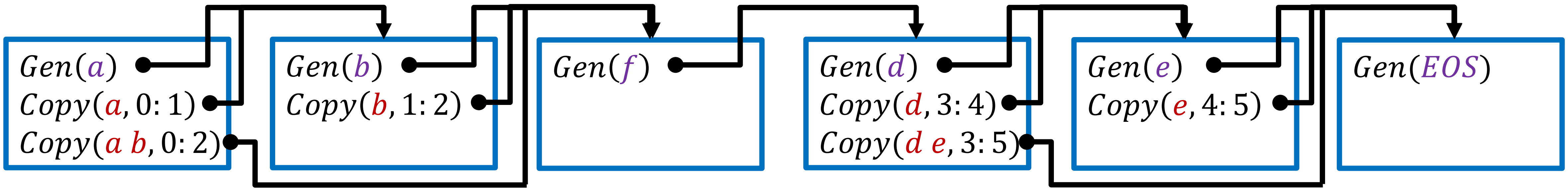
Zhou et al. (2018)
Predict any one of the correct actions at each step

$$\prod_{0 \le k \le m} \sum_{\alpha \in A_k} q(\alpha \mid \boldsymbol{in}, o_0 \ldots o_{k-1})$$

- 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 $o_0 \ldots o_m$

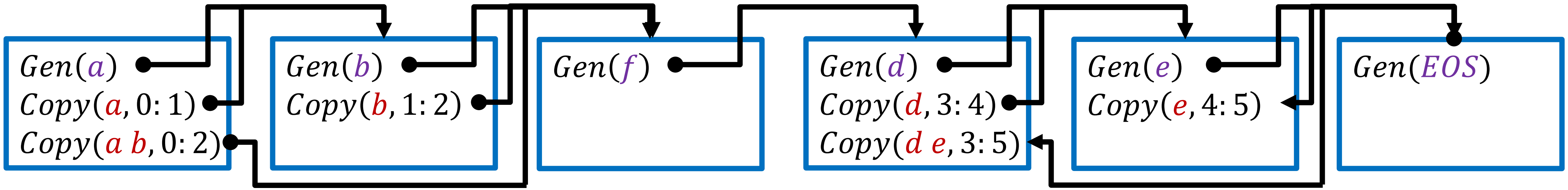
a b c d e → a b f d e



Seq2Seq + Span Copying – Training

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

a b c d e → a b f d e



Seq2Seq + Span Copying – Training

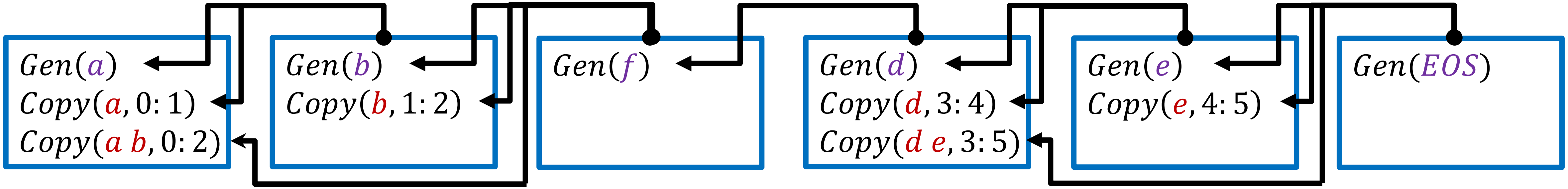
Objective: Maximize $p(o_0 \dots o_m \mid \mathbf{in})$

$$p(o_k \dots o_m \mid \mathbf{in}, o_0 \dots o_{k-1}) = \sum_{\substack{\alpha \in A_k \\ l = [\![\alpha]\!]}} q(\alpha \mid \mathbf{in}, o_0 \dots o_{k-1}) \times p(o_{k+l} \dots o_m \mid o_0 \dots o_{k+l-1})$$

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

a b c d e → a b f d e

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



$p(EOS \mid \mathbf{in}, a\ b\ f\ d\ e)$

$p(e \mid \mathbf{in}, a\ b\ f\ d)$

$p(d\ e \mid \mathbf{in}, a\ b\ f)$

$p(f\ d\ e \mid \mathbf{in}, a\ b) = [Gen(f) \times p(d\ e \mid \mathbf{in}, a\ b\ f)]$

$p(b\ f\ d\ e \mid \mathbf{in}, a) = [Gen(b) \times p(f\ d\ e \mid \mathbf{in}, a\ b)] + [Copy(b, 1:2) \times p(f\ d\ e \mid \mathbf{in}, a\ b)]$

$$p(a\ b\ f\ d\ e \mid \mathbf{in}) = [Gen(a) \times p(b\ f\ d\ e \mid \mathbf{in}, a)] + [Copy(a, 0:1) \times p(b\ f\ d\ e \mid \mathbf{in}, a)] + [Copy(a\ b, 0:2) \times p(f\ d\ e \mid \mathbf{in}, a\ b)]$$

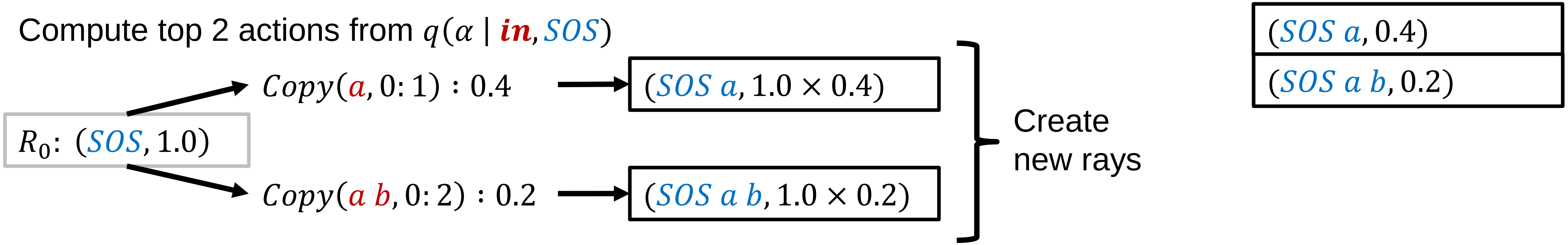
Seq2Seq + Span Copying – Inference

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 $= [R_0\colon (SOS, 1.0)]$ $l = 1$ ***in***: a b c d e Beam width = 2

Compute top 2 actions from $q(\alpha \mid \boldsymbol{in}, SOS)$

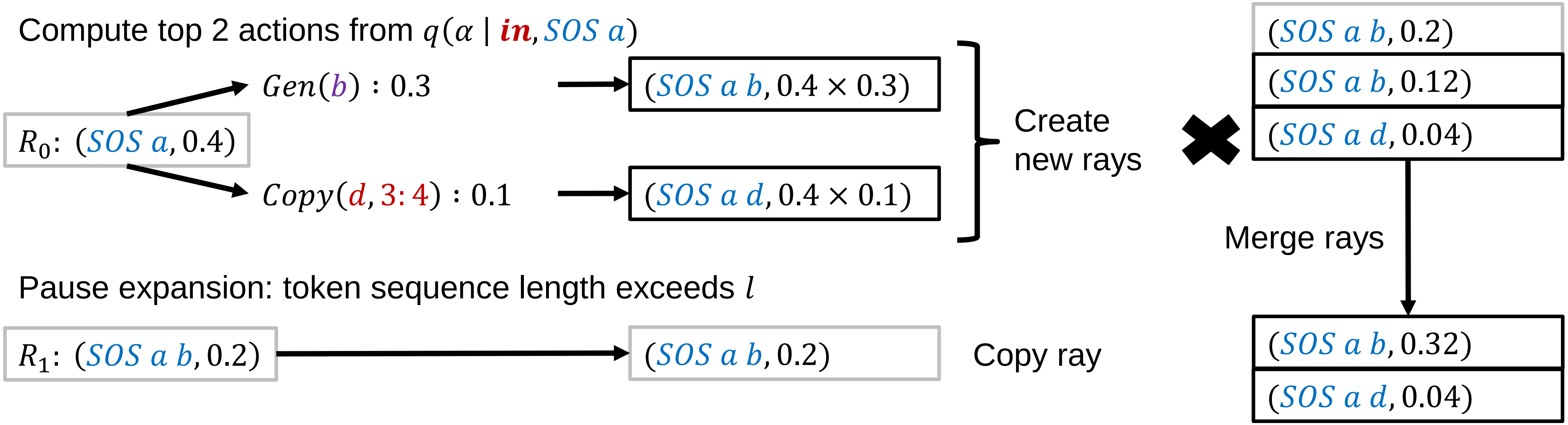


Seq2Seq + Span Copying – Inference

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



Experimental Evaluation: Bug Fixing

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

in

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

out

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

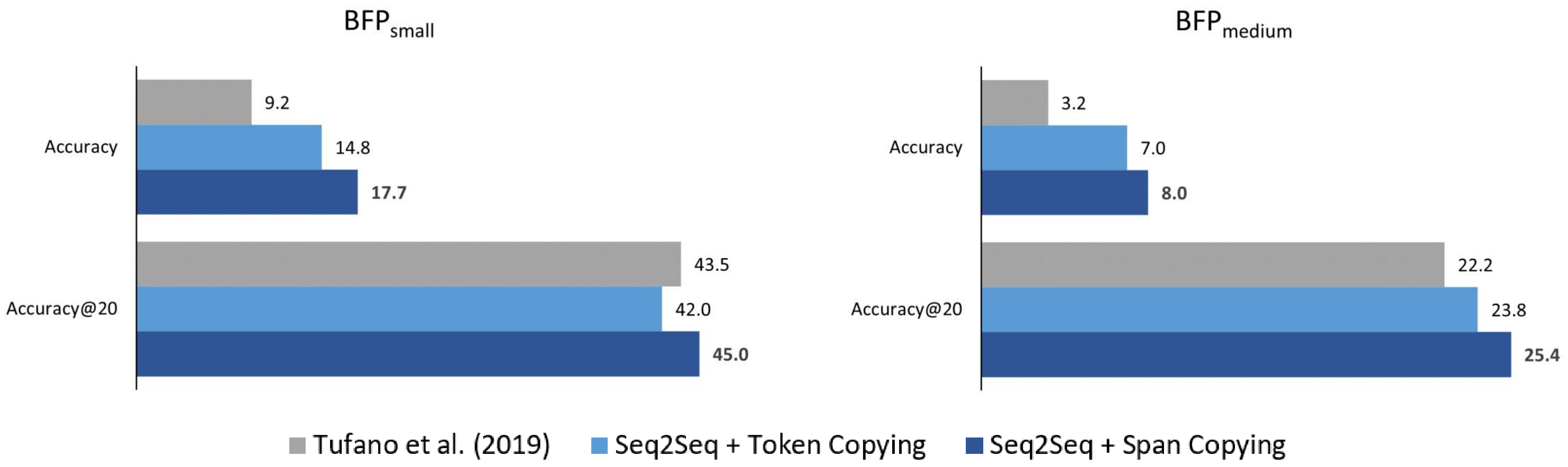
Bug-fix pair (BFP) datasets *(Tufano et al., 2019*) consisting of Java code snippets:

- $\textbf{BFP}_{\textbf{small}}$: ≤ 50 code tokens
- $\textbf{BFP}_{\textbf{medium}}$: 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



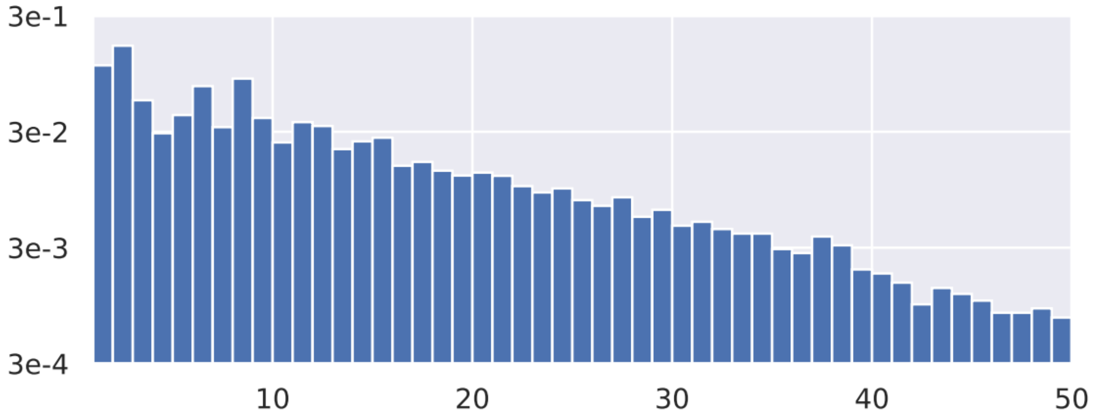
Seq2Seq + Span Copying outperforms baselines, achieving new state-of-the-art on BFP datasets

Experimental Evaluation: Bug Fixing

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

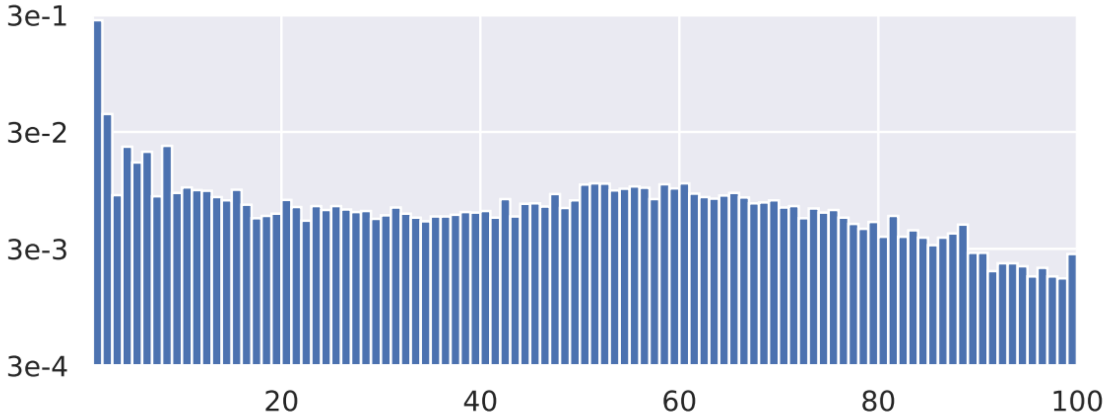
BFP_{small}:

- μ = 9.6, median = 7
- 11.2% of $Copy$ actions correspond to spans of length 1



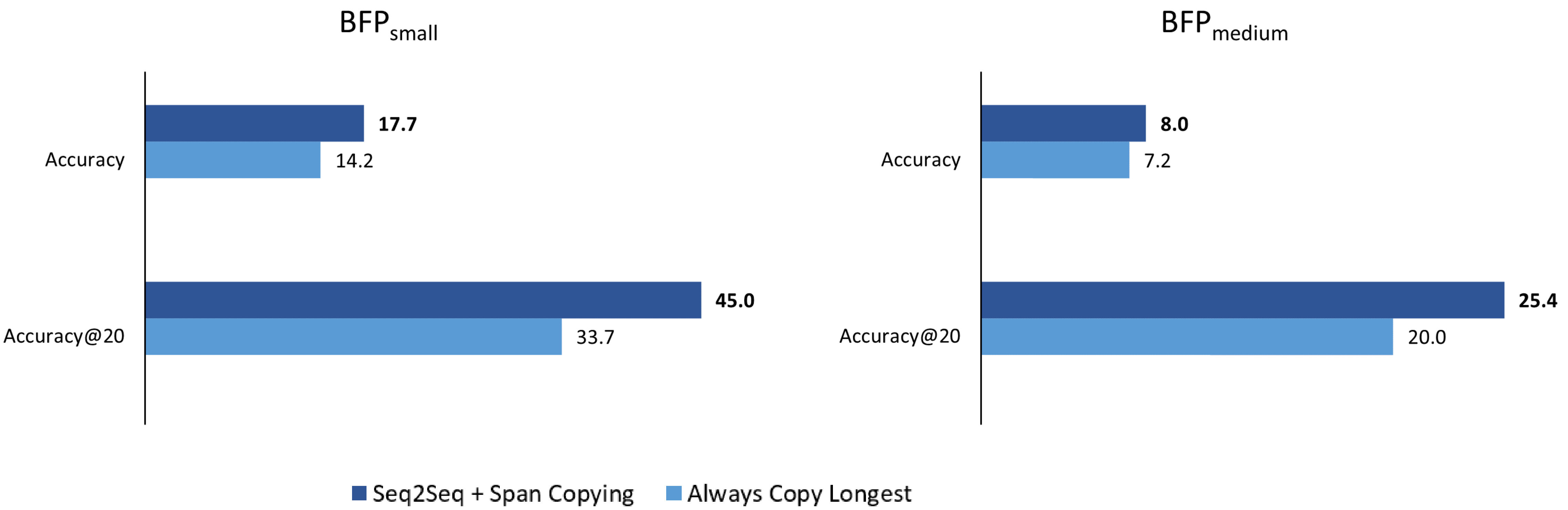
BFP_{medium}:

- μ = 29.5, median = 19
- 27.1% of $Copy$ actions correspond to spans of length 1



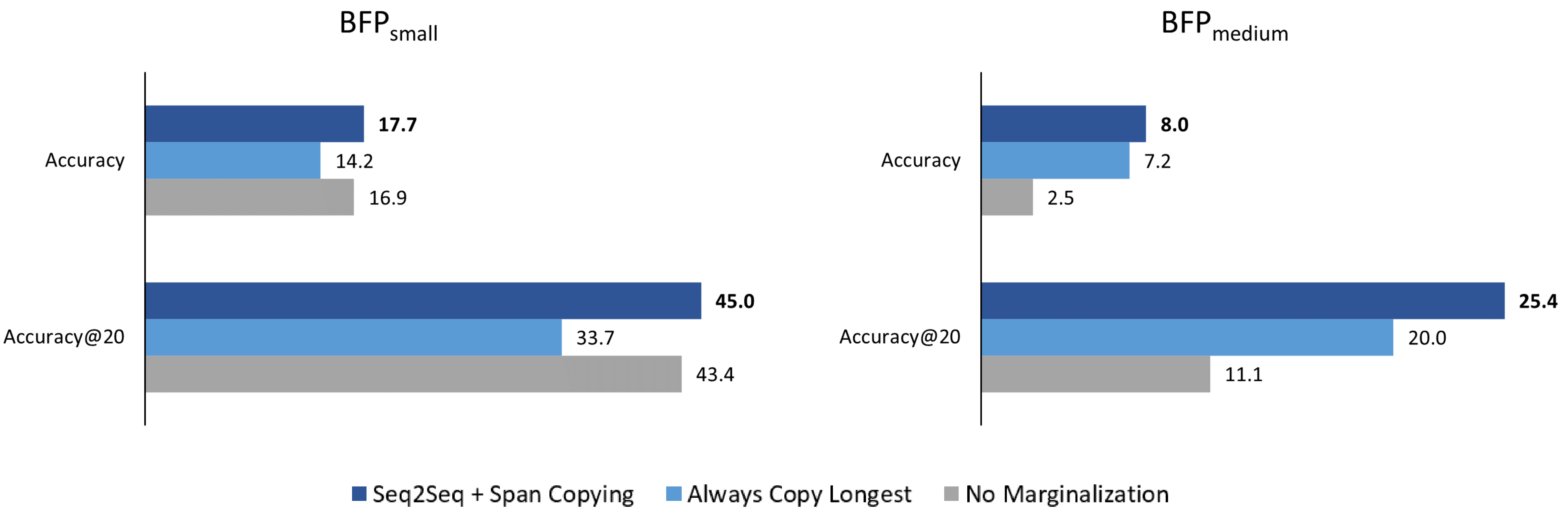
$Copy$ actions tend to yield long copy spans

Experimental Evaluation: Bug Fixing



- **Heuristic *Copy* action selection fails to capture the entire spectrum of correct actions.**

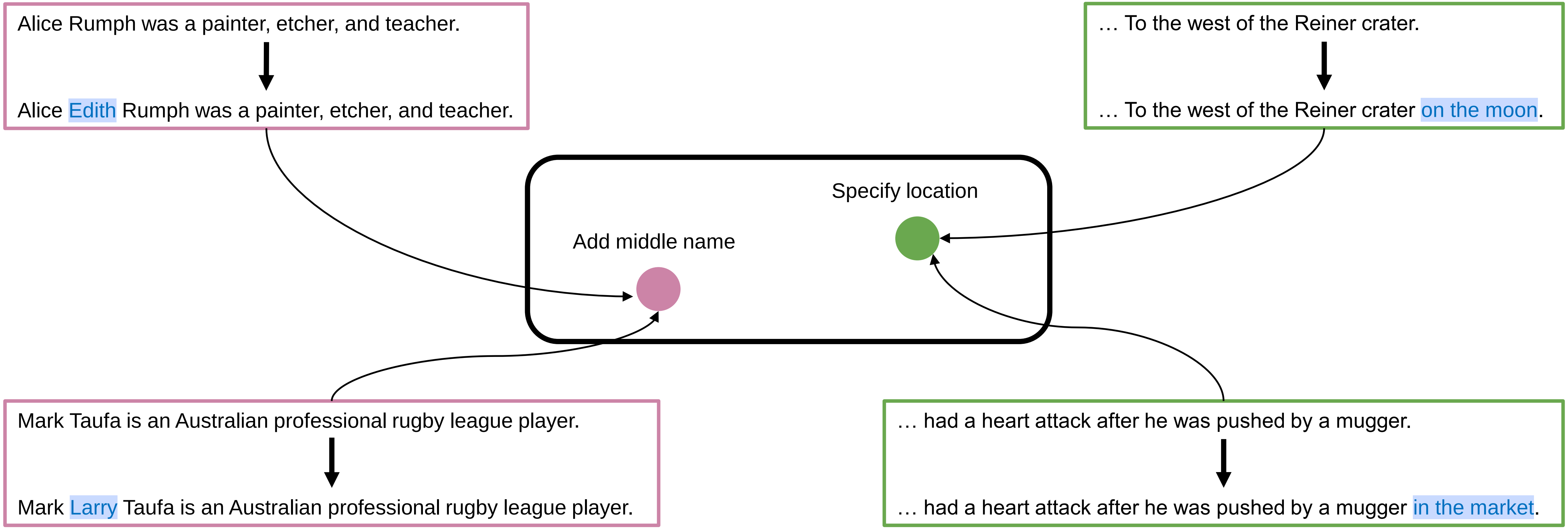
Experimental Evaluation: Bug Fixing



- **Heuristic *Copy* action selection fails to capture the entire spectrum of correct actions.**
- **Marginalization incentivizes the model to use as few actions as possible.**

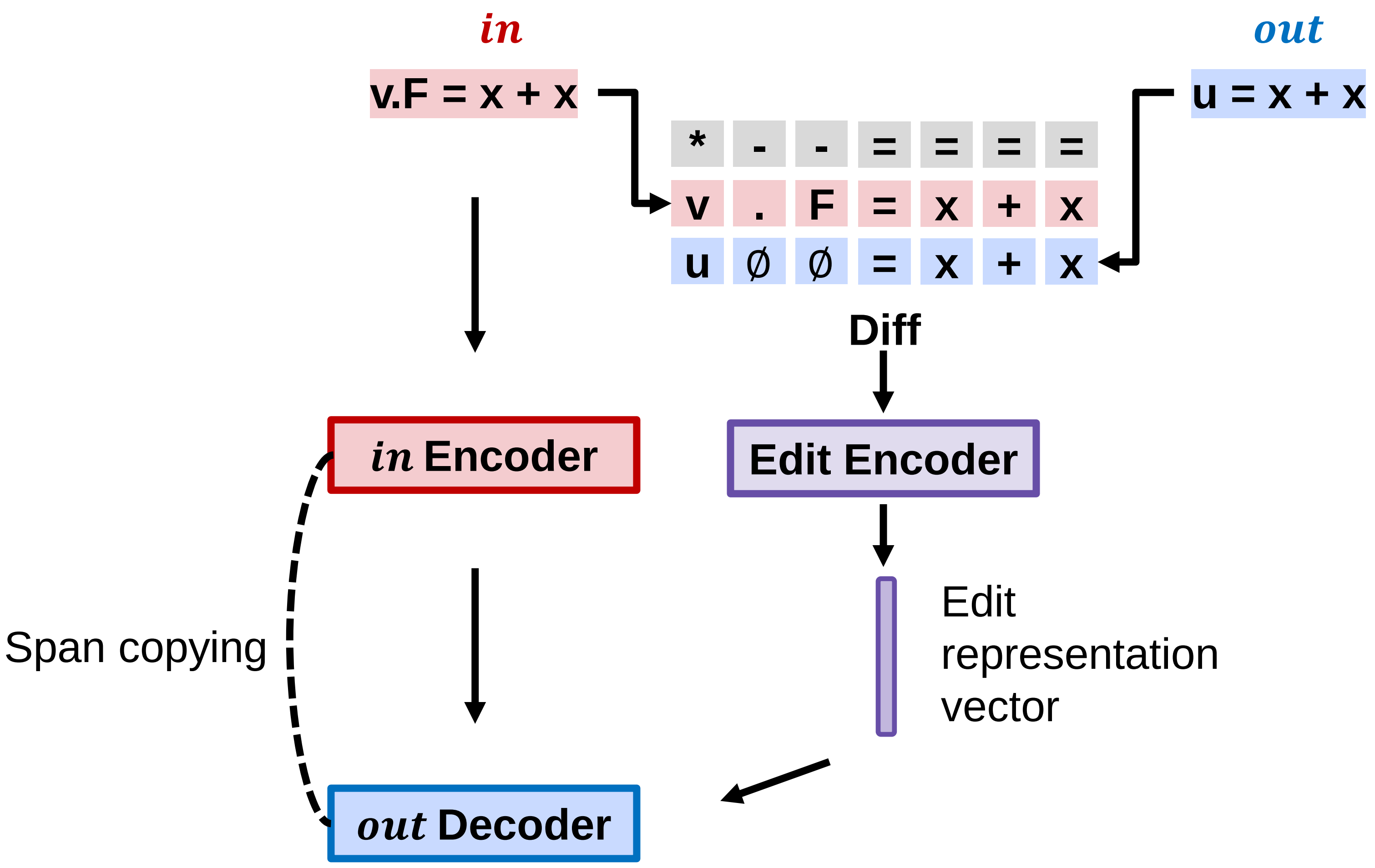
Experimental Evaluation: Learning Edit Representations

Learn to embed similar edit patterns nearby in a vector space



Experimental Evaluation: Learning Edit Representations

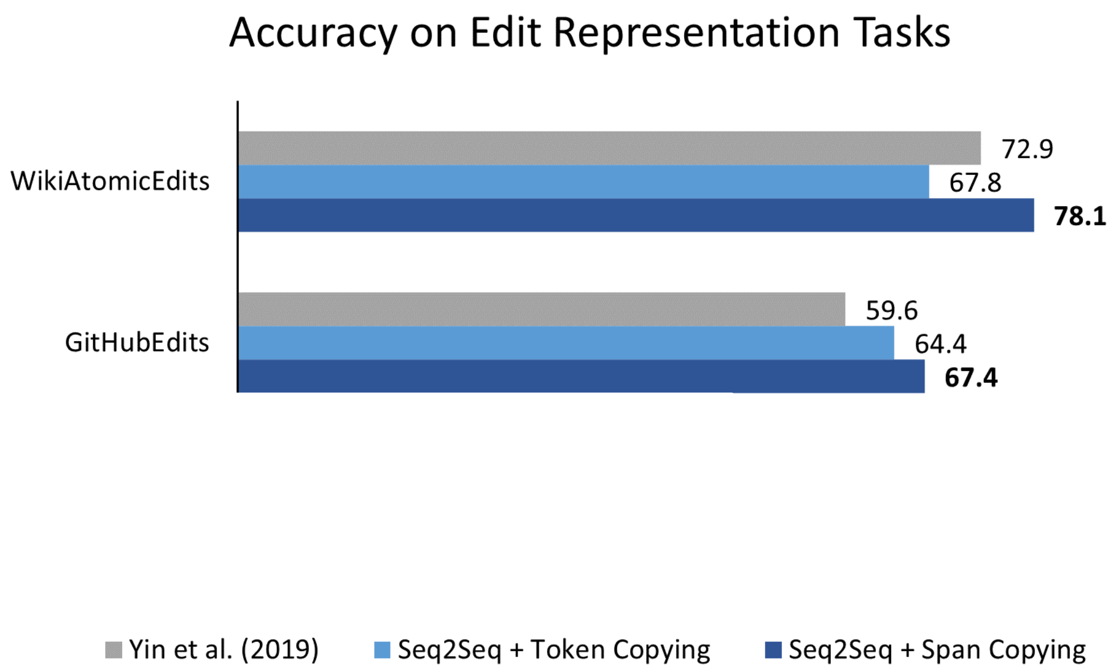
Autoencoder-like model structure *(Yin et al., 2019)*



Experimental Evaluation: Learning Edit Representations

Datasets:

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



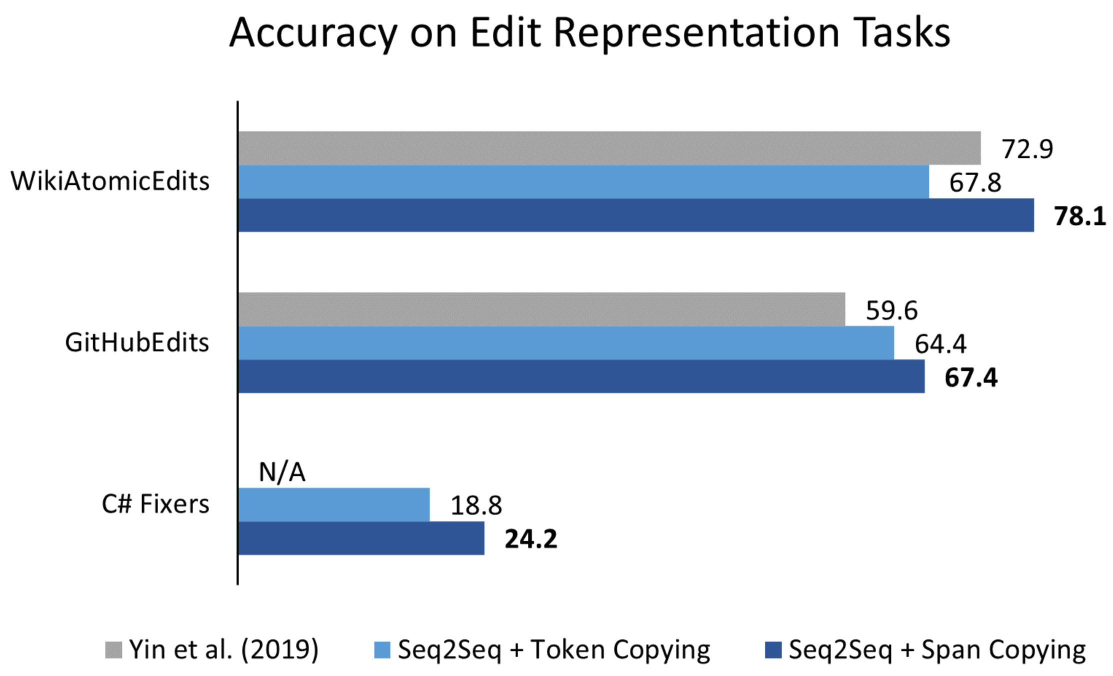
Span copying…

- 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.

Sheena Panthaplackel
The University of Texas at Austin
TEXAS
The University of Texas at Austin

Miltiadis Allamanis   Marc Brockschmidt
Microsoft Research, Cambridge, UK
Microsoft