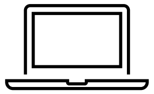
From *Partial* to *Strictly* Incremental Constituent Parsing

Ana Ezquerro   Carlos Gómez-Rodríguez   David Vilares

Funded by the European Union
erc European Research Council Established by the European Commission
citic
UNIVERSIDADE DA CORUÑA
MINISTERIO DE CIENCIA E INNOVACIÓN
UNIÓN EUROPEA
FONDO EUROPEO DE DESARROLLO REGIONAL
"Una manera de hacer Europa"
AGENCIA ESTATAL DE INVESTIGACIÓN

How do SoTA parsers work?

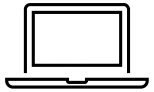


State-of-the-art System

How do SoTA parsers work?

I have a flight to Malta

Full sentence

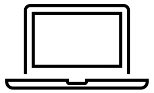


State-of-the-art System

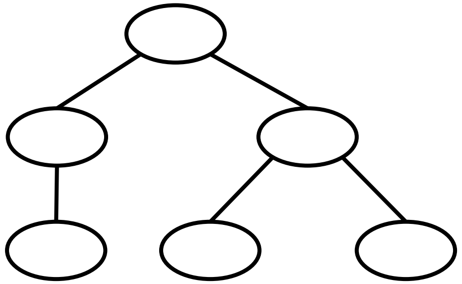
How do SoTA parsers work?

I have a flight to Malta

Full sentence



State-of-the-art System

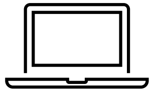


Full syntactic tree

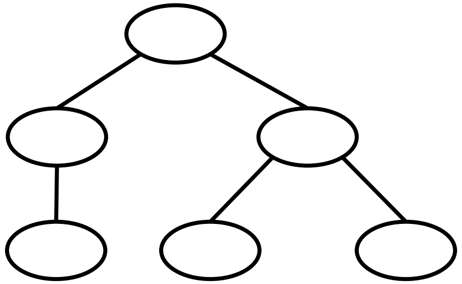
How do SoTA parsers work?

I have a flight to Malta

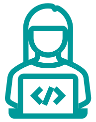
Full sentence



State-of-the-art System



Full syntactic tree

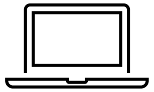


- Non incremental.
- Simultaneous access to all elements of the sentence.
- No reference of the information provided by each word to build the tree.

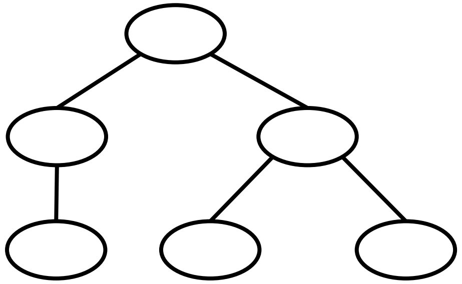
Human-like Incremental Parsing

I have a flight to Malta

Full sentence



State-of-the-art System



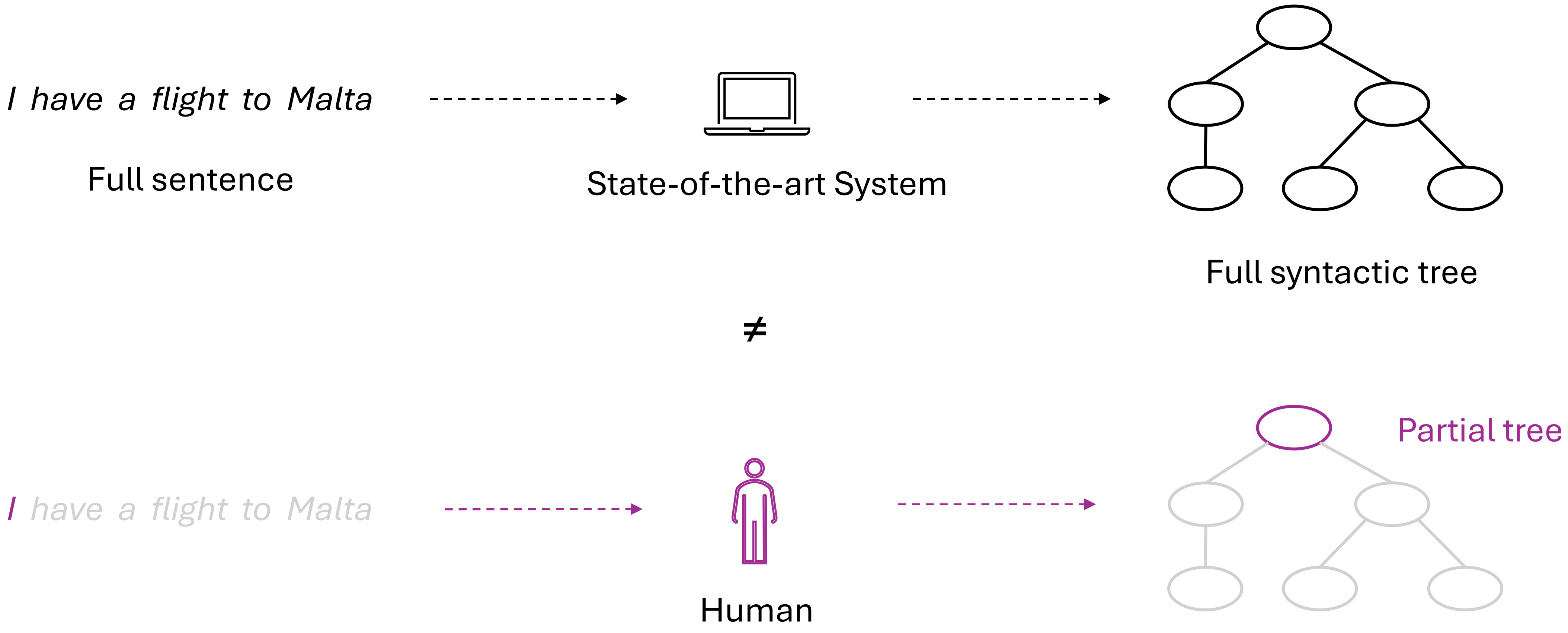
Full syntactic tree

≠

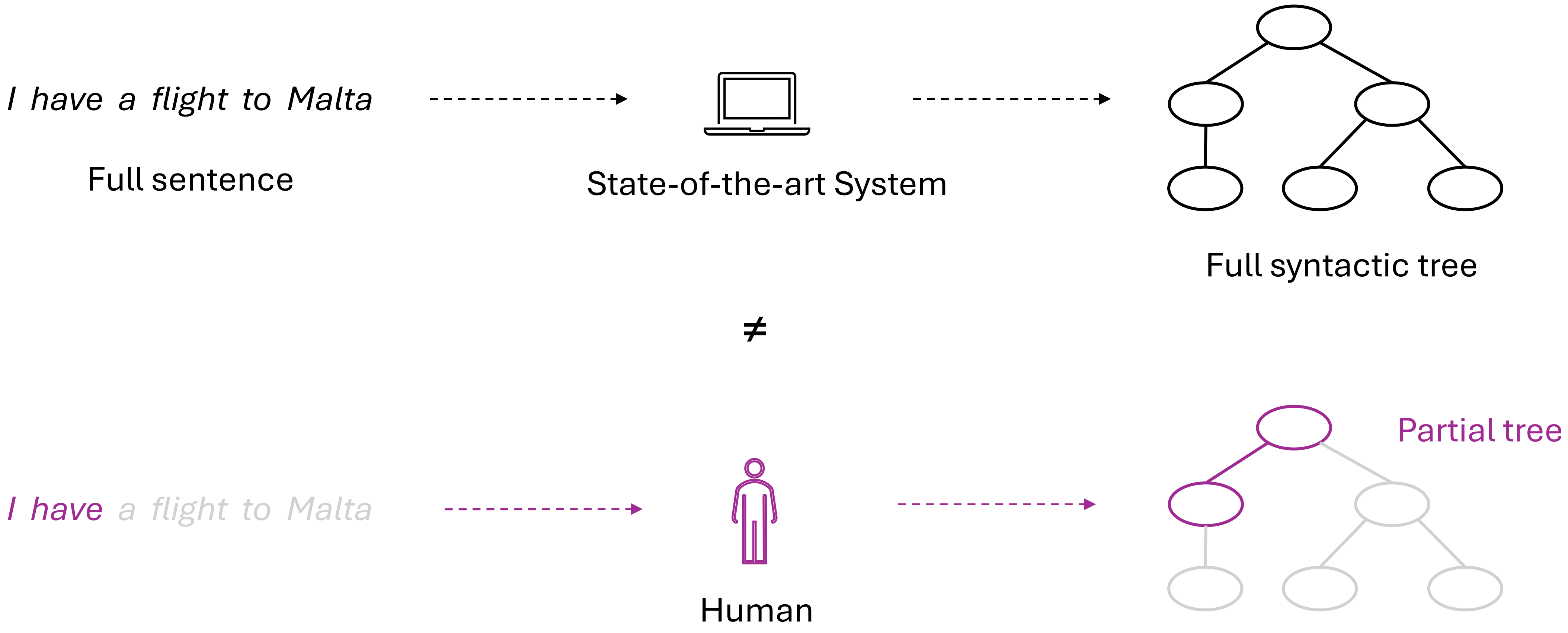
?

Human

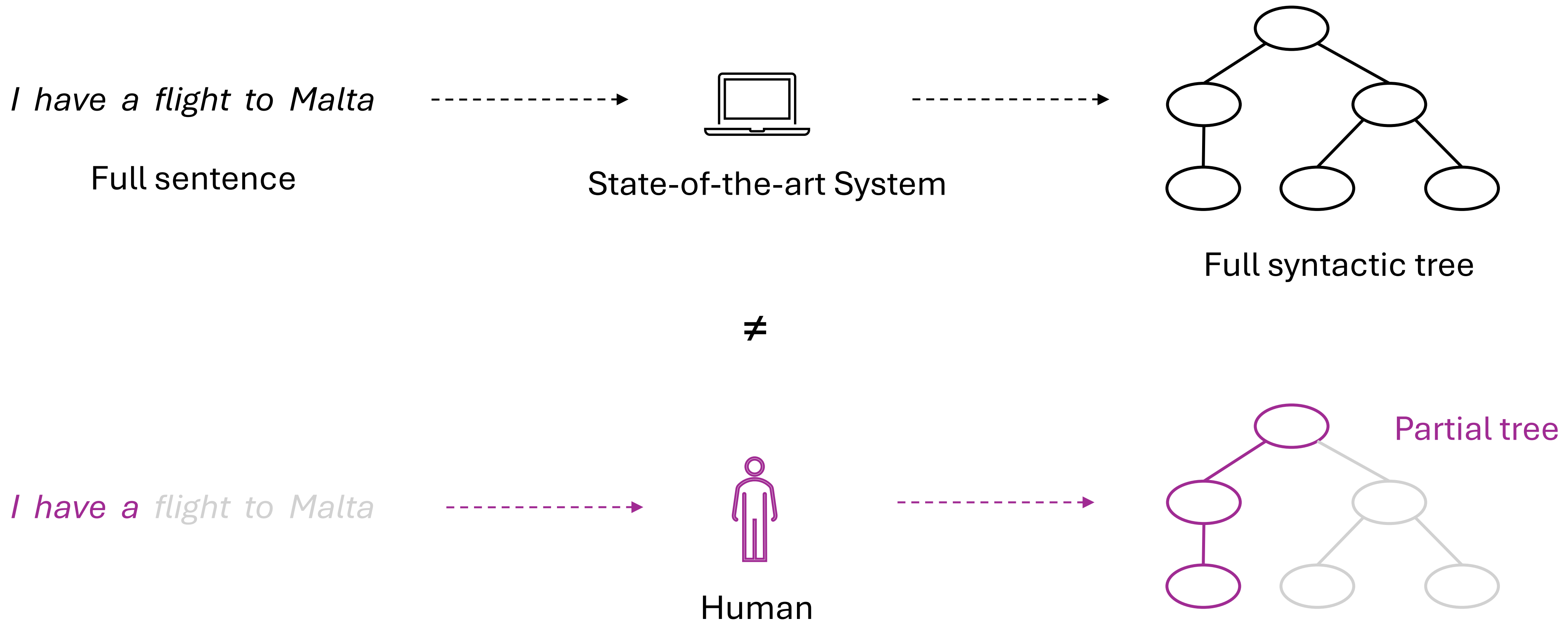
Human-like Incremental Parsing



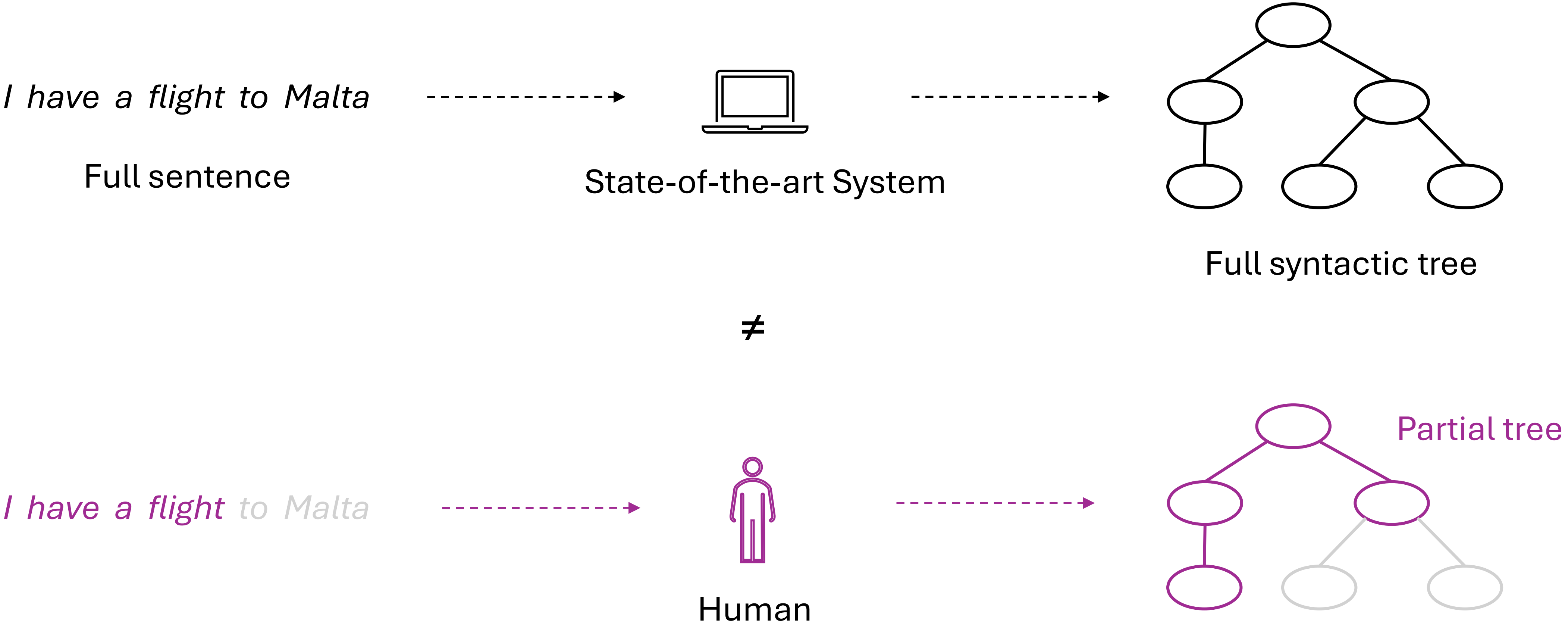
Human-like Incremental Parsing



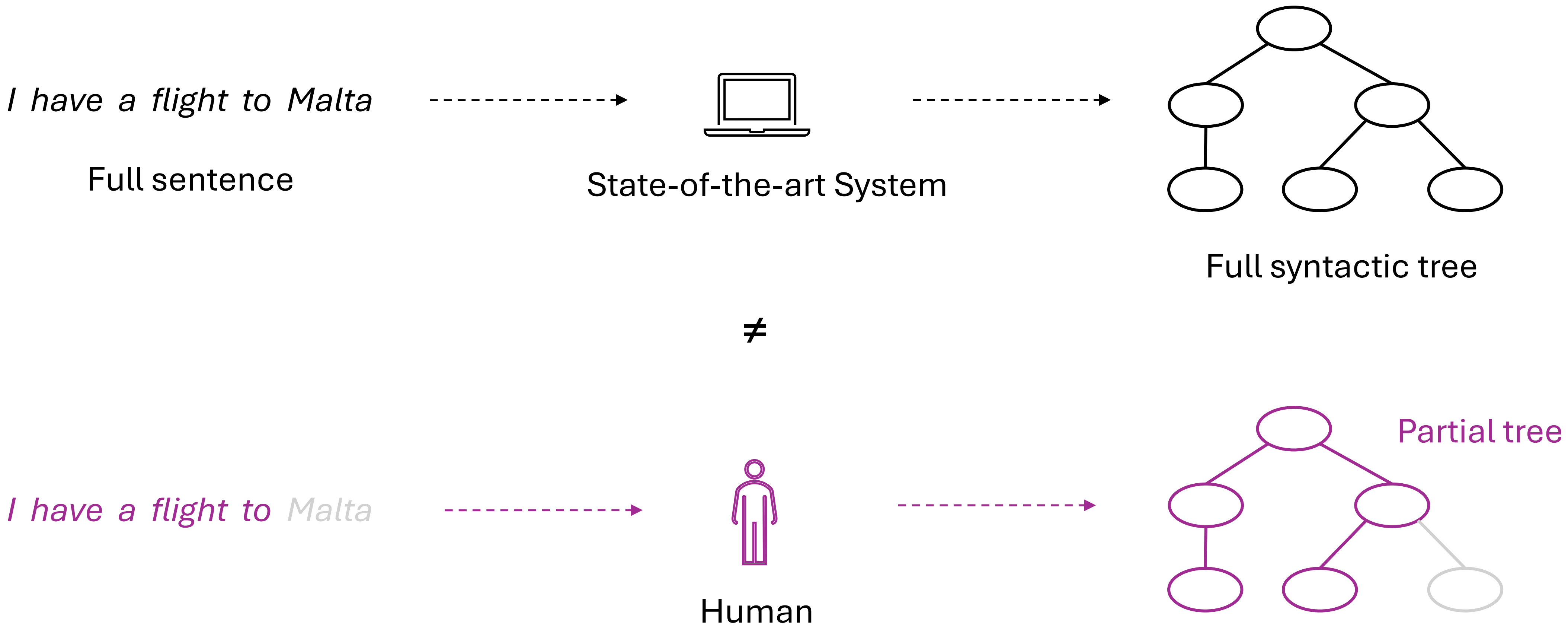
Human-like Incremental Parsing



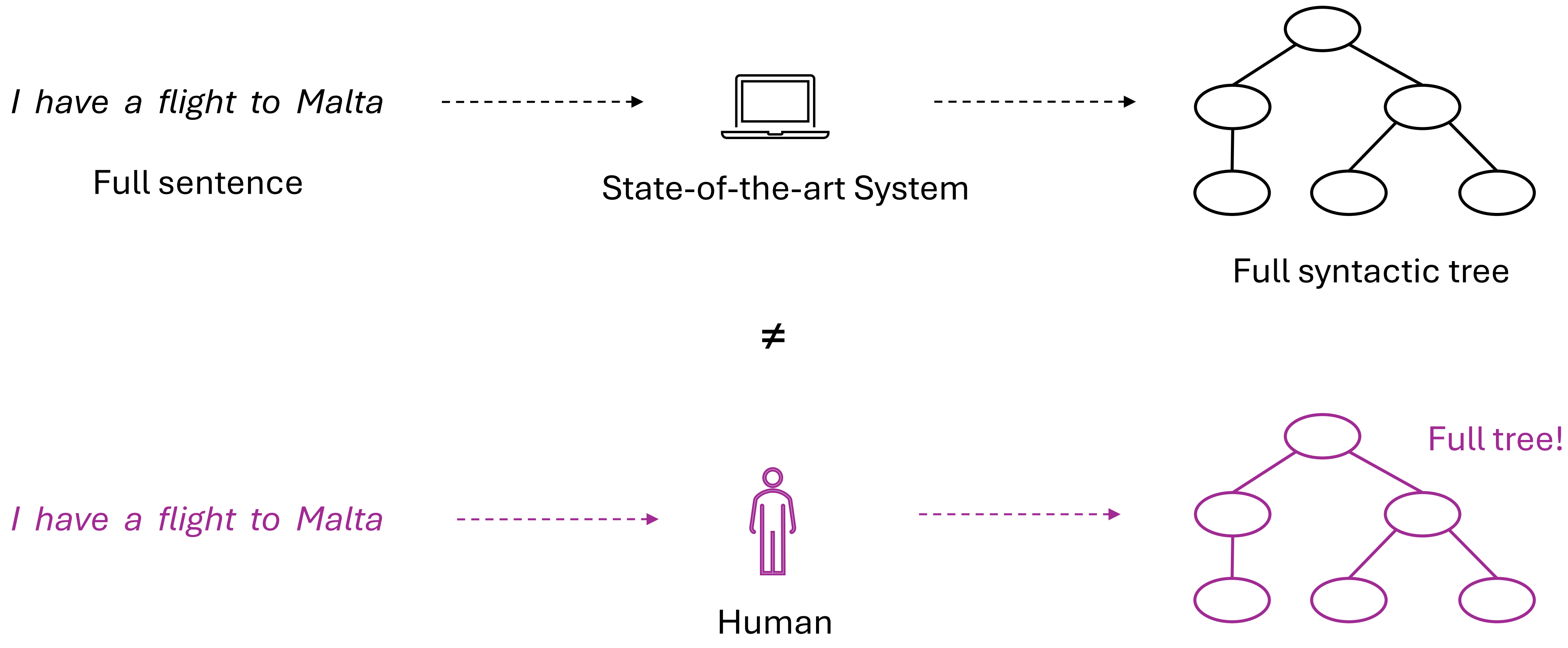
Human-like Incremental Parsing



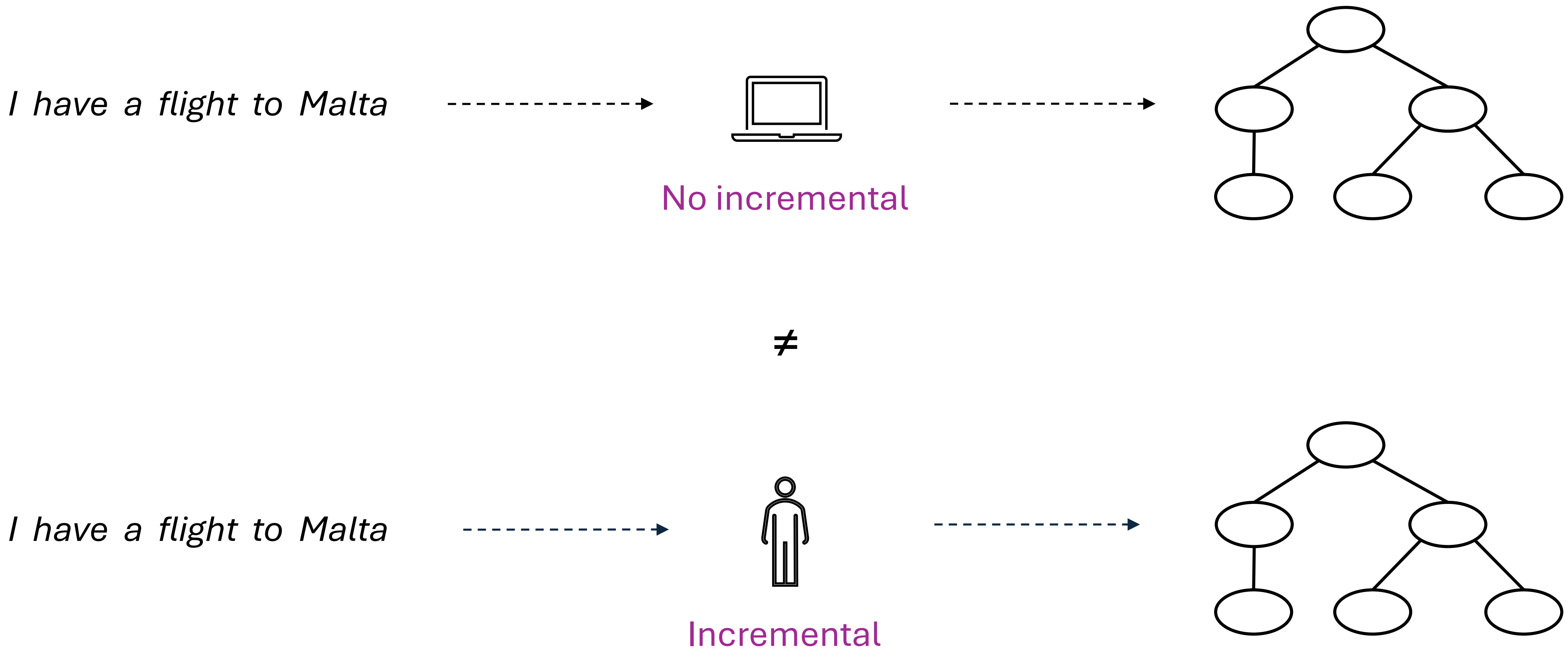
Human-like Incremental Parsing



Human-like Incremental Parsing



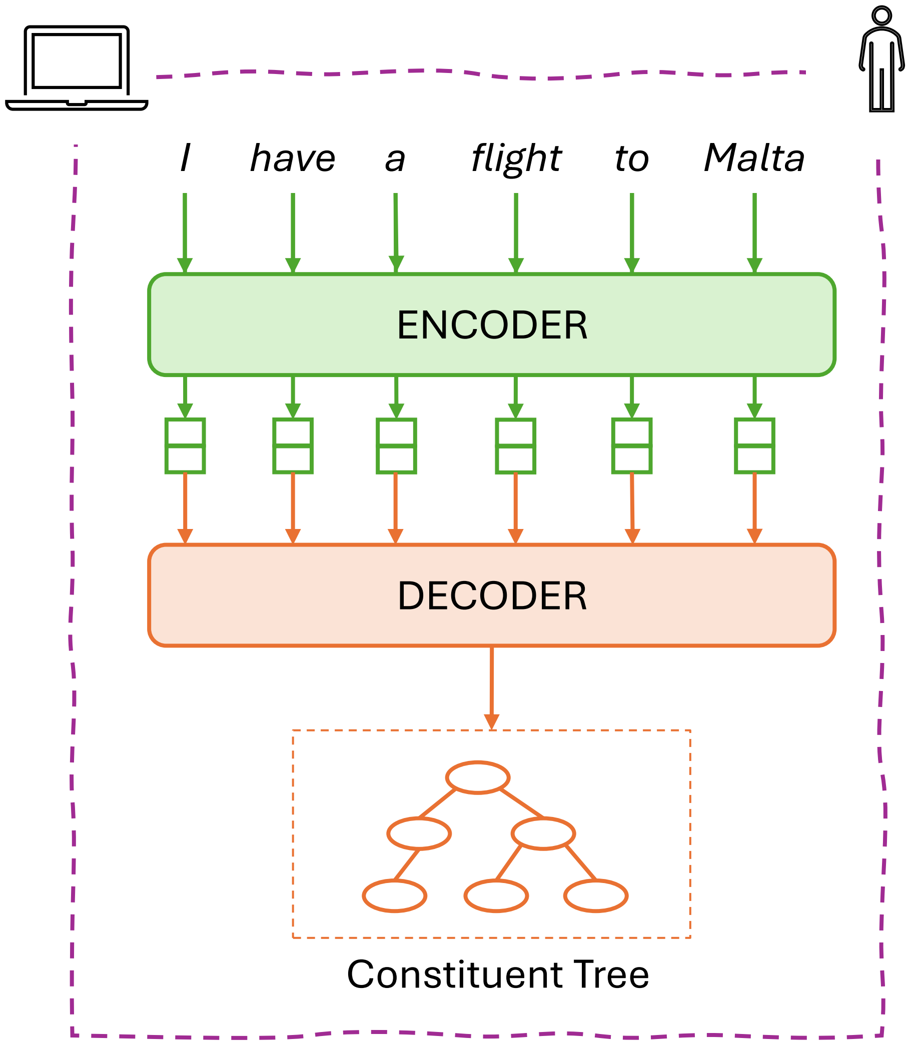
Human-like Incremental Parsing



Incremental *Constituent* Parsing

State-of-the-Art System

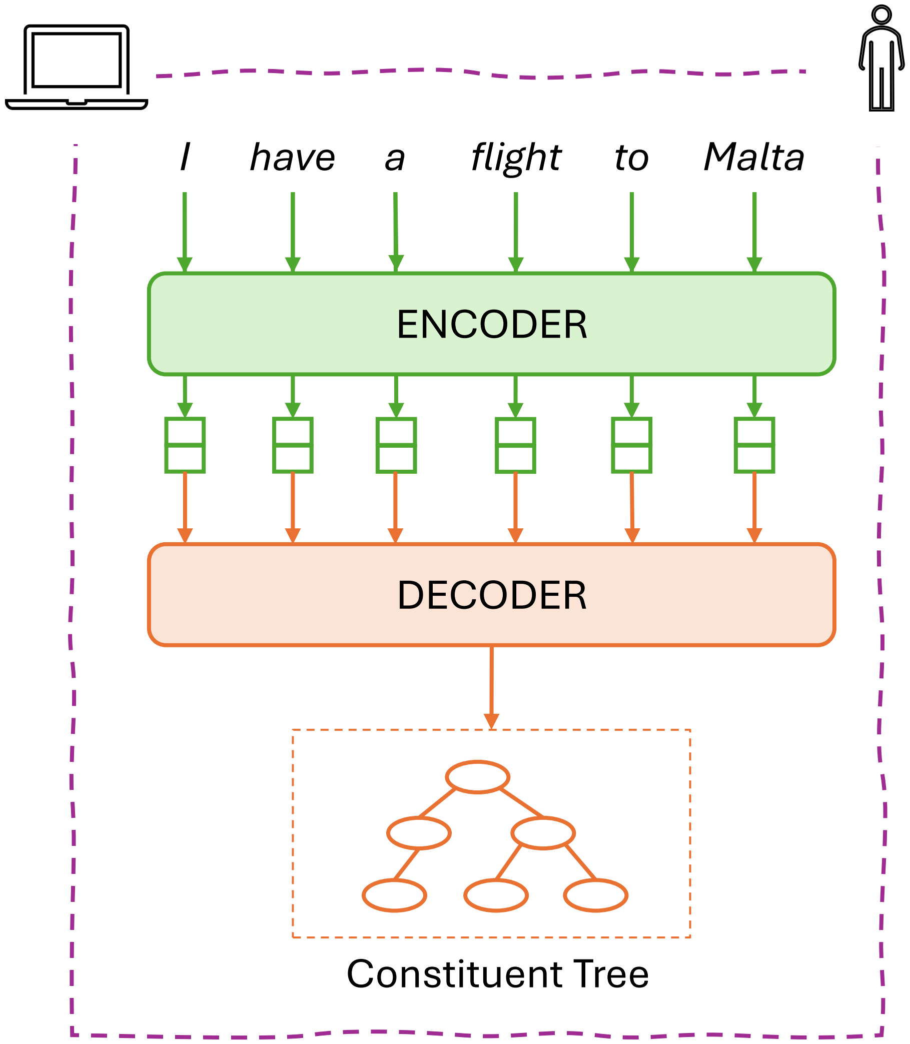
- Bidirectional encoder:
  - BERT & ELMo.
- Non-incremental decoder.
  - Kitaev & Klein (2018).



Incremental System

- Unidirectional encoder:
  - GPT & LSTMs.
- Incremental decoder.
  - Transition-based.
  - Sequence labeling.

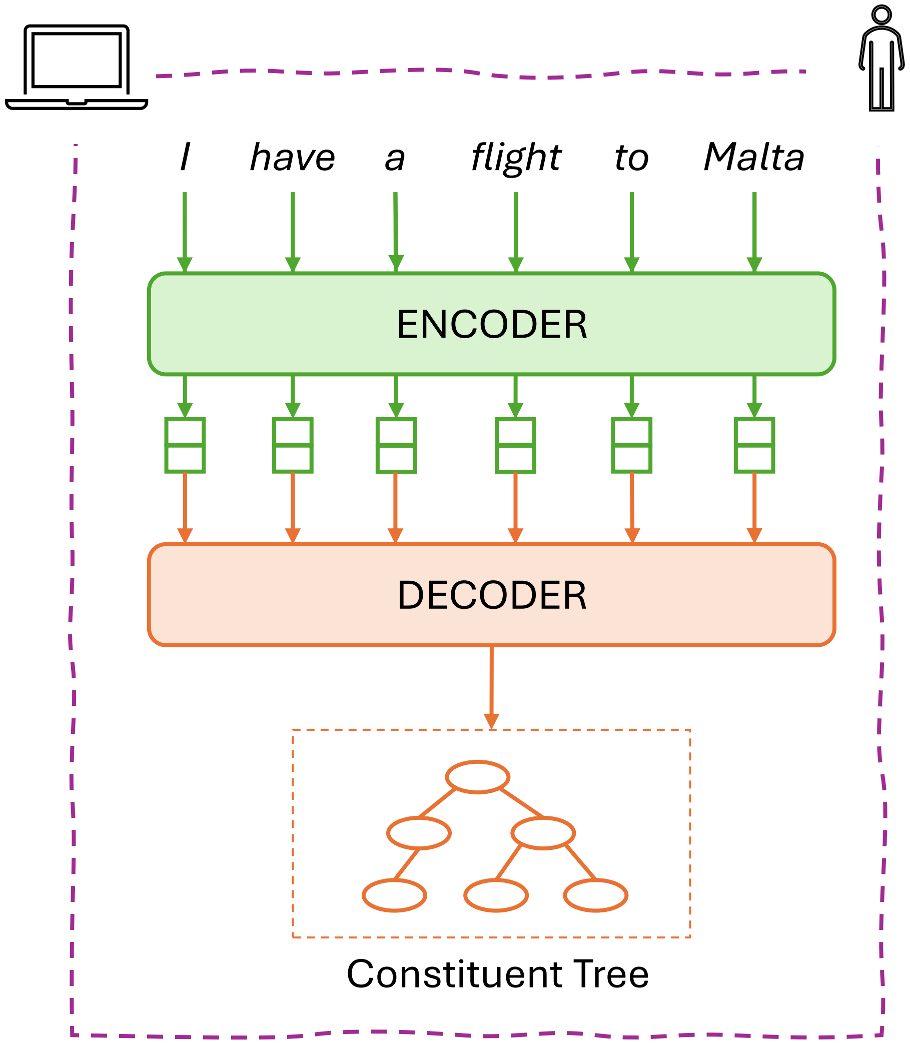
From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

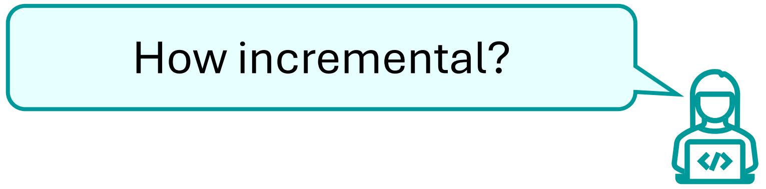
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

From Partial to Strictly Incremental *Constituent* Parsing

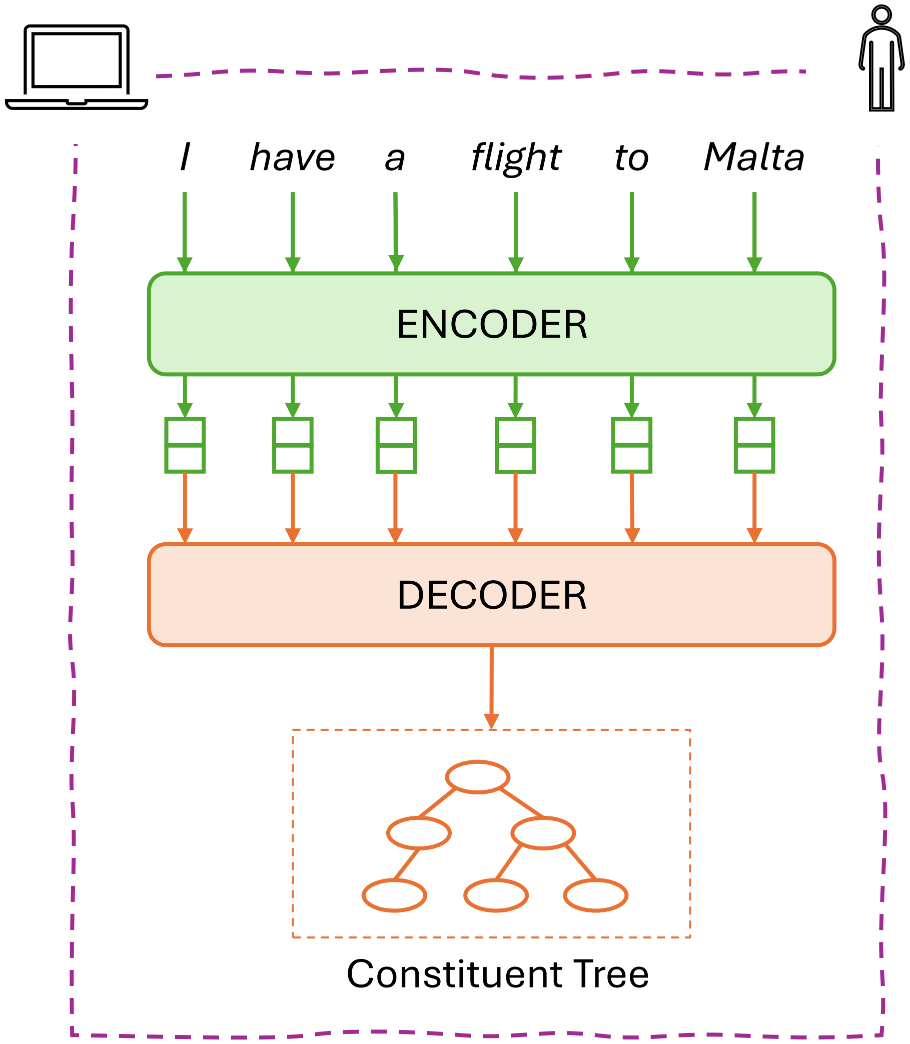


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**



From Partial to Strictly Incremental *Constituent* Parsing

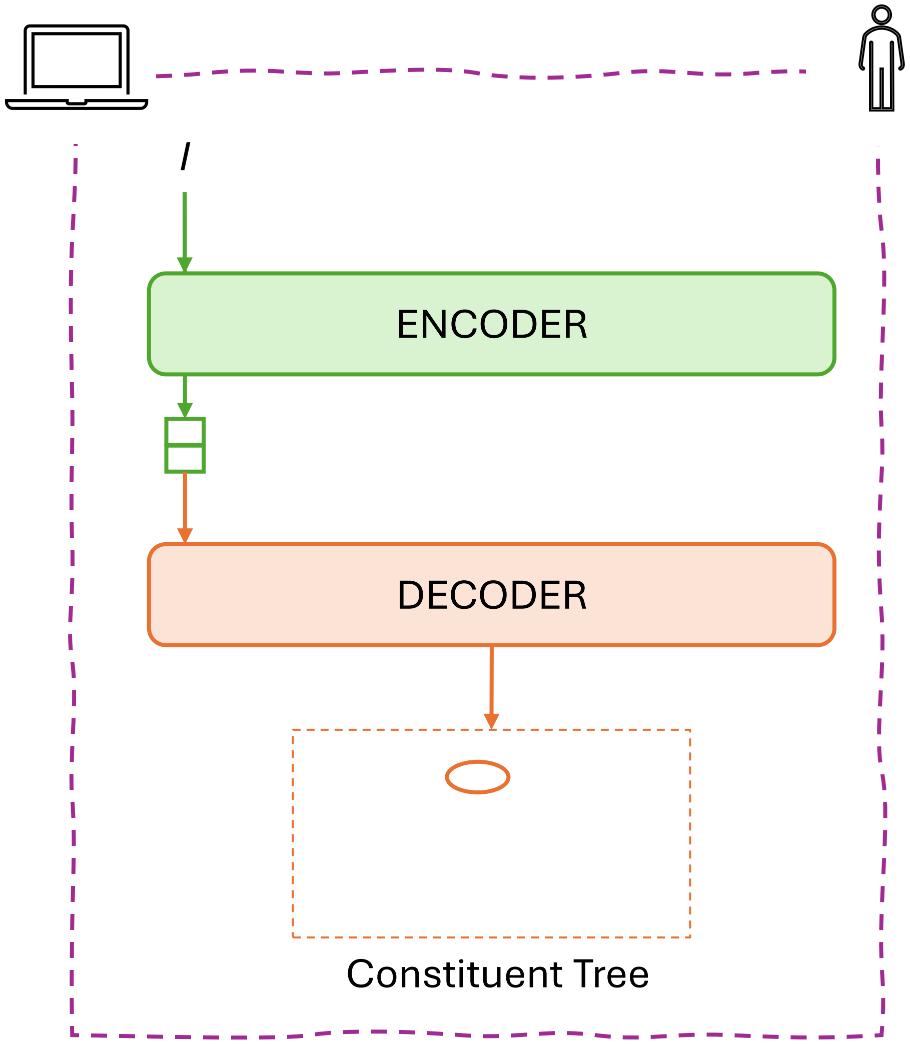


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

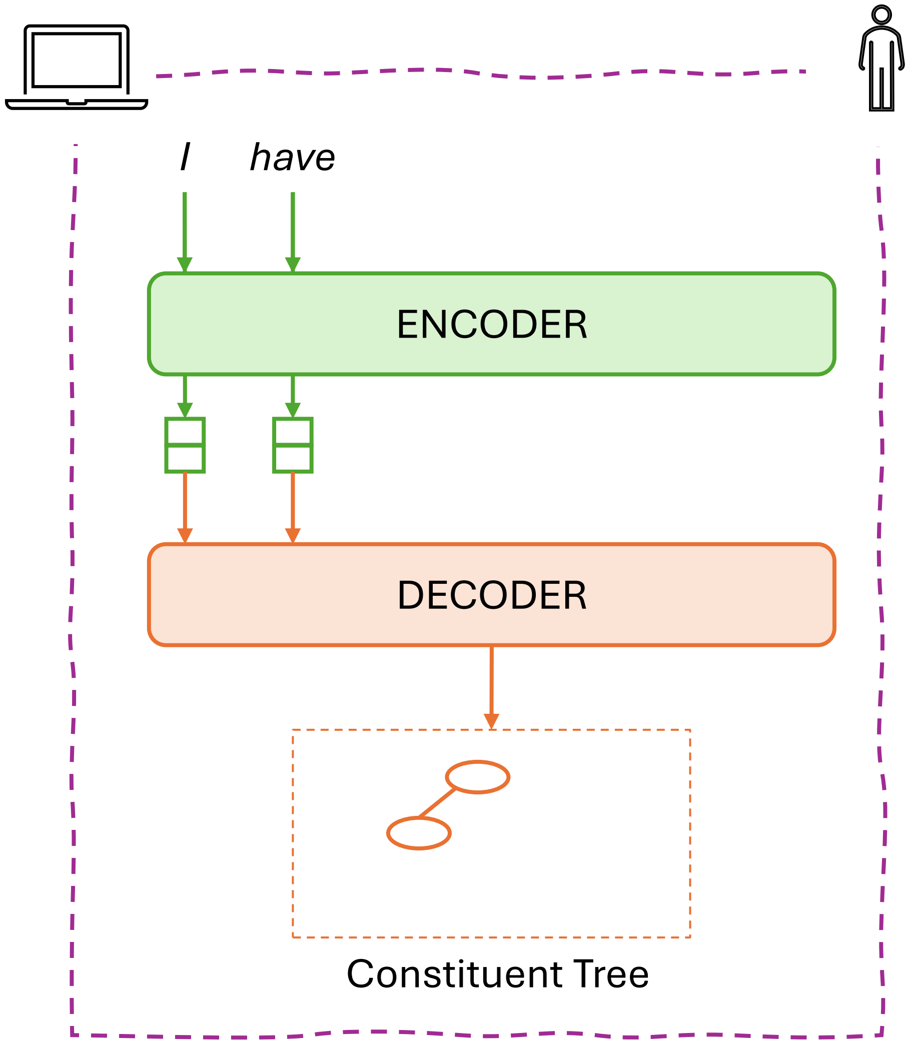


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

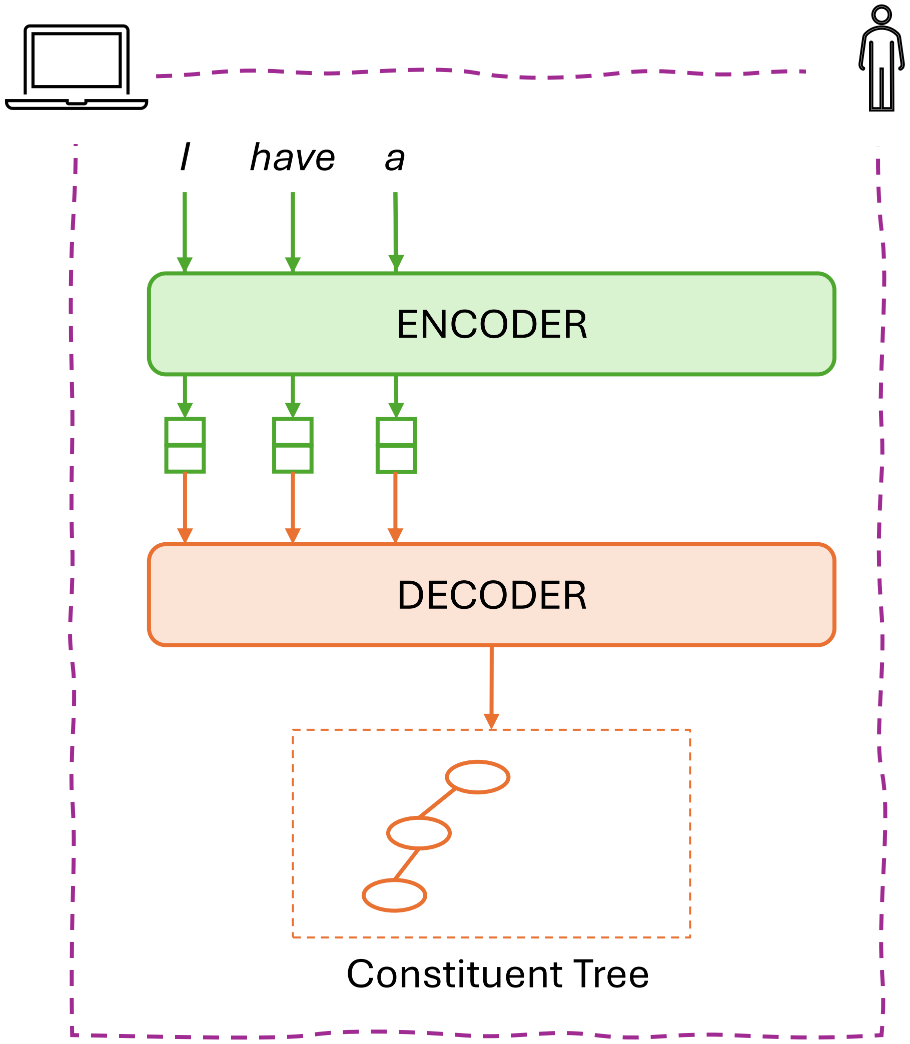


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

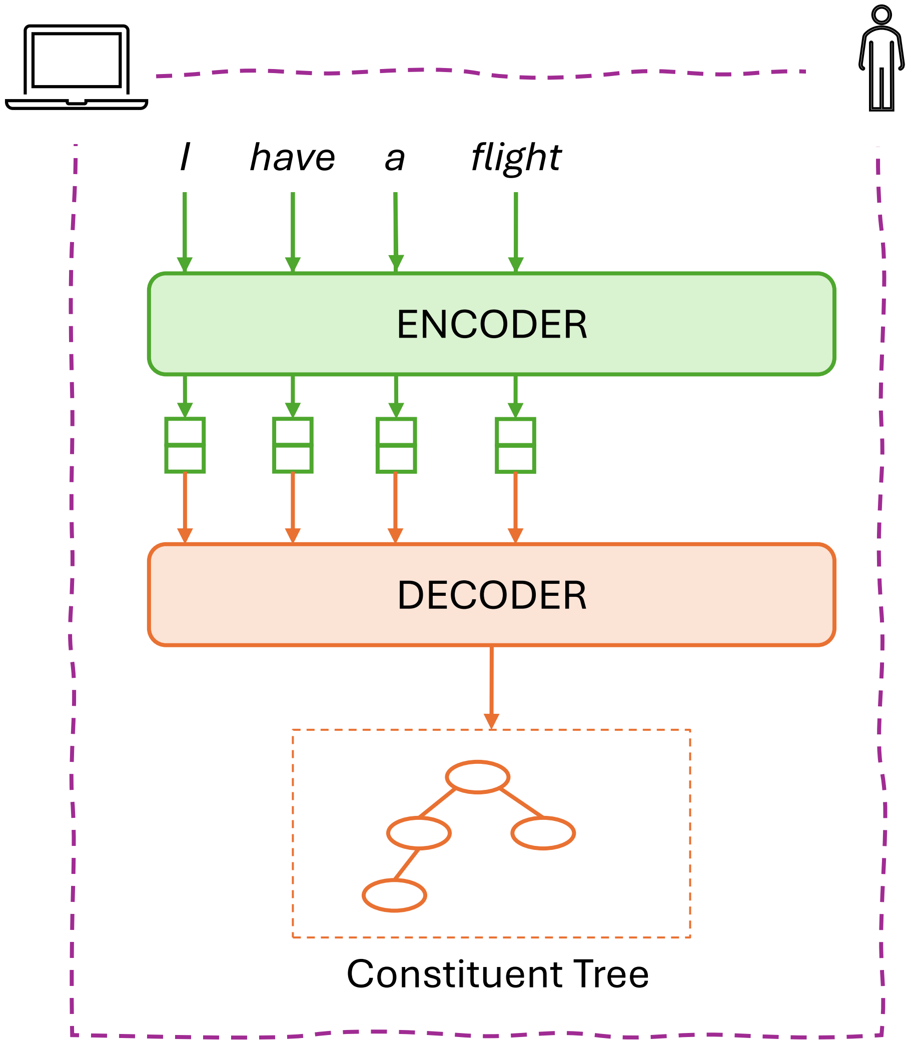


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

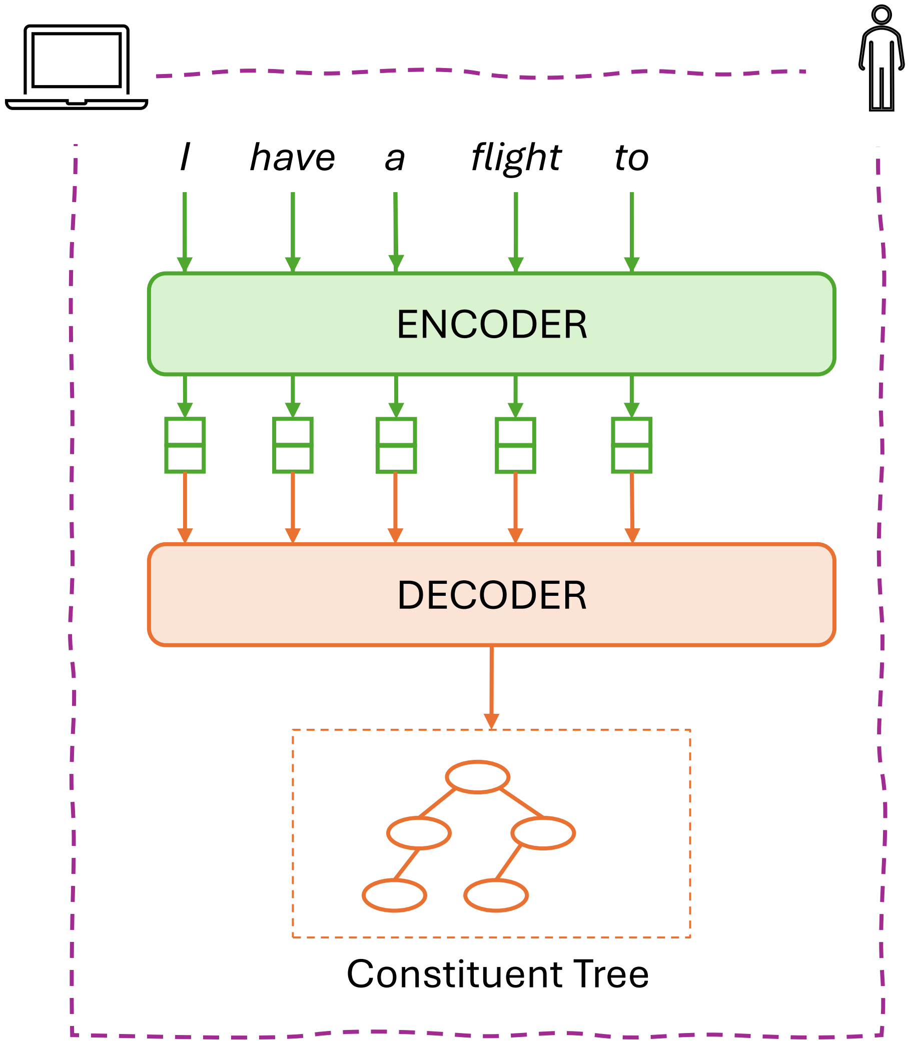


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

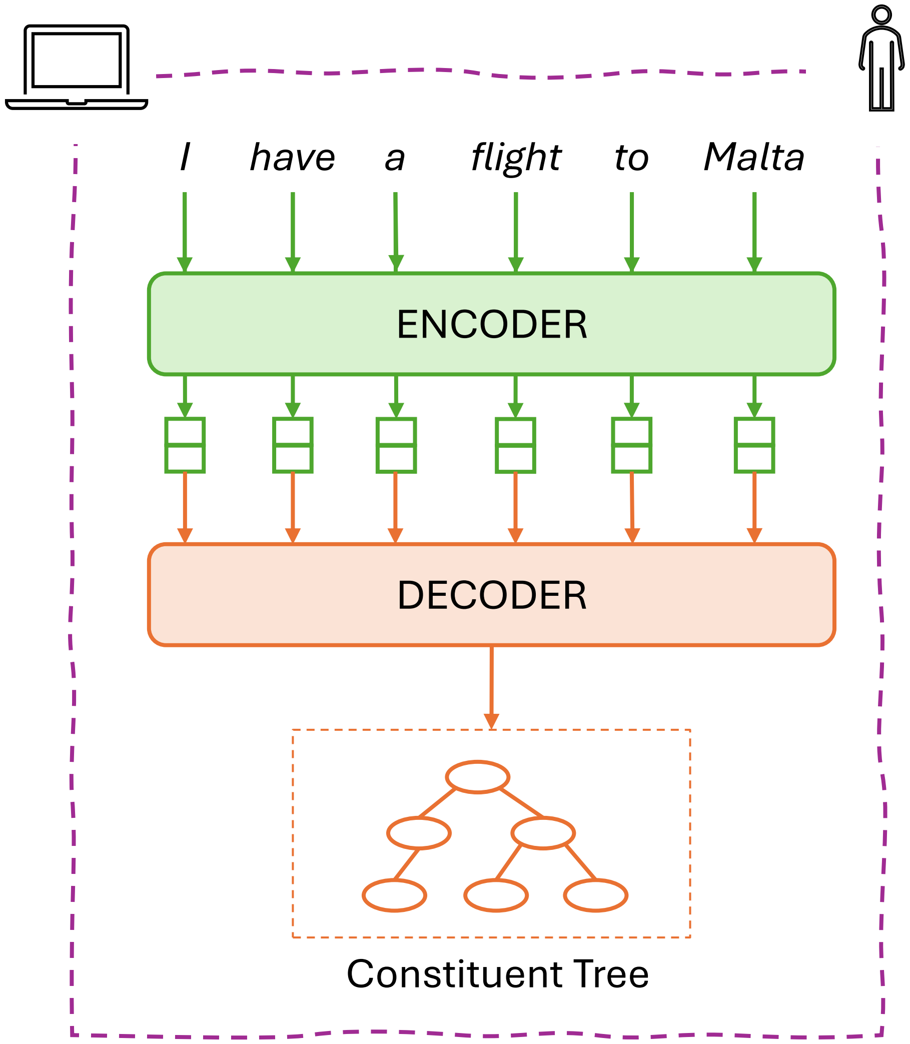


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing

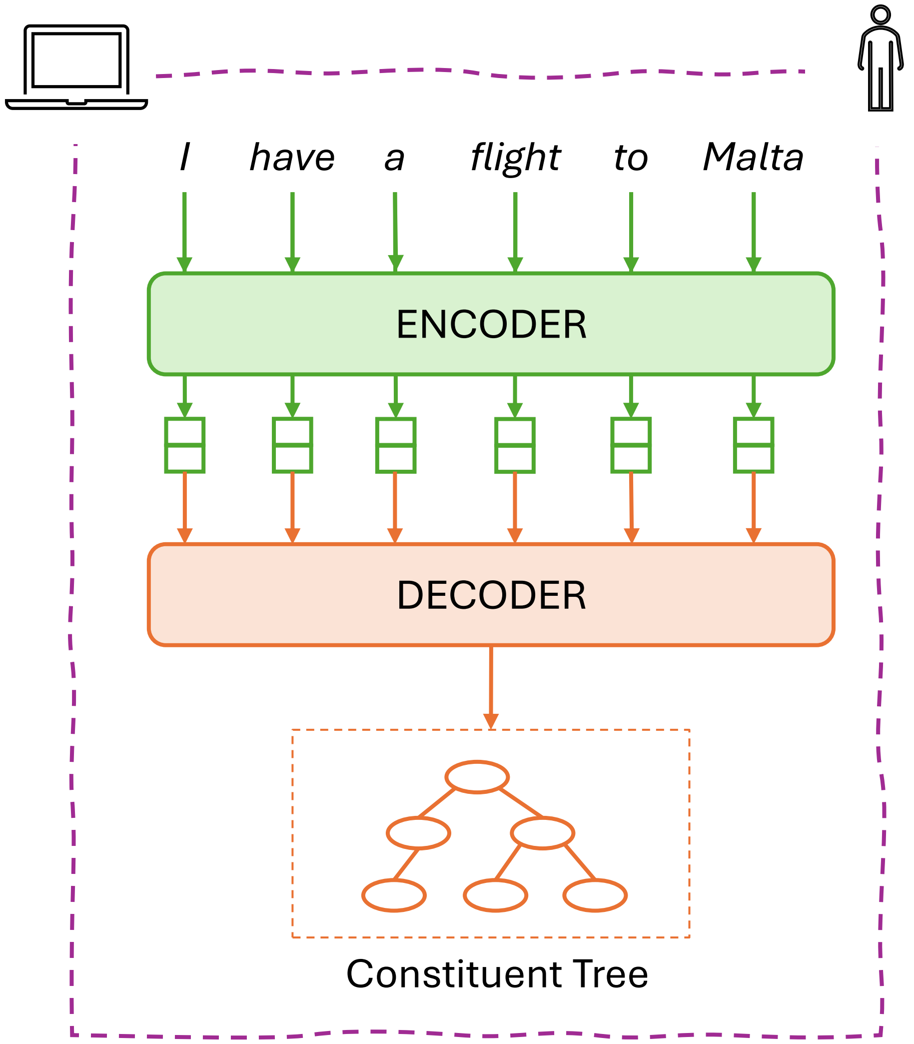


Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Each processed word w_i builds a partial tree from w_1 to w_i.

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

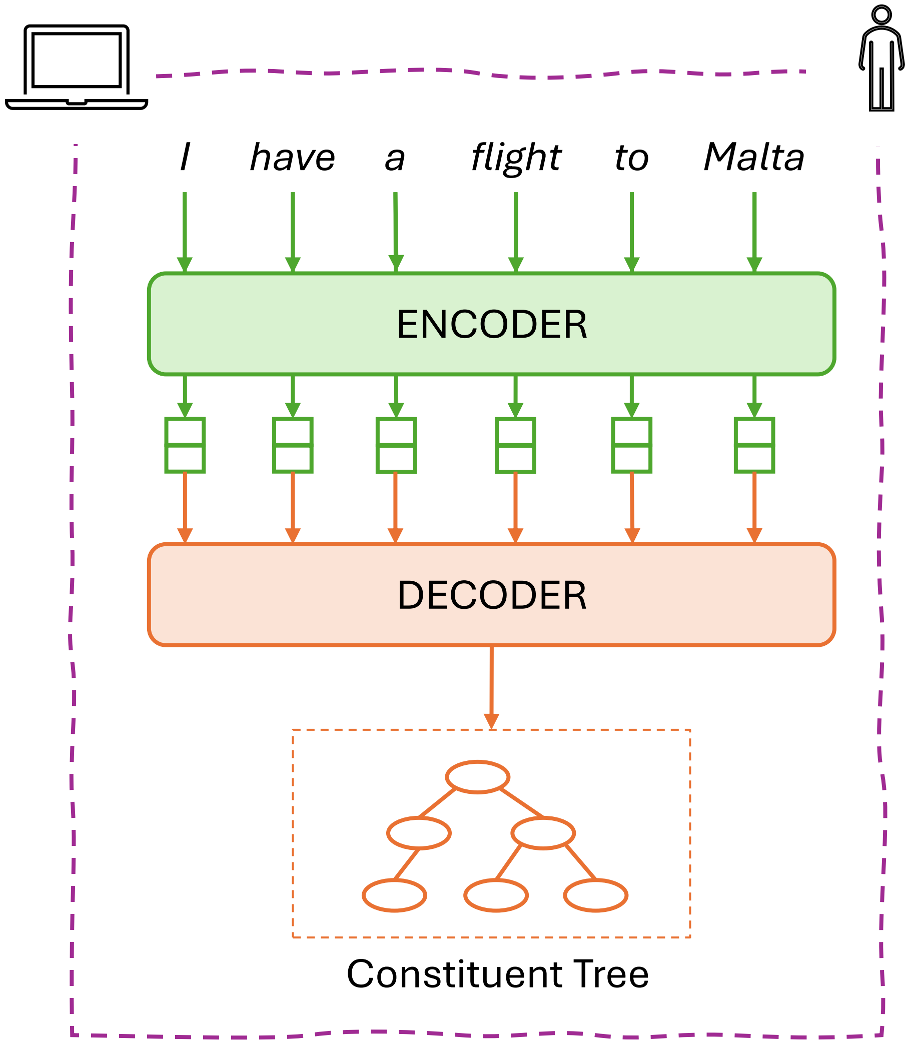
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k (by default, $k = 0$) .
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_{\mathrm{k}}(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

What happens if delay $k > 0$?

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

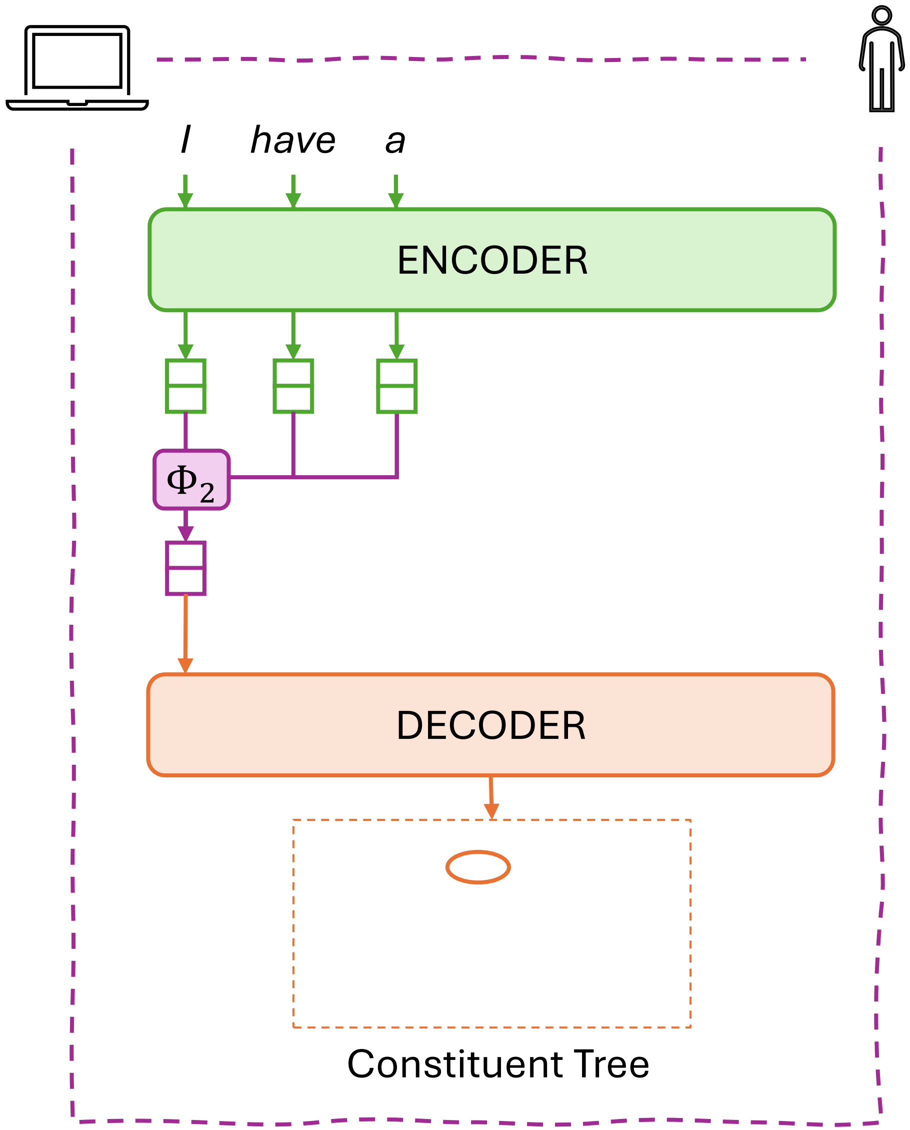
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k (by default, $k = 0$) .
- *Incrementally* encode each word w_i with $w_1, \dots, w_{i+k}$.
- In practice: $\Phi_\text{k}(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

Use $w_1, \dots, w_{i+k}$ to build a tree from w_1 to w_i!

From Partial to Strictly Incremental *Constituent* Parsing

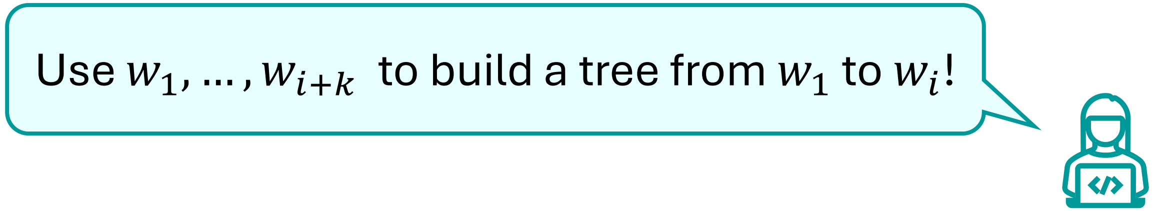


Incremental decoder

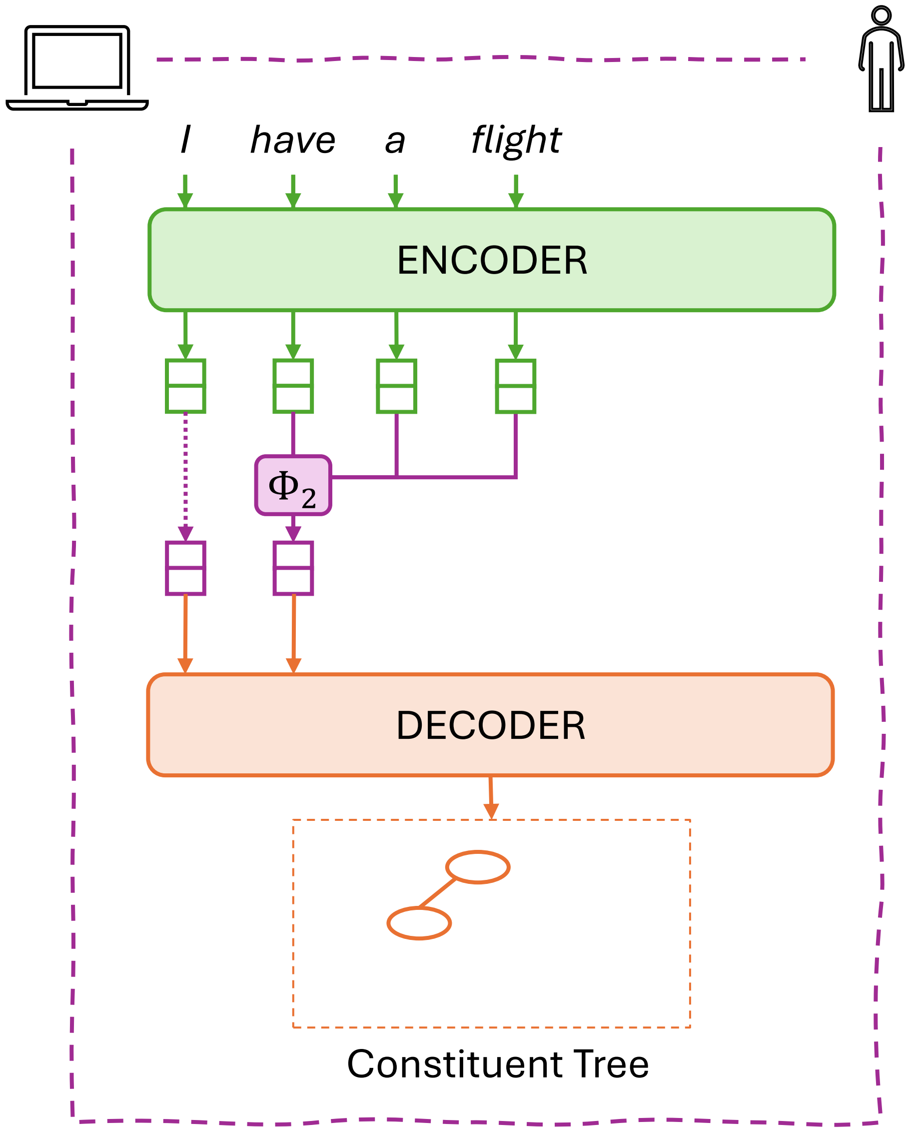
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k (by default, $k = 0$) .
- *Incrementally* encode each word w_i with $w_1, \dots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.



From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

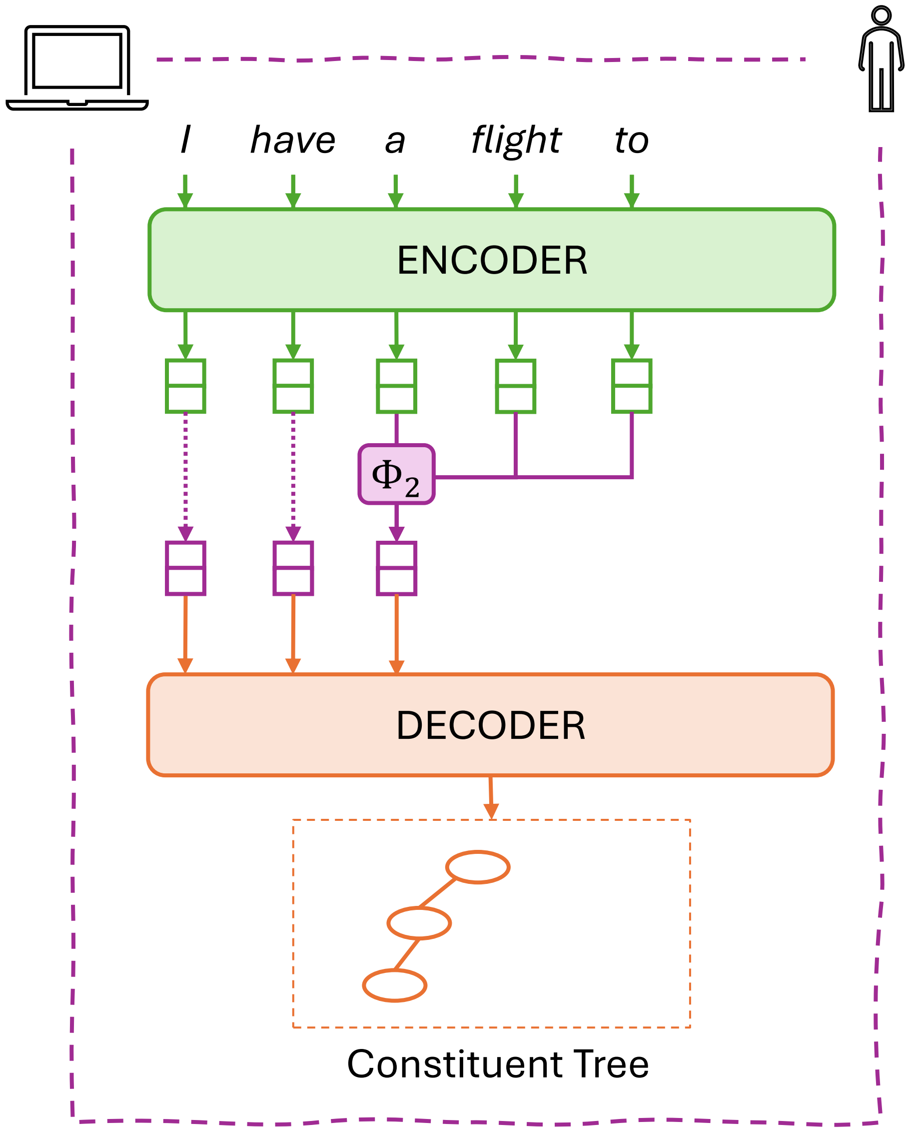
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k.
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

Use $w_1, \ldots, w_{i+k}$ to build a tree from w_1 to w_i!

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

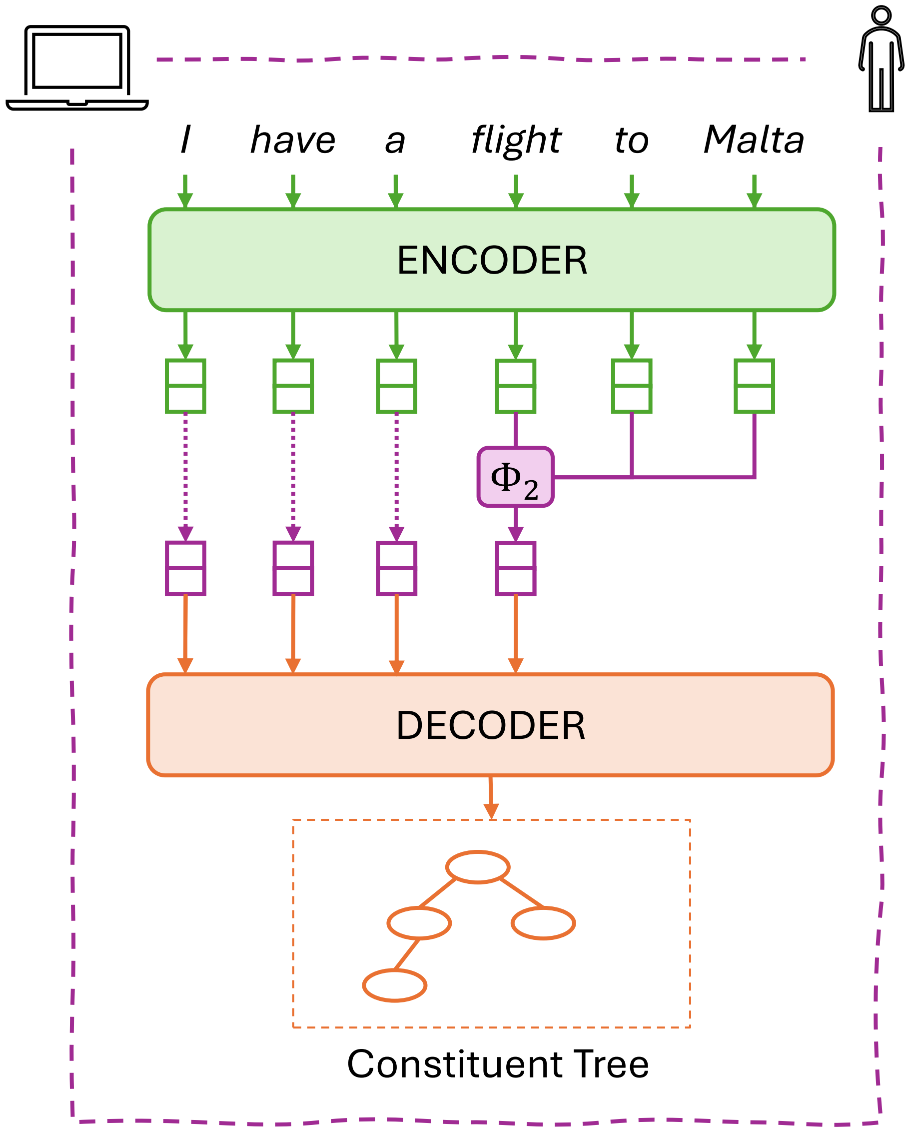
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k.
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

Use $w_1, \ldots, w_{i+k}$ to build a tree from w_1 to w_i!

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

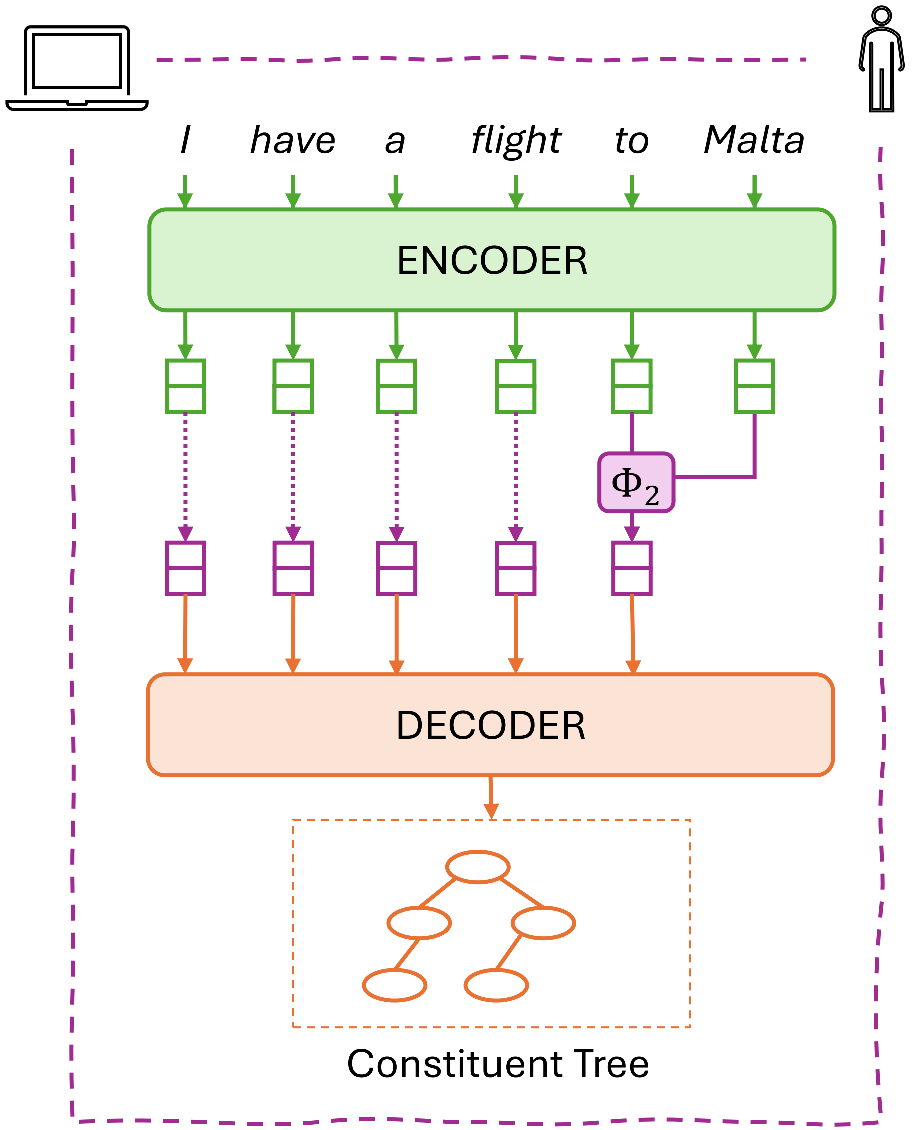
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k.
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

Use $w_1, \ldots, w_{i+k}$ to build a tree from w_1 to w_i!

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

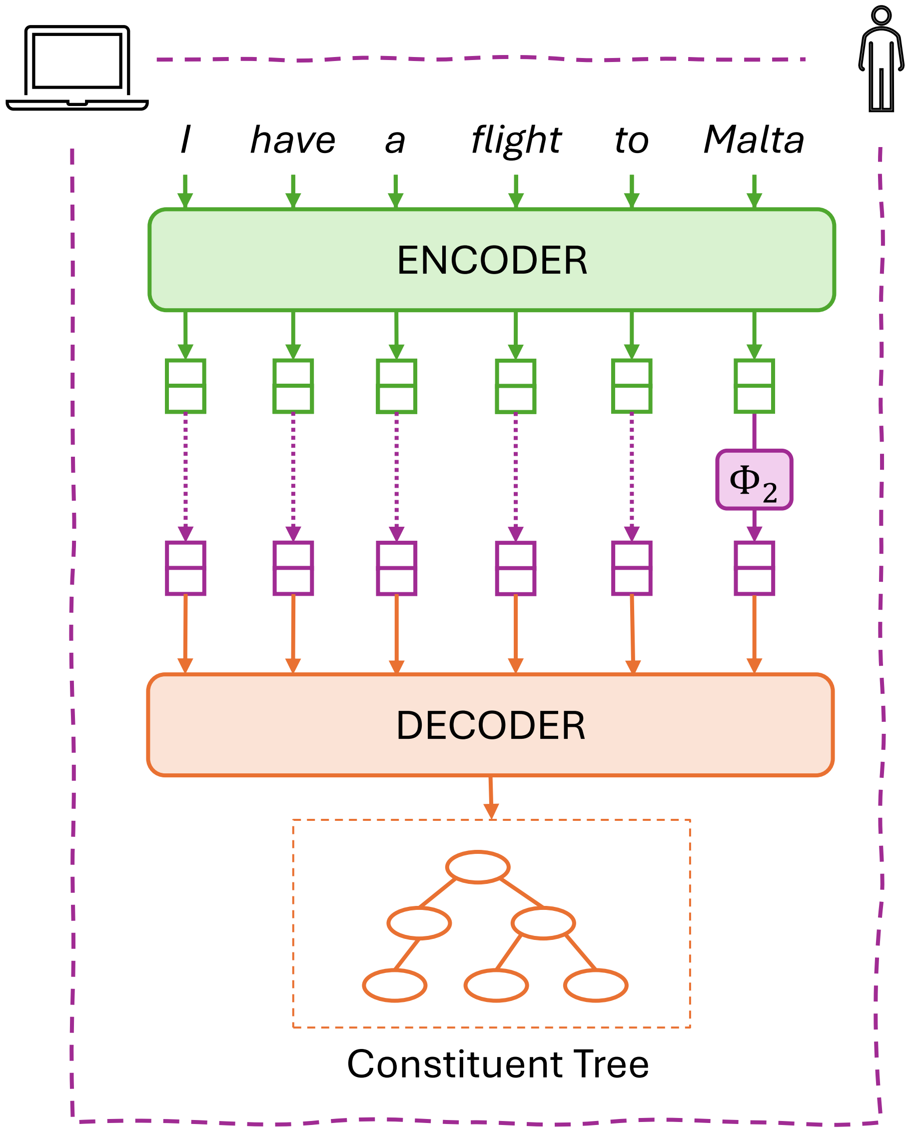
- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k.
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

Use $w_1, \ldots, w_{i+k}$ to build a tree from w_1 to w_i!

From Partial to Strictly Incremental *Constituent* Parsing



Incremental decoder

- **Attach-Juxtapose** from **Yang & Deng (2020).**
- **Sequence Labeling** from **Gómez-Rodríguez & Vilares (2018).**

Delayed incremental processing

- Parameter k.
- *Incrementally* encode each word w_i with $w_1, \ldots, w_{i+k}$.
- In practice: $\Phi_k(\mathbf{h}_i \cdots \mathbf{h}_{i+k})$ where Φ is a feed-forward network.

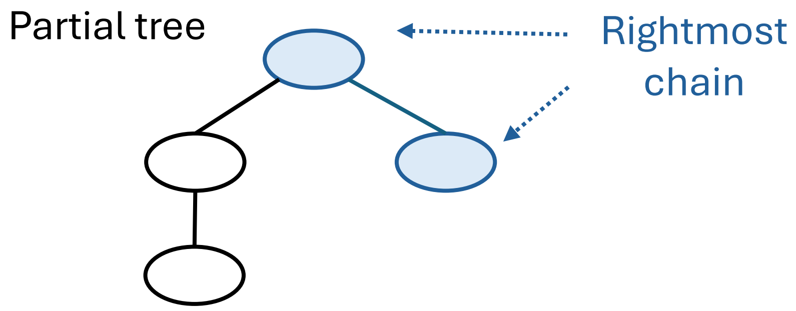
Use $w_1, \ldots, w_{i+k}$ to build a tree from w_1 to w_i!

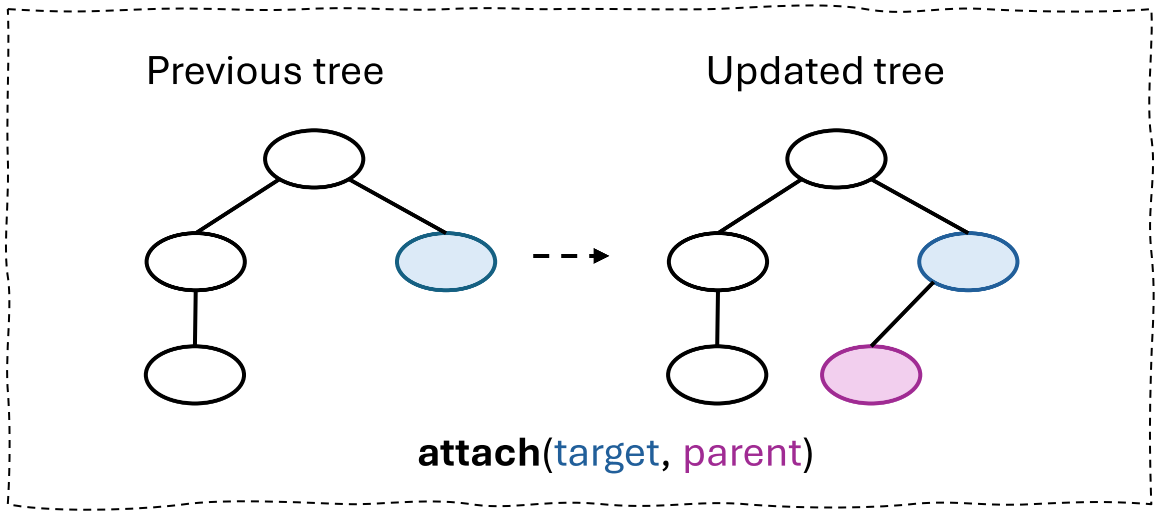
Attach-Juxtapose (Yang & Deng, 2020)

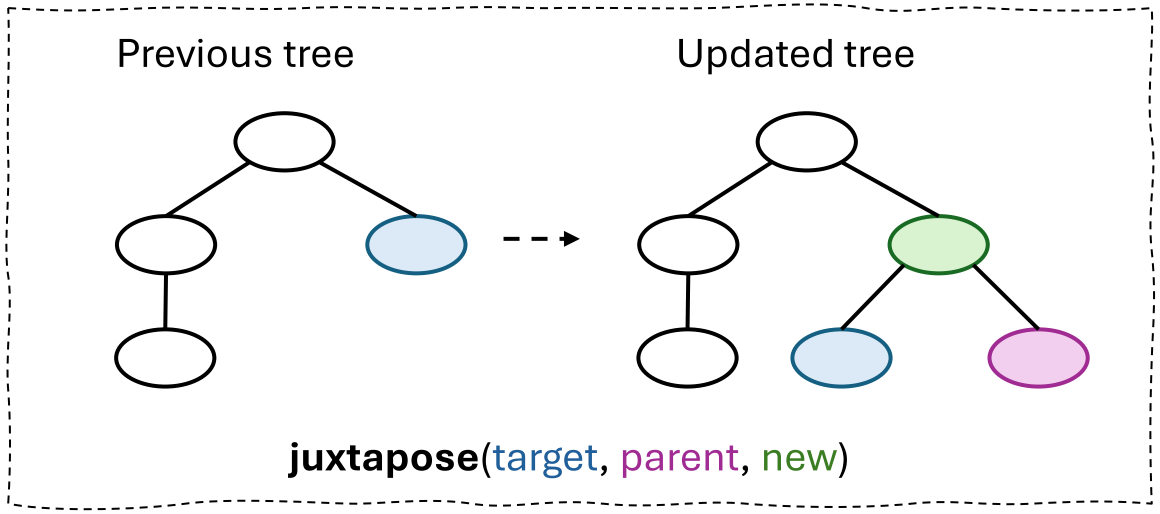
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.





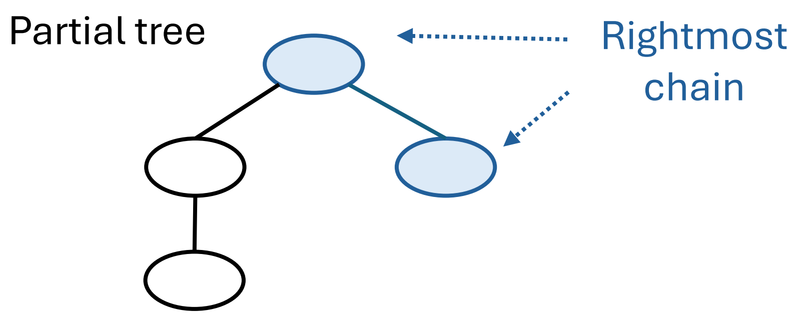


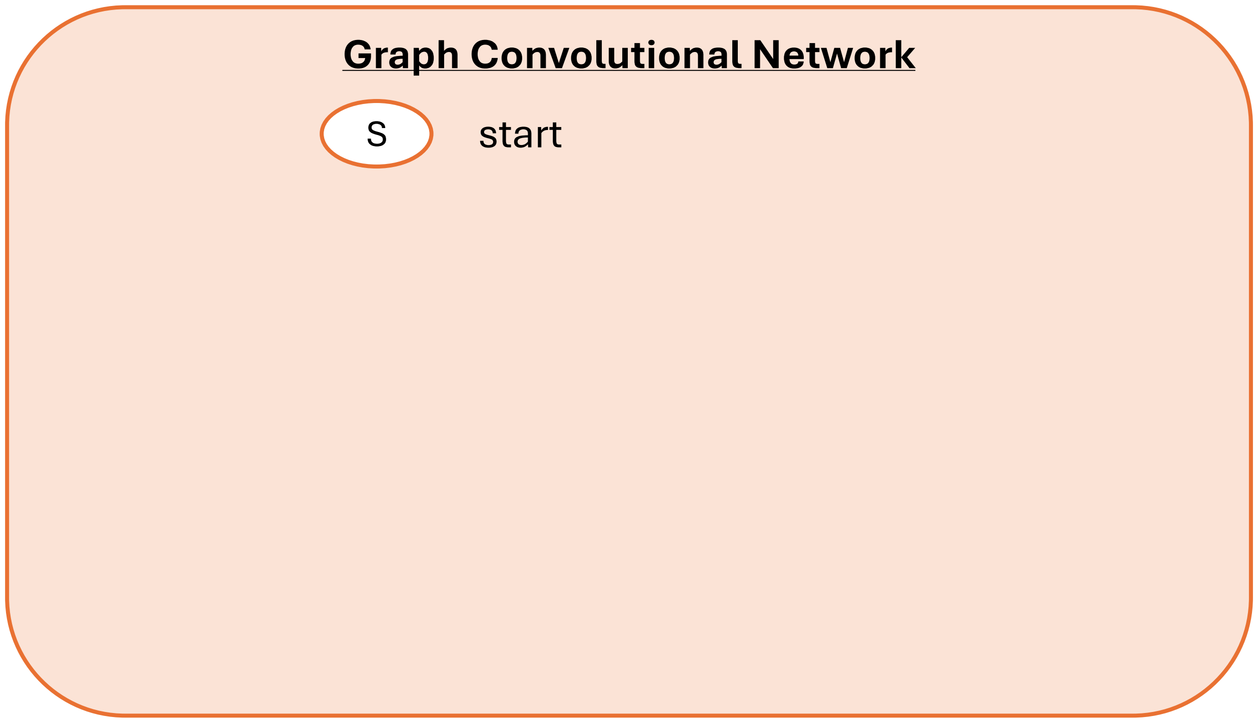
Attach-Juxtapose (Yang & Deng, 2020)

- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \ldots, w_n \rightarrow t_1, \ldots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



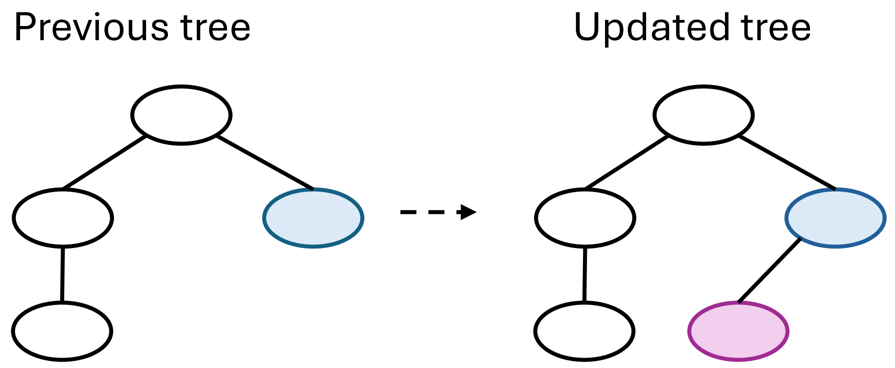


Attach-Juxtapose (Yang & Deng, 2020)

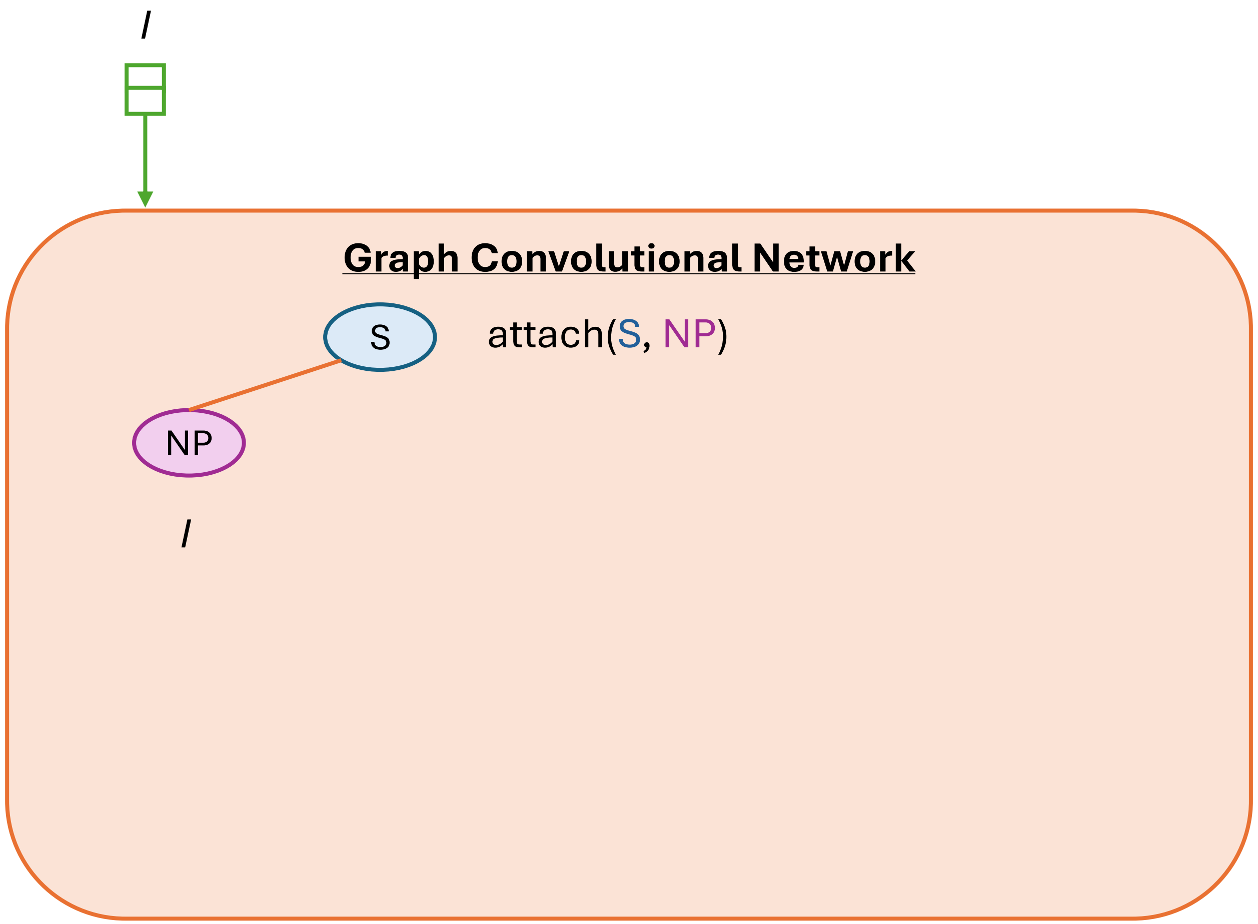
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \ldots, w_n \rightarrow t_1, \ldots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



attach(target, parent)

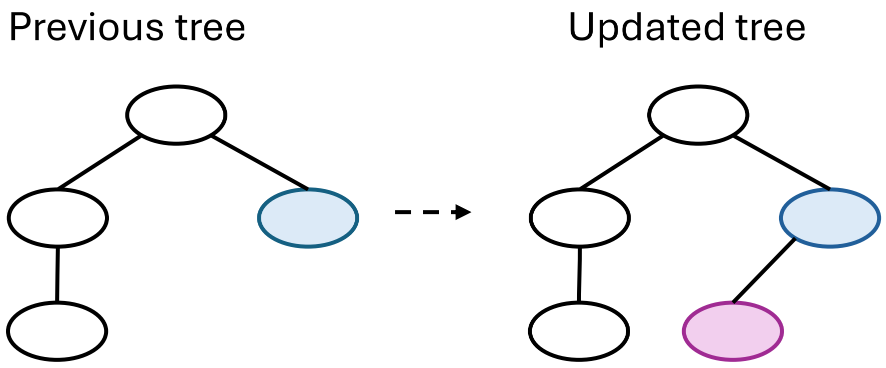


Attach-Juxtapose (Yang & Deng, 2020)

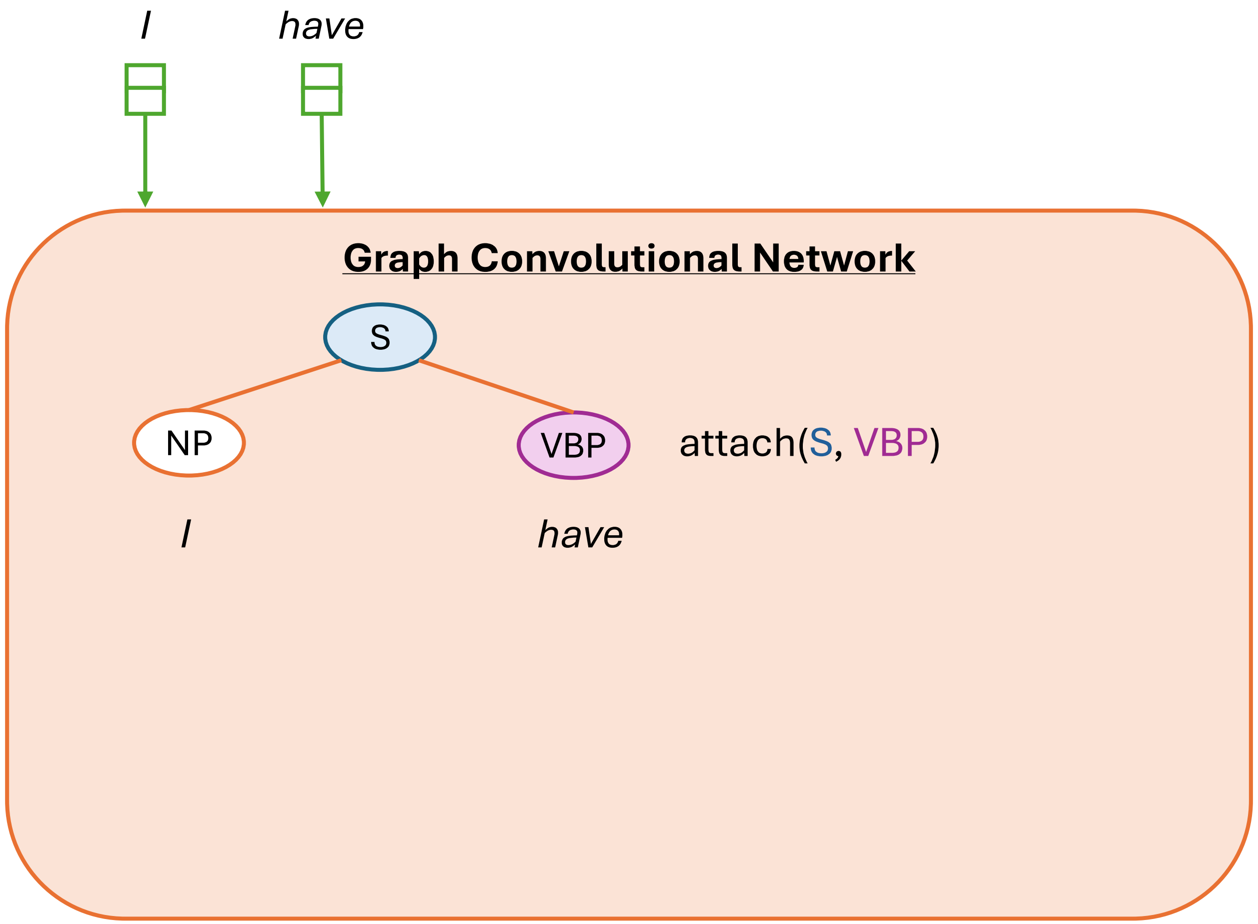
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



attach(target, parent)

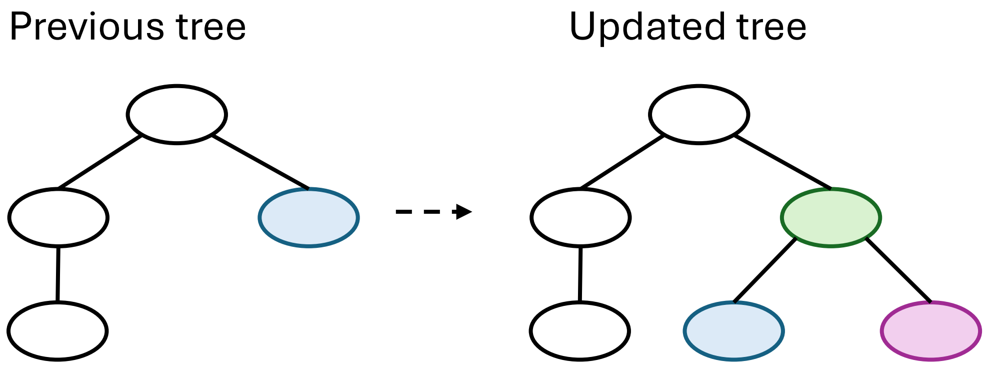


Attach-Juxtapose (Yang & Deng, 2020)

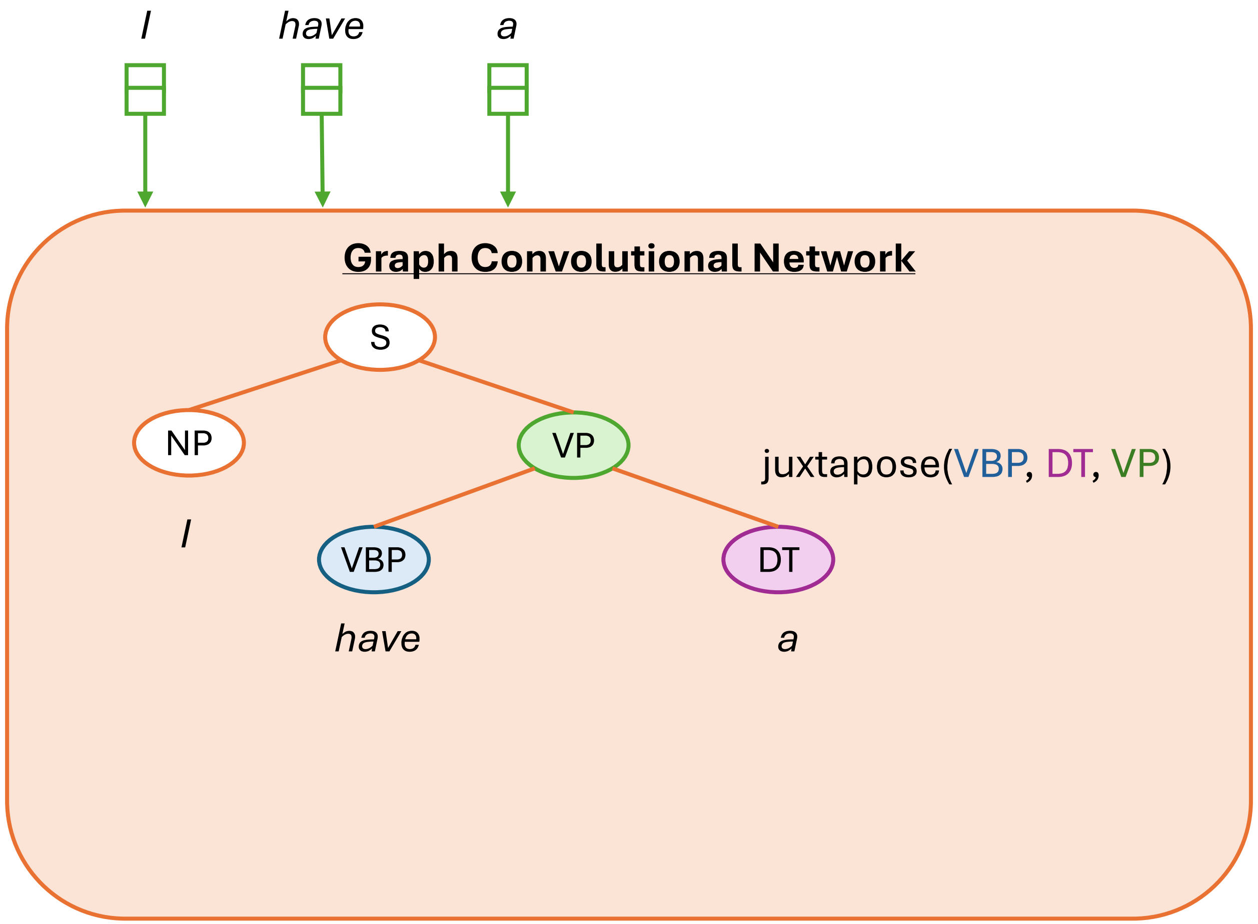
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



juxtapose(target, parent, new)



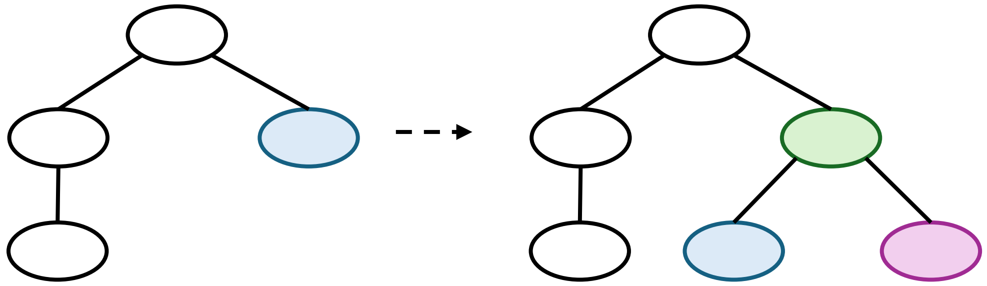
Attach-Juxtapose (Yang & Deng, 2020)

- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

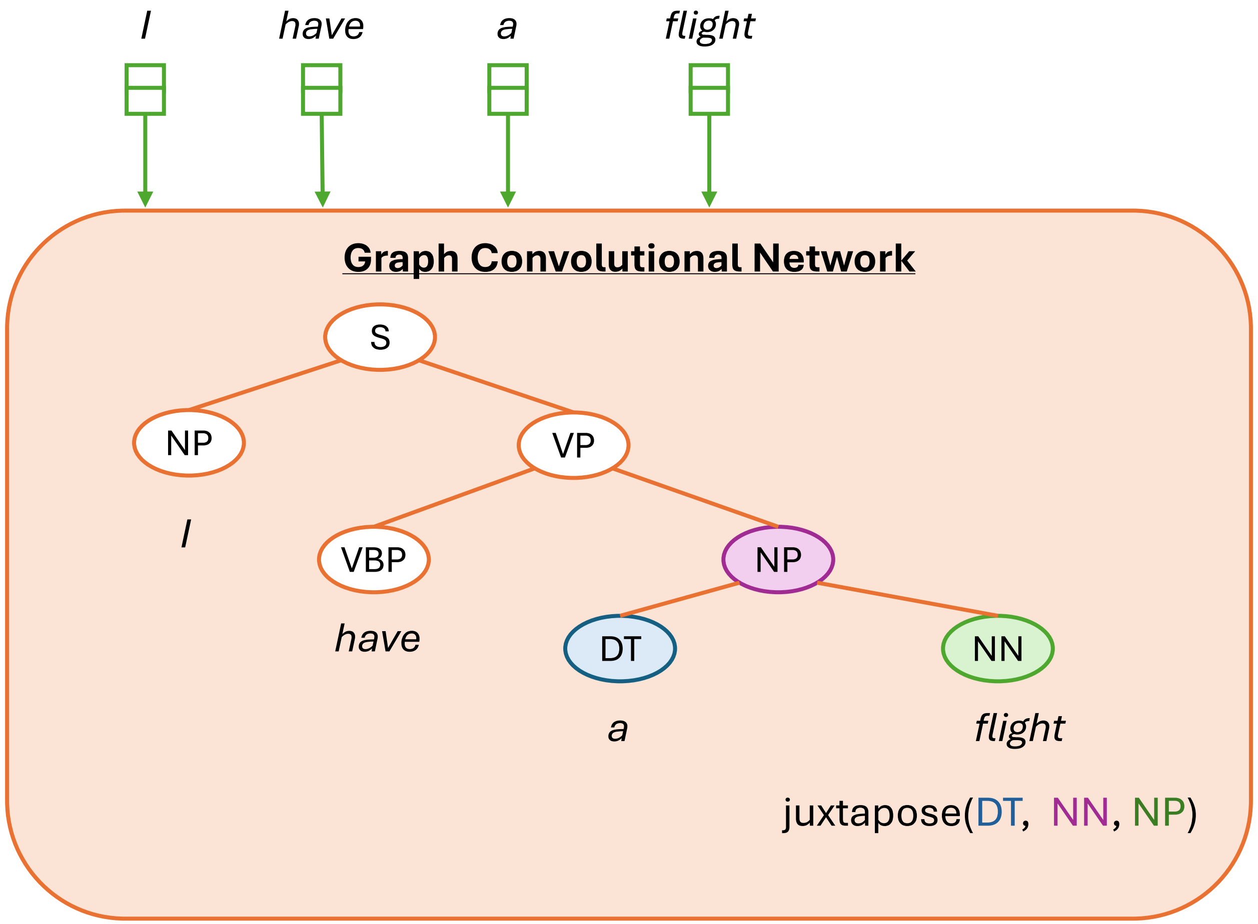
$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.

Previous tree   Updated tree



juxtapose(target, parent, new)

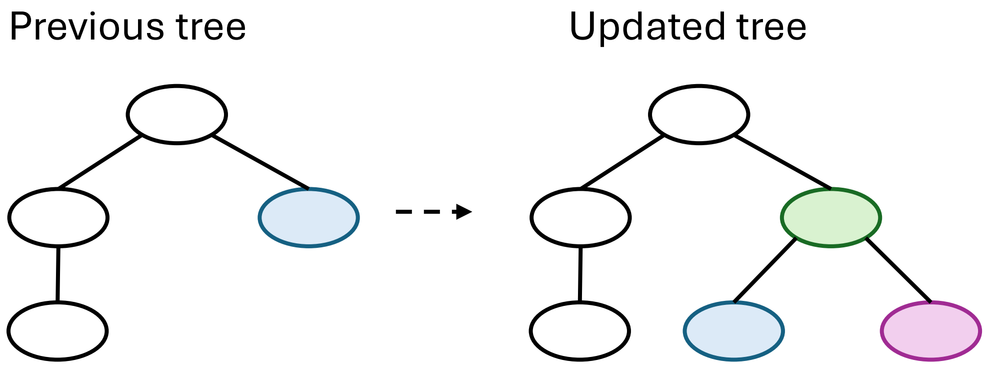


Attach-Juxtapose (Yang & Deng, 2020)

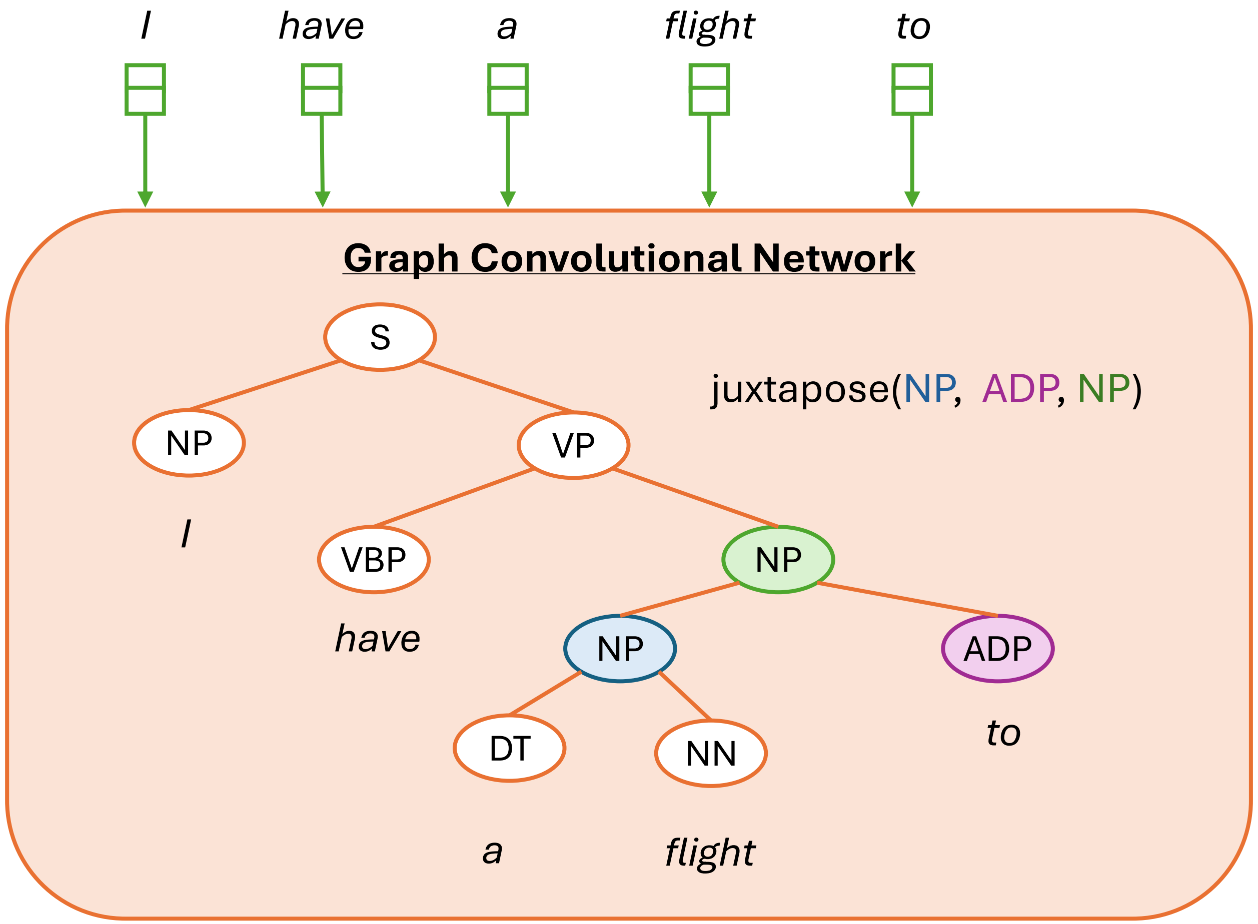
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



juxtapose(target, parent, new)



Attach-Juxtapose (Yang & Deng, 2020)

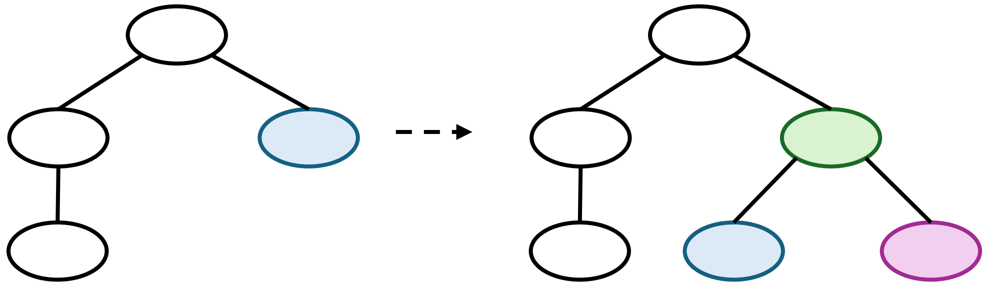
- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \to t_1, \dots, t_n$$

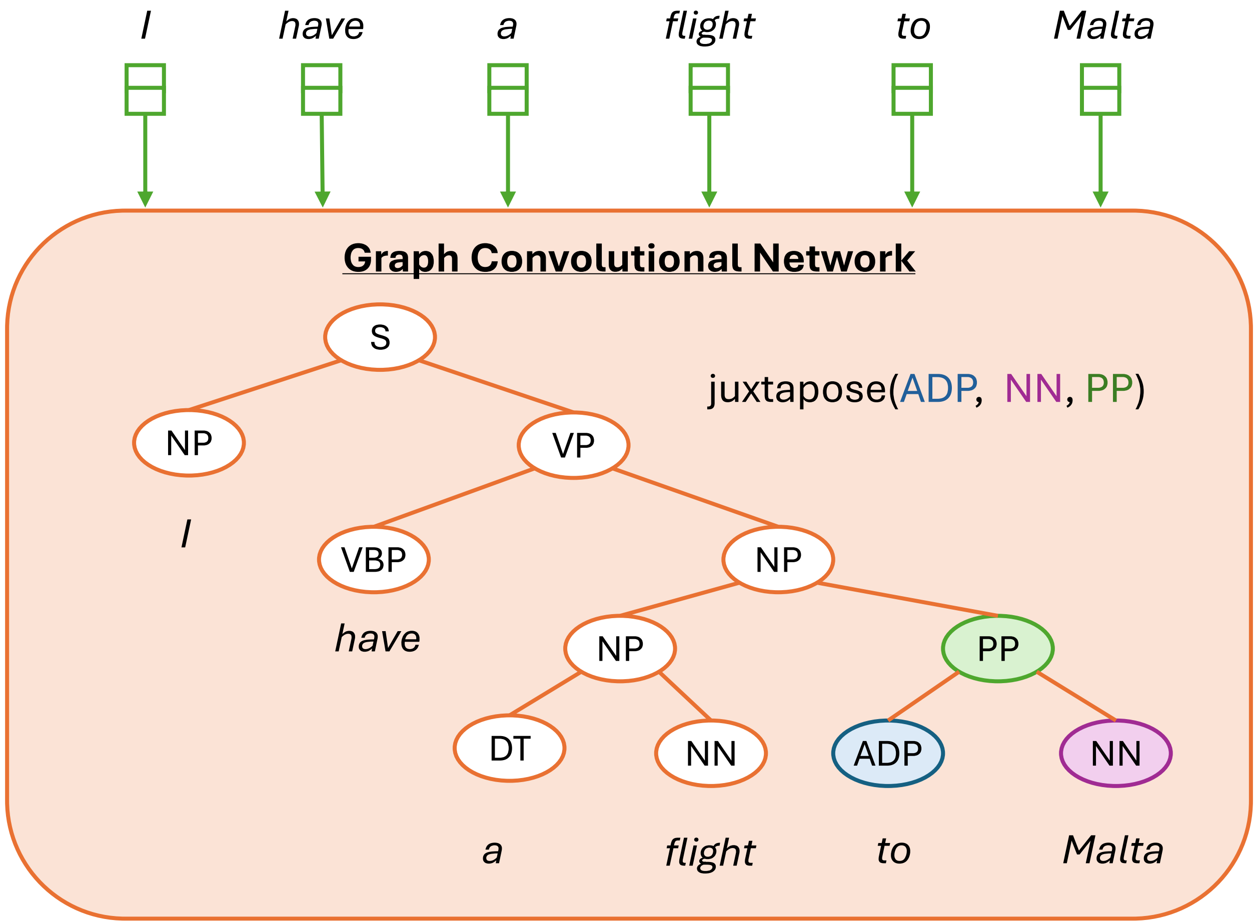
- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.

Previous tree

Updated tree



juxtapose(target, parent, new)

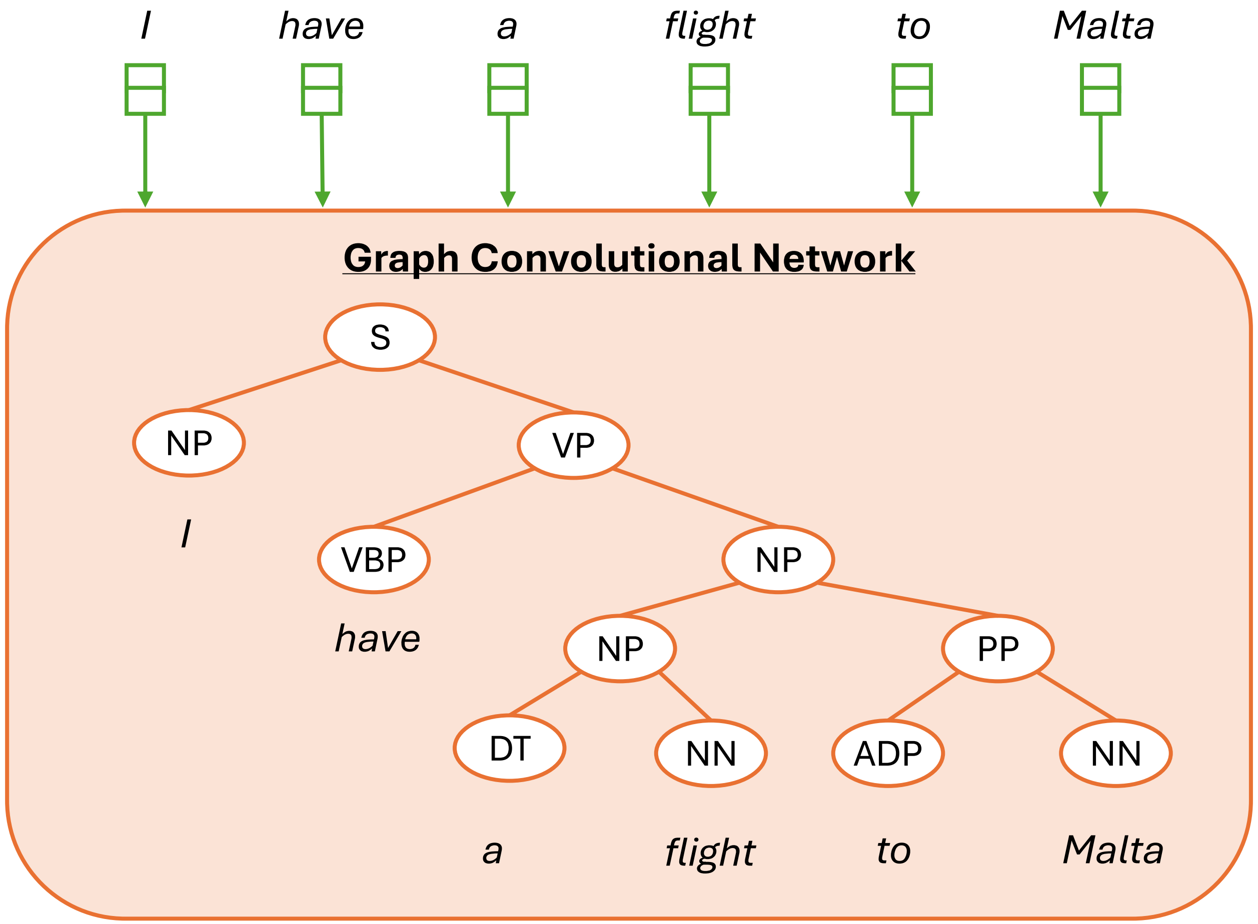


Attach-Juxtapose (Yang & Deng, 2020)

- Transition-based system.
- Two actions: **attach** & **juxtapose**.
- Sentence of n words to n transitions.

$$w_1, \dots, w_n \rightarrow t_1, \dots, t_n$$

- Graph Convolutional Network (GCN).
- Append subtrees to the rightmost chain.



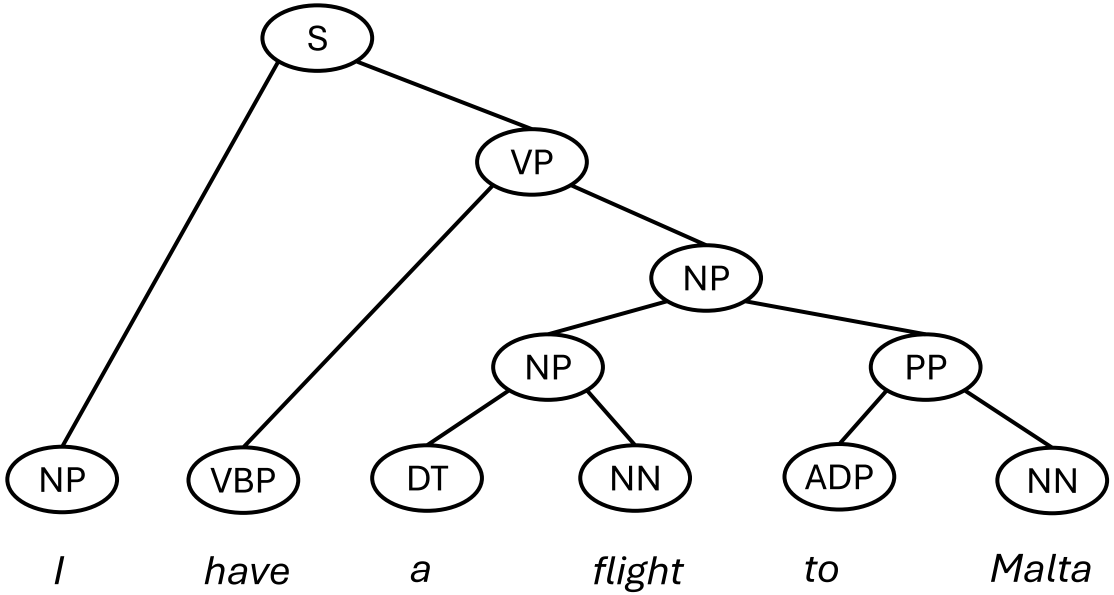
Full constituent tree!

Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \to \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
    - l_i: # common constituents of w_i and w_{i+1}.
    - Absolute: $d_i = l_i$.
    - Relative: $d_i = l_i - l_{i-1}$.
    - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).

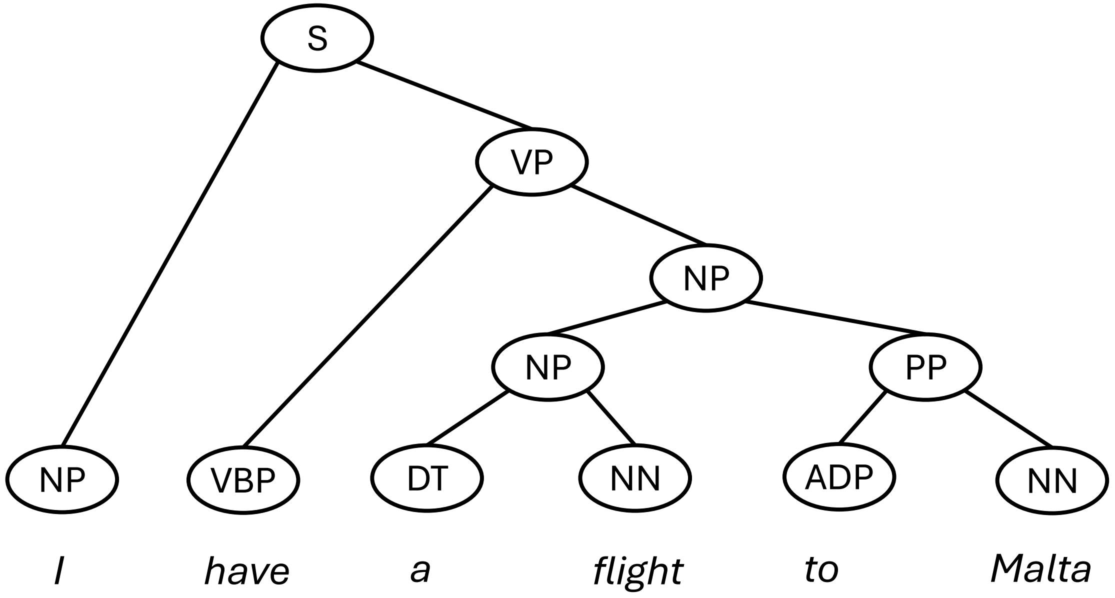


Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \to \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



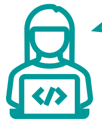
How to encode?

Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

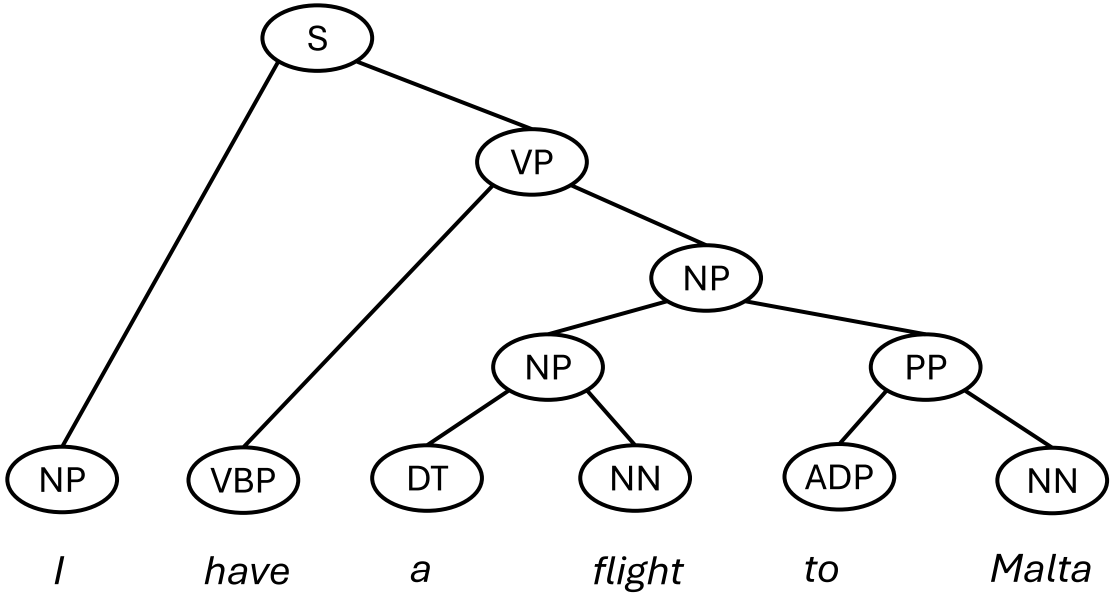
- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \ldots, w_n \rightarrow \ell_1, \ldots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
    - l_i: # common constituents of w_i and w_{i+1}.
    - Absolute: $d_i = l_i$.
    - Relative: $d_i = l_i - l_{i-1}$.
    - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



Incrementally obtain ℓ_i and append right nodes.

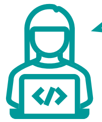


Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

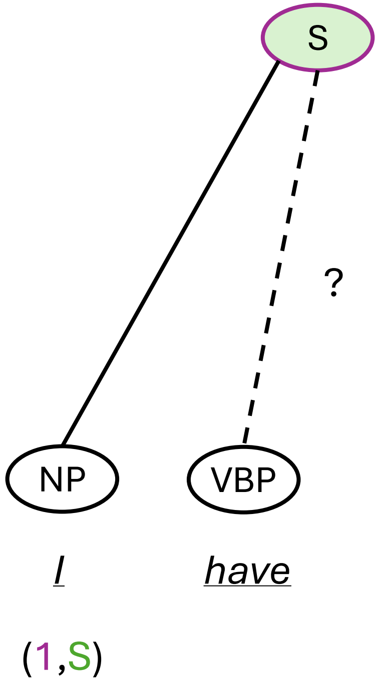
- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \ldots, w_n \to \ell_1, \ldots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



Incrementally obtain ℓ_i
and append right nodes.

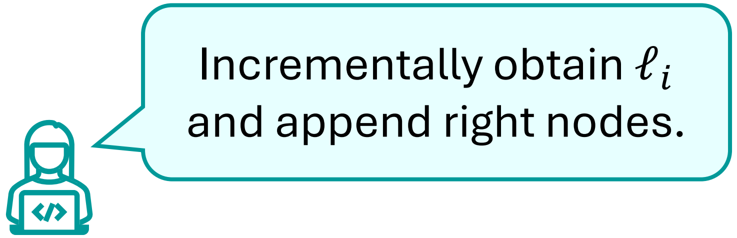


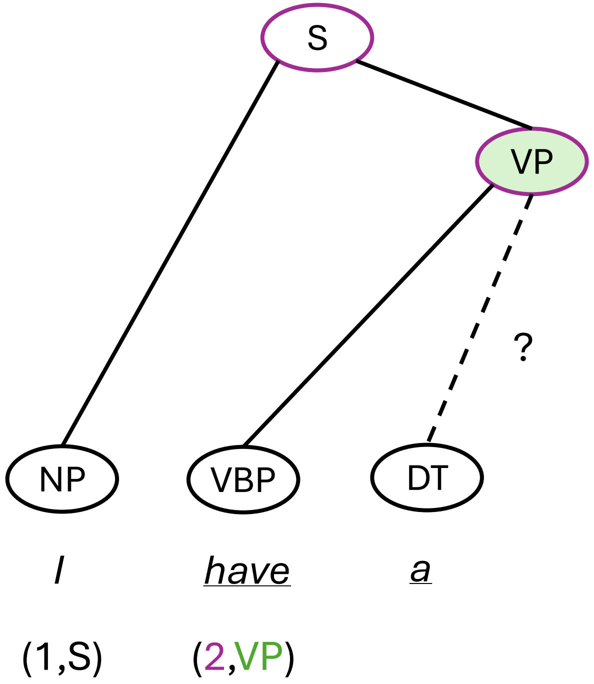
Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \to \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



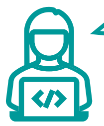


Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

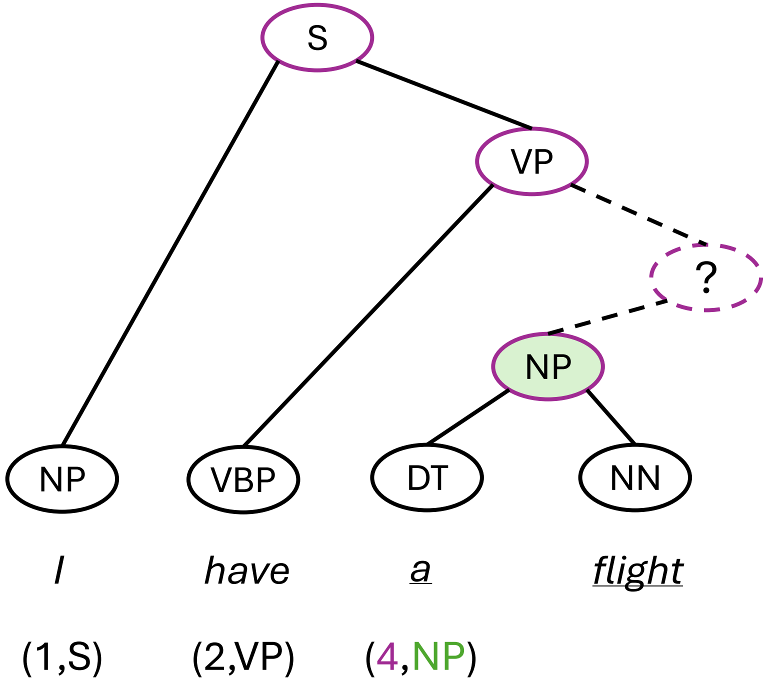
- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \ldots, w_n \rightarrow \ell_1, \ldots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



Incrementally obtain ℓ_i and append right nodes.

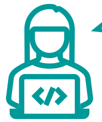


Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

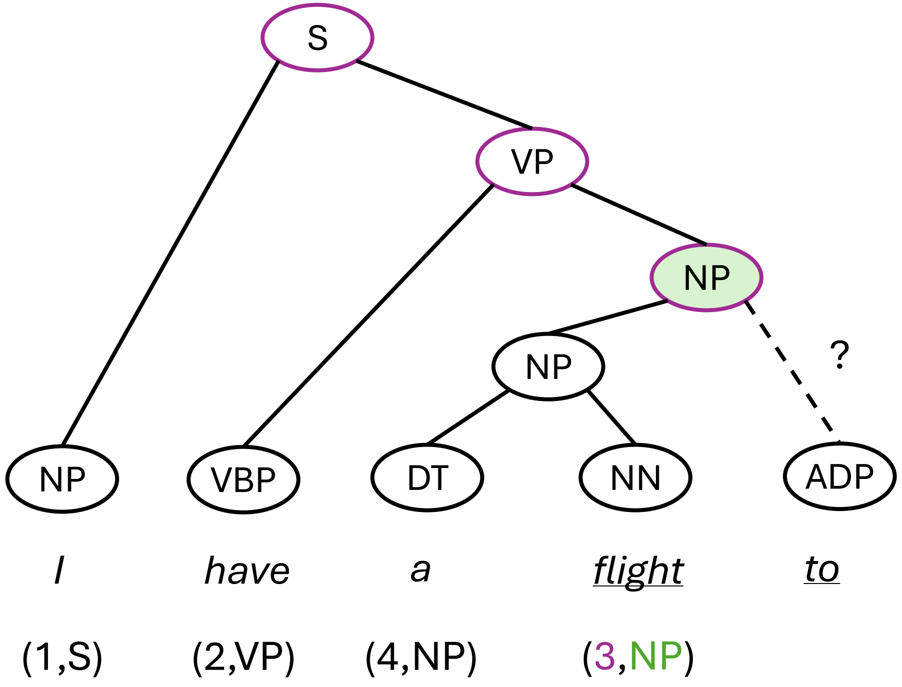
- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \rightarrow \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



Incrementally obtain ℓ_i and append right nodes.



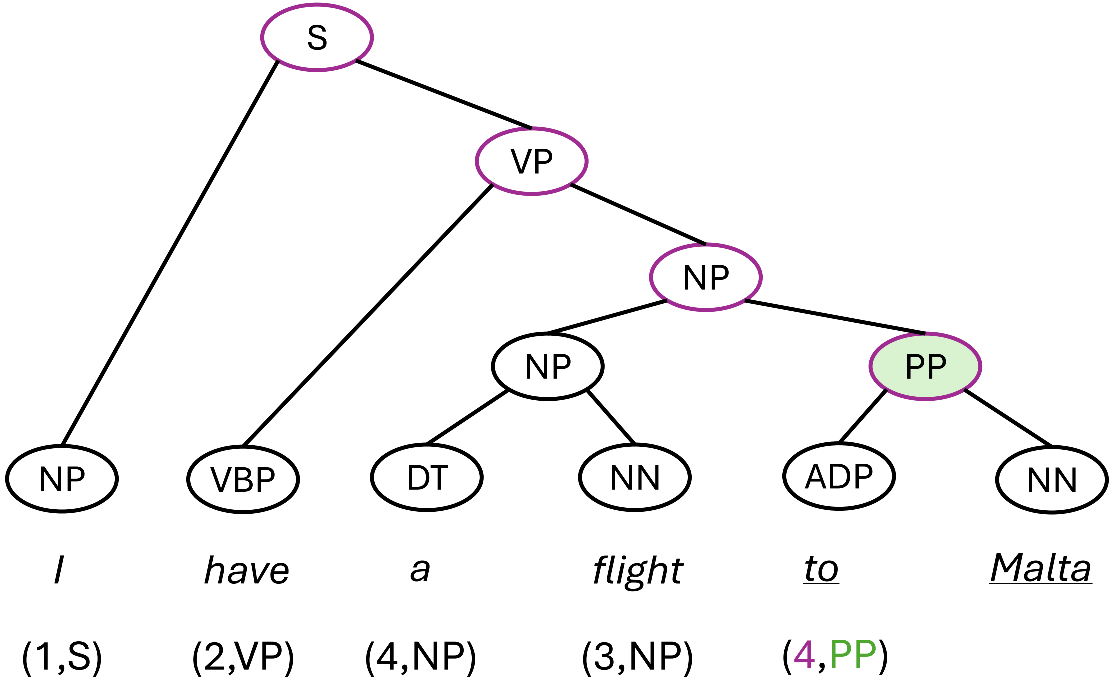
Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \to \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).

Incrementally obtain ℓ_i
and append right nodes.

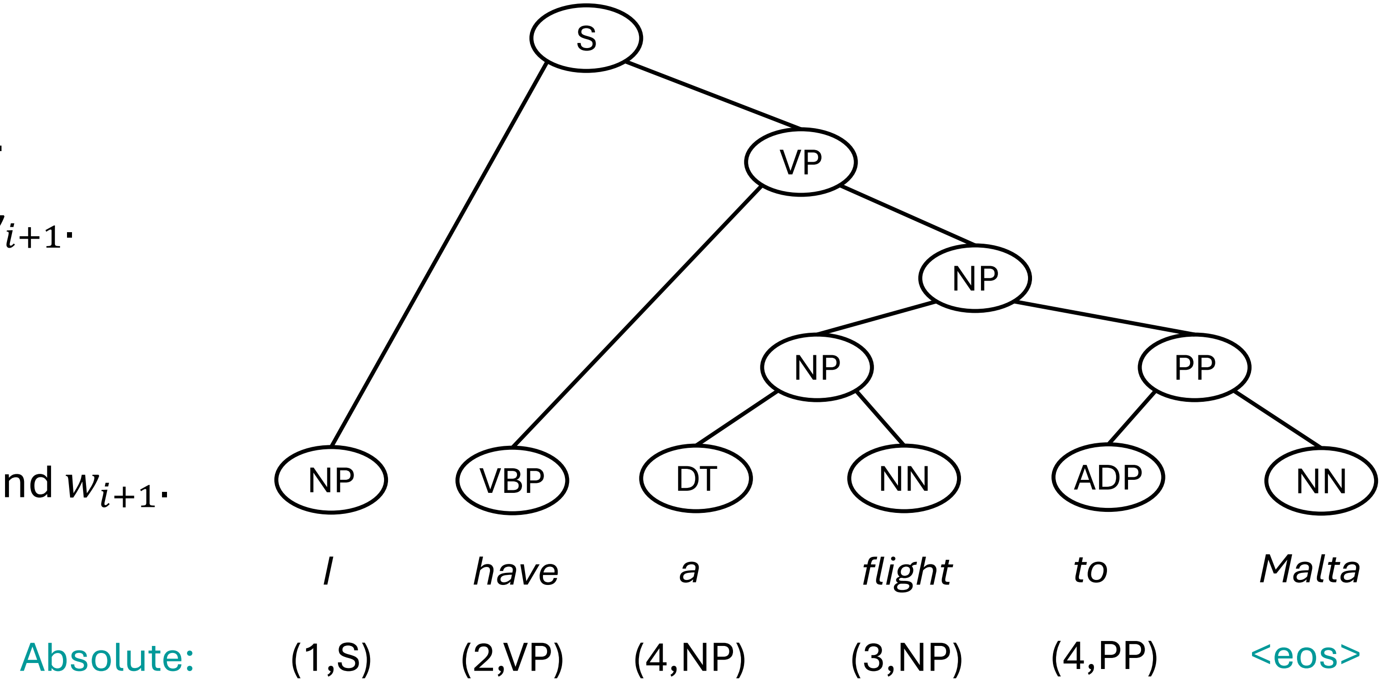


Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \rightarrow \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



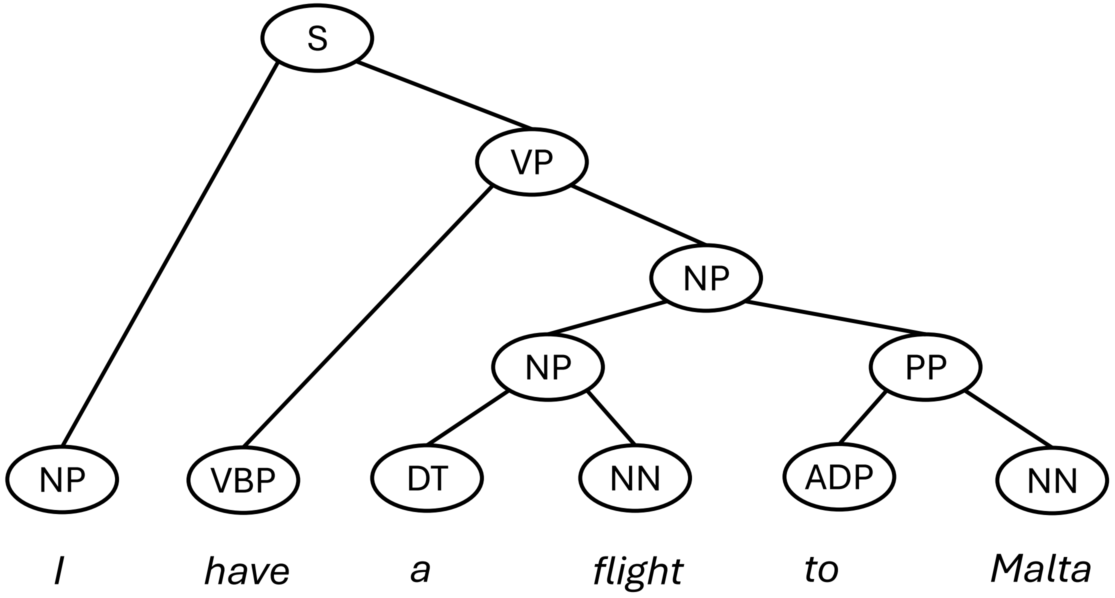
Absolute encoding!

Absolute & Relative Indexing (Gómez-Rodríguez & Vilares, 2018)

- Sequence labeling method.
- Sentence of n words to n labels.

$$w_1, \dots, w_n \rightarrow \ell_1, \dots, \ell_n$$

- Each label has 2 components: $\ell_i = (d_i, c_i)$.
  - l_i: # common constituents of w_i and w_{i+1}.
  - Absolute: $d_i = l_i$.
  - Relative: $d_i = l_i - l_{i-1}$.
  - c_i: lowest common constituent of w_i and w_{i+1}.
- Two feed forward networks (d_i, c_i).



Absolute:	(1,S)	(2,VP)	(4,NP)	(3,NP)	(4,PP)	<eos>
Relative:	(1,S)	(1,VP)	(2,NP)	(-1,NP)	(1,PP)	<eos>

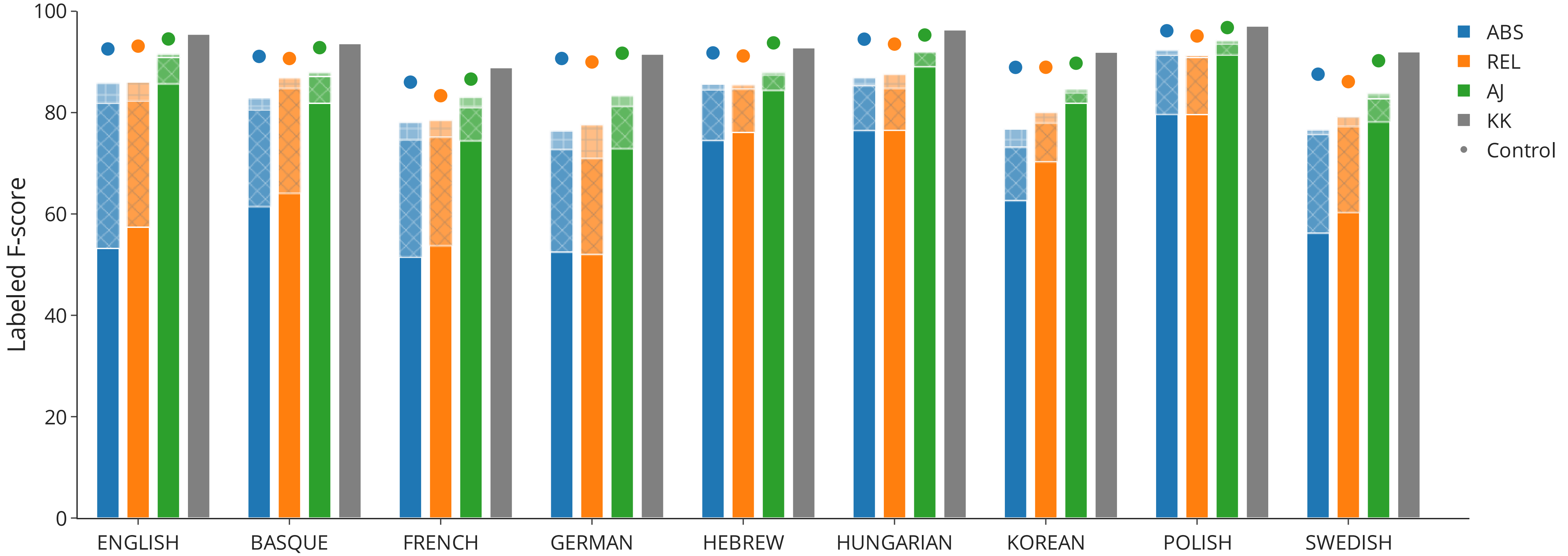
Relative: $d_i = l_i - l_{i-1}$

Experiments

- **Multilingual benchmark:** PTB + SPMRL (wo. Arabic).
- **Baseline:** Kitaev & Klein (2018).
    - Bidirectional encoder: XLM-RoBERTa.
    - Non-incremental decoder: span-based.
- **Encoders**:
    - Bidirectional: 4-BiLSTM, XLM-RoBERTa.
    - Unidirectional: 4-LSTM, BLOOM-560M, mGPT.
- **Delay experiments:** $k = 0, 1, 2$.

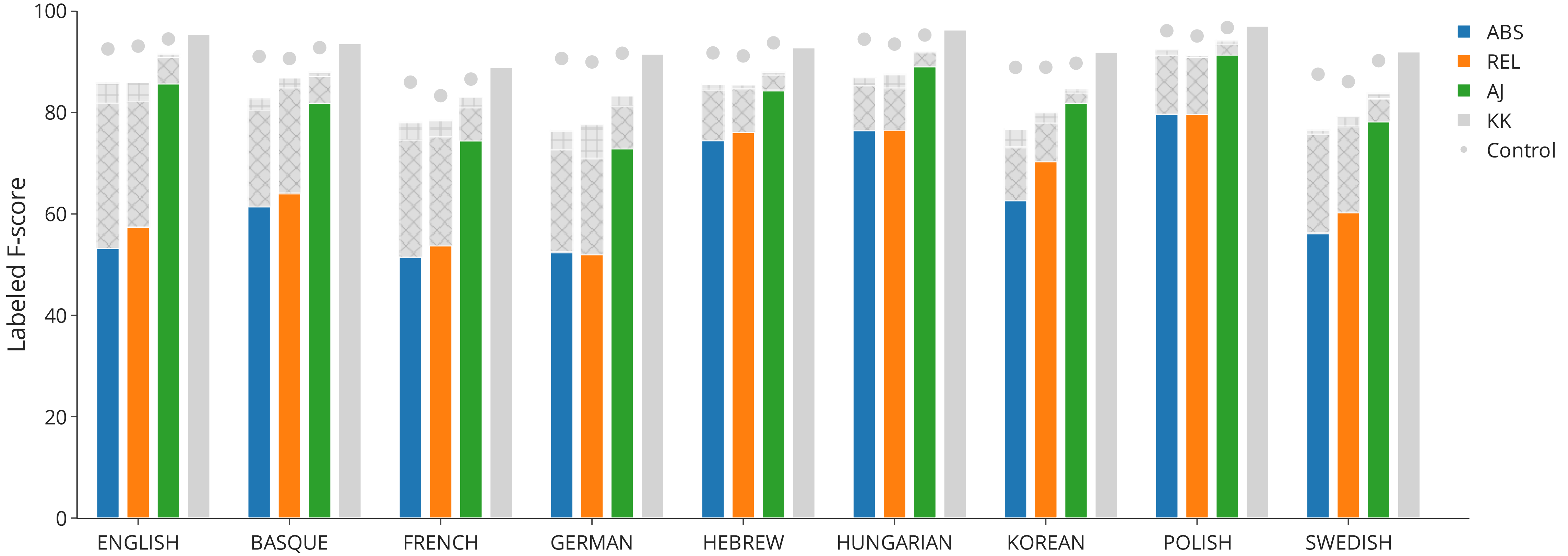
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (⊠, ⊠, ⊠) and **Delay 2** (⊞, ⊞, ⊞).



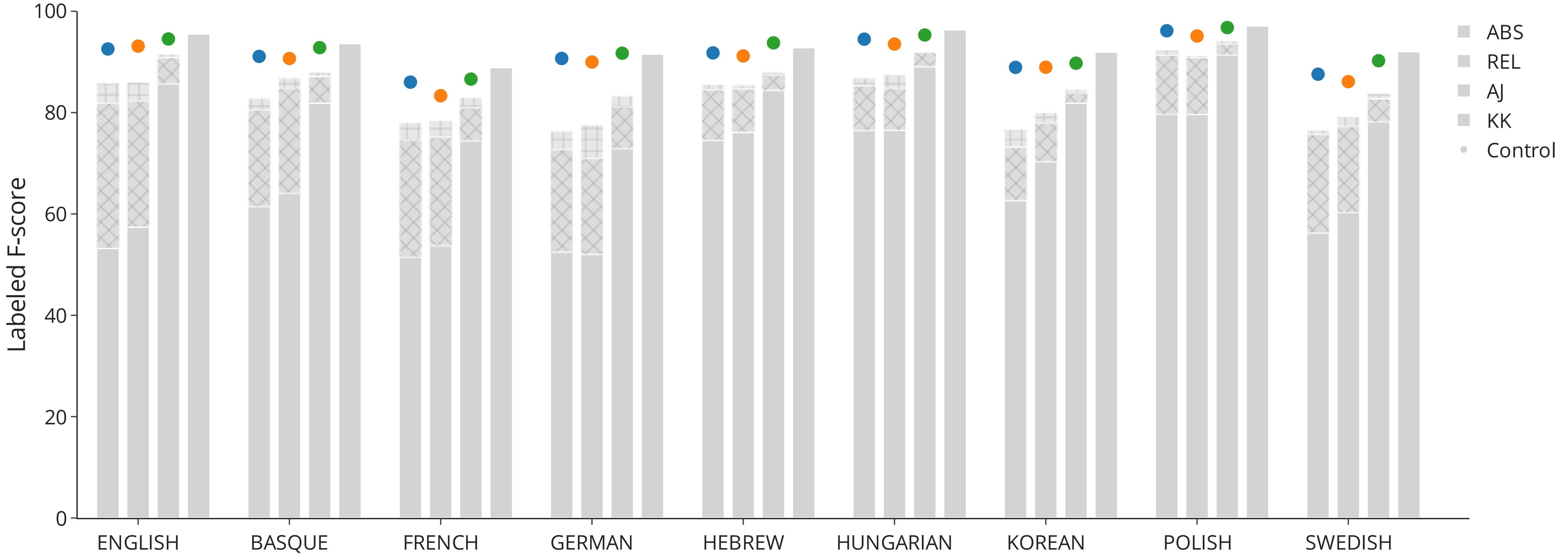
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (☒, ☒, ☒) and **Delay 2** (⊞, ⊞, ⊞).



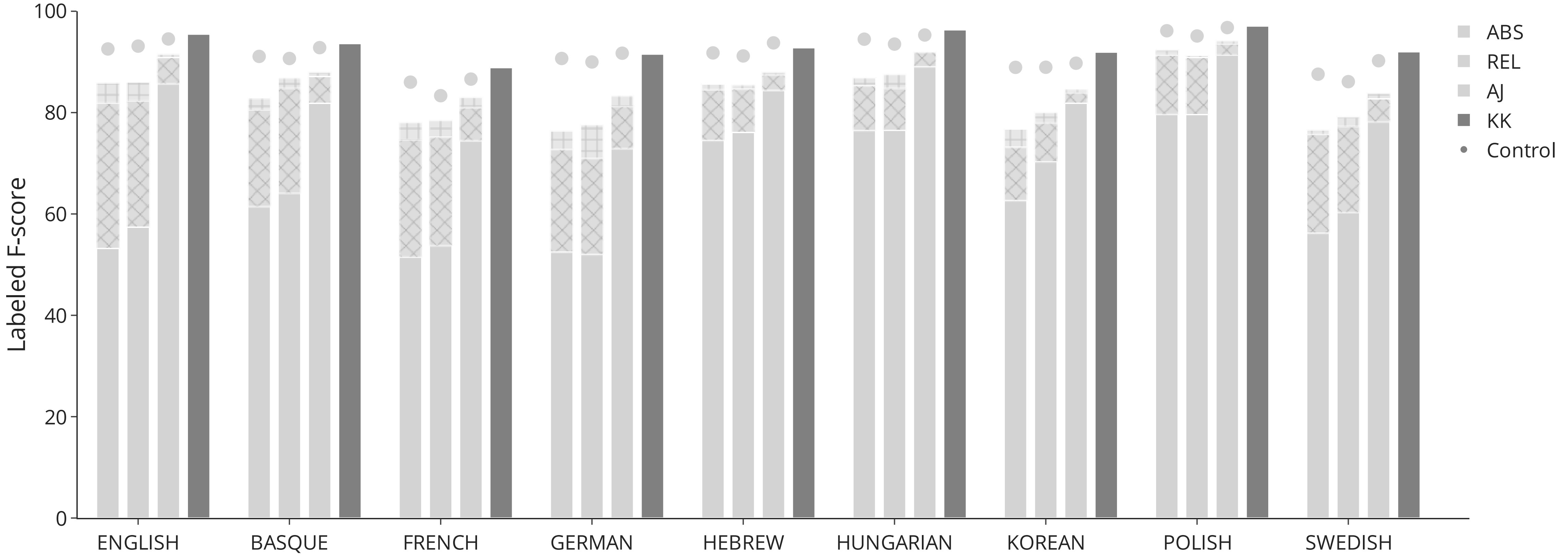
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (☒, ☒, ☒) and **Delay 2** (⊞, ⊞, ⊞).



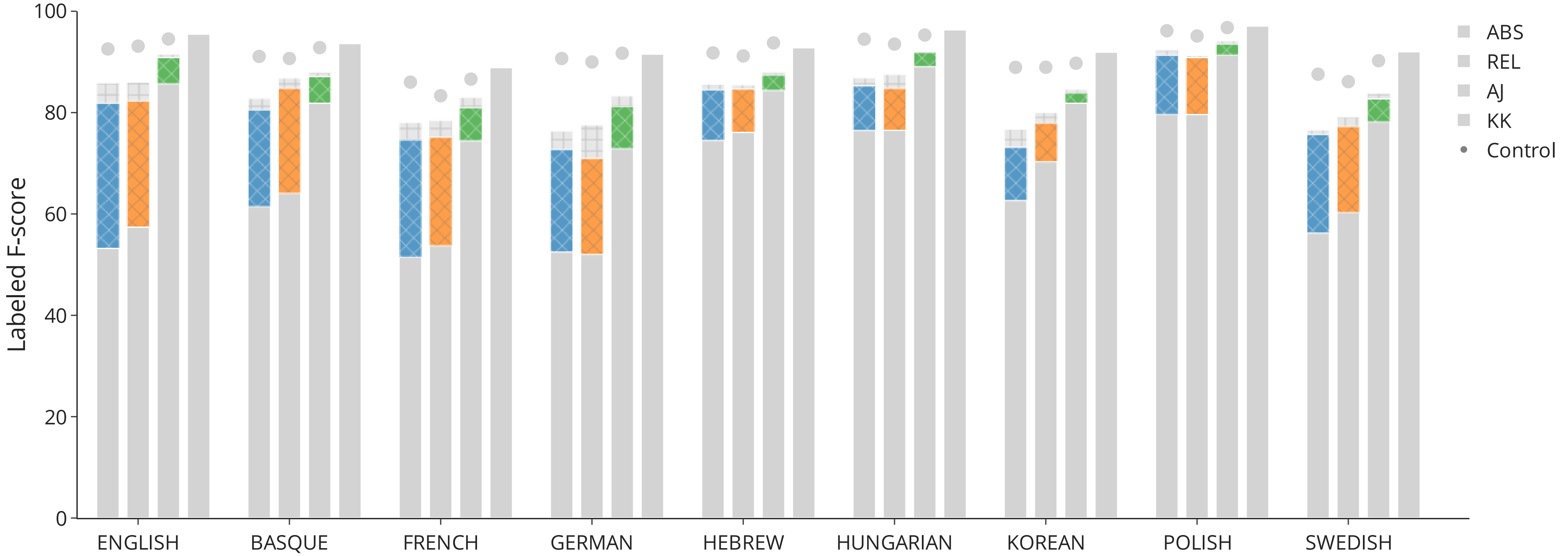
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (⊠, ⊠, ⊠) and **Delay 2** (⊞, ⊞, ⊞).



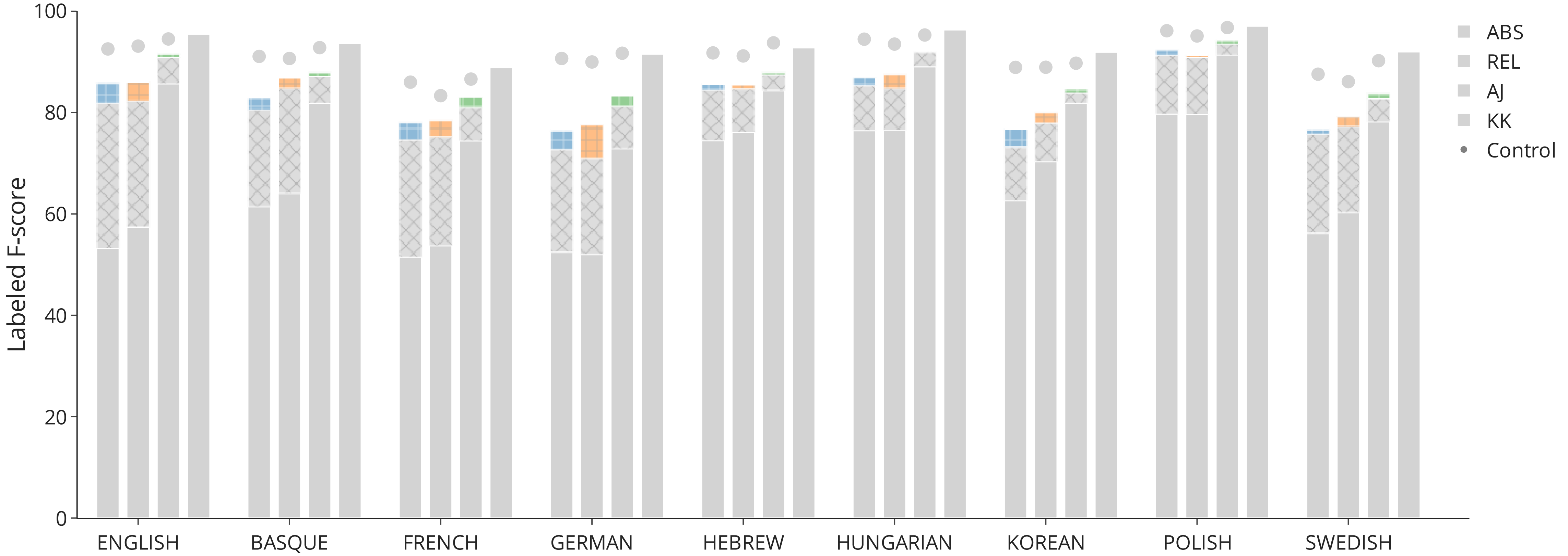
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (☒, ☒, ☒) and **Delay 2** (⊞, ⊞, ⊞).



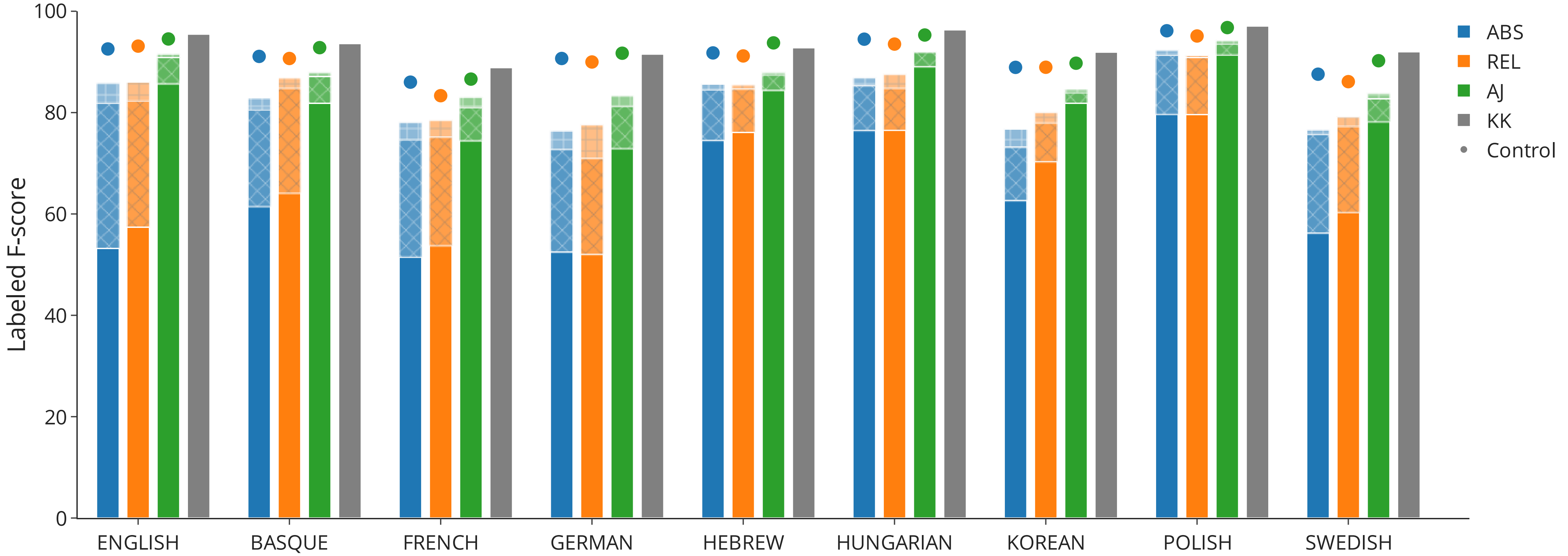
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (☒, ☒, ☒) and **Delay 2** (⊞, ⊞, ⊞).



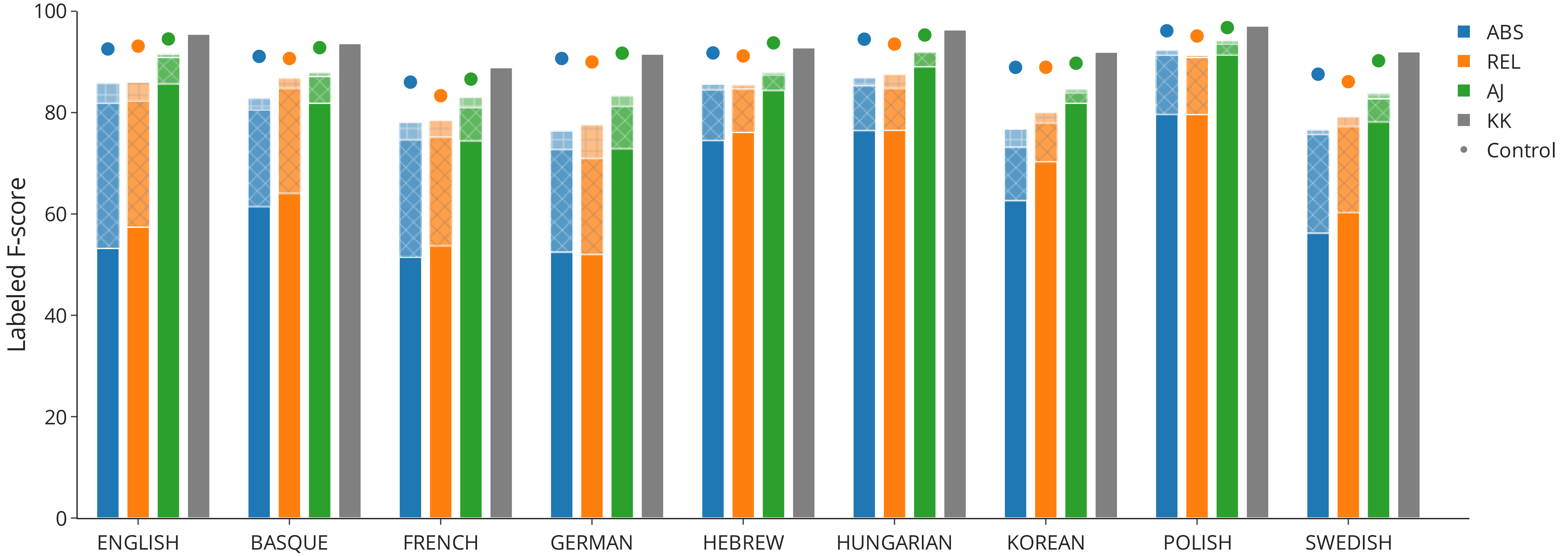
Results

- **Incremental** (mGPT): absolute (■), relative (■) and attach-juxtapose (■).
- **Control** (XLM): absolute (●), relative (●) and attach-juxtapose (●).
- **Non-incremental** (XLM): Kitaev & Klein, 2018 (■).
- **Delay 1** (⊠, ⊠, ⊠) and **Delay 2** (⊞, ⊞, ⊞).

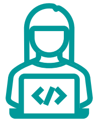


Conclusions

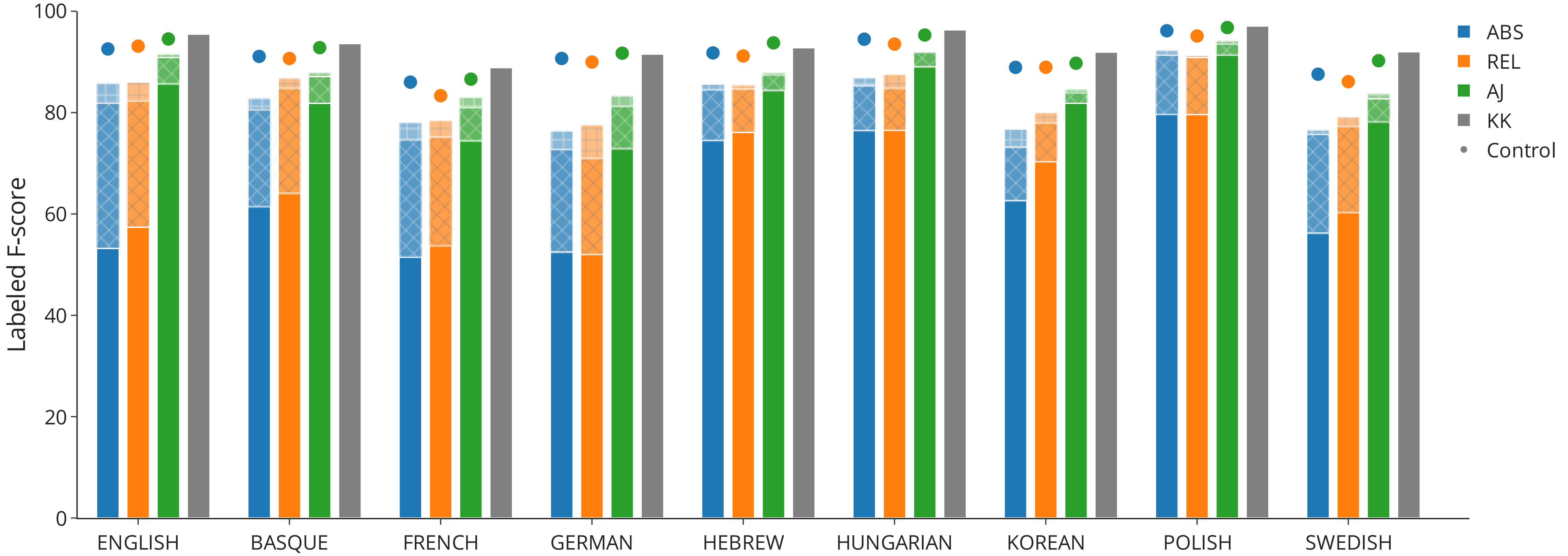
1. Control parsers (● ● ●) ≈ Kitaev & Klein, 2018 (■).
   - Meaning? State-of-the-art relies on a bidirectional encoder.
2. Incremental parsers (■ ■ ■) considerably worse than Control (● ● ●) and KK (■).
   - But introducing delay (☒, ☒, ☒) significantly improves the performance.



Conclusions



3. Sequence Labeling performance (■ ■) lags behind Attach-Juxtapose (■).
   - Attach-Juxtapose (■) relies on a powerful neural decoder (GCN).
   - Considering a larger decoder will improve the incremental results?
   - ABS (■) and REL (■) are more benefited of delayed processing than AJ (■).



Thanks for listening!

Ana Ezquerro Carlos Gómez-Rodríguez David Vilares

Acknowledgements:

This work has received funding by the European Research Council (ERC), under the Horizon Europe research and innovation programme (SALSA, grant agreement No 101100615), ERDF/MICINN-AEI (SCANNER-UDC, PID2020-113230RB-C21), Xunta de Galicia (ED431C 2020/11), Grant GAP (PID2022-139308OA-I00) funded by MCIN/AEI/10.13039/501100011033/ and by ERDF 'A way of making Europe', and Centro de Investigación de Galicia 'CITIC', funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centers of the Galician University System (CIGUS).