X-BAR LM

NYCU ILC
陳畯田
李恬甄
曾宜苹
甘沛潔

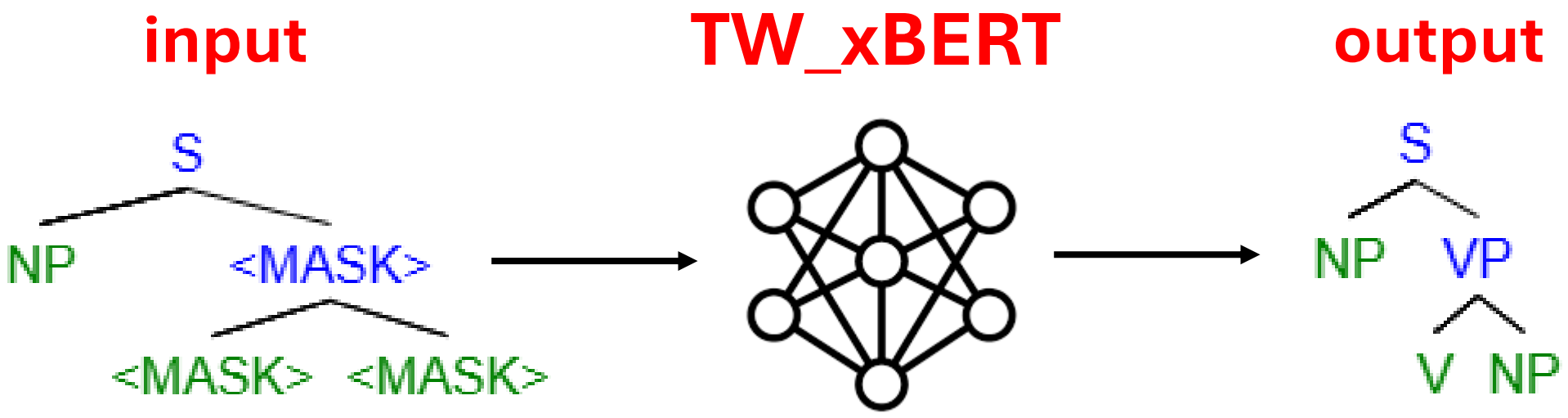
Roadmap

1. PHASE 1:
    1. Phrase structure prediction
    2. twStructure2enStructure: (tw structure predicts en structure)
2. PHASE 2:
    1. Structure-based word filling
3. Application
    1. Translation

Roadmap

1. PHASE 1:
    1. Phrase structure prediction
    2. twStructure2enStructure: (tw structure predicts en structure)
2. PHASE 2:
    1. Structure-based word filling
3. Application
    1. Translation

PHASE 1 – structure_model



Tokenization – POS

TW_sentence → **POS --- Articut** (Total 62 kinds of POS)

我是保羅

<pronoun>	我
<AUX>	是
<person>	保羅

我	</pronoun>
是	</AUX>
保羅	</person>

Tokenization – POS

“我是保羅”

Articut parse

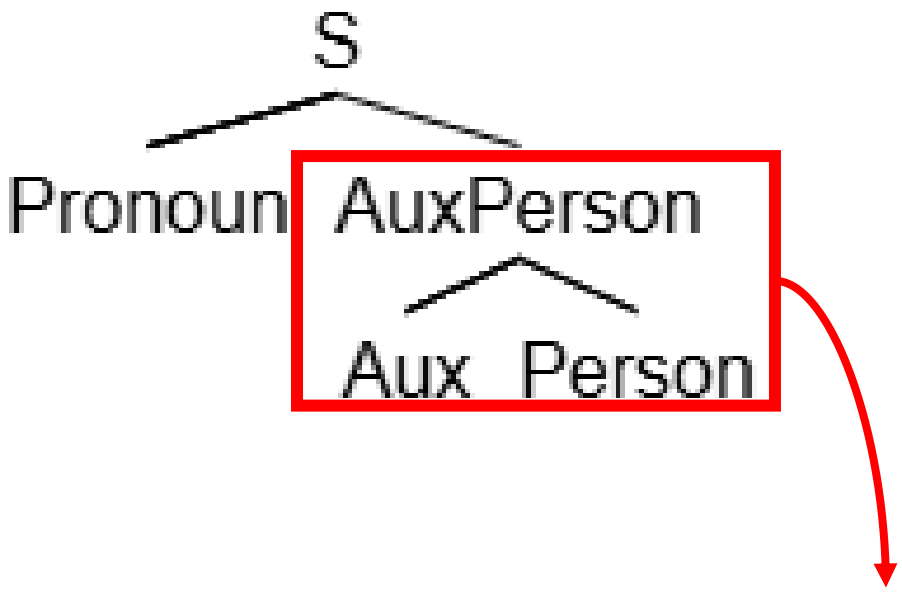
(Total 62 kinds of POS)

<pronoun>	我	</pronoun>
<AUX>	是	</AUX>
<person>	保羅	</person>

[<pronoun>, <AUX>, <person>]

Tokenization – Binary Tree Representation

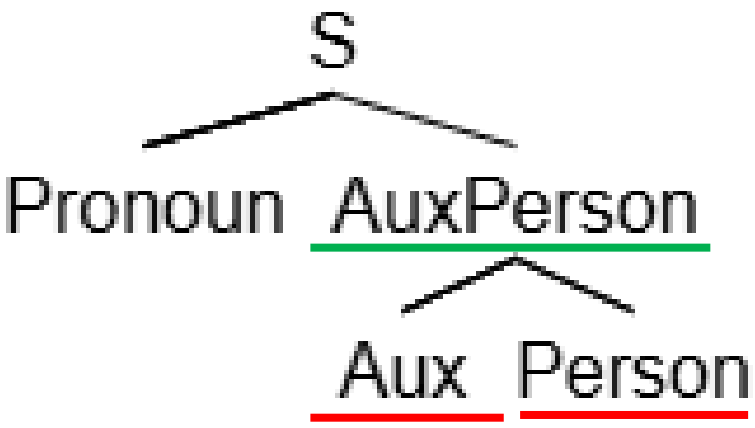
[**<pronoun>, <AUX>, <person>**]



[**<pronoun>,** [**<AUX>, <person>**]]

Tokenization – Binary Tree Representation

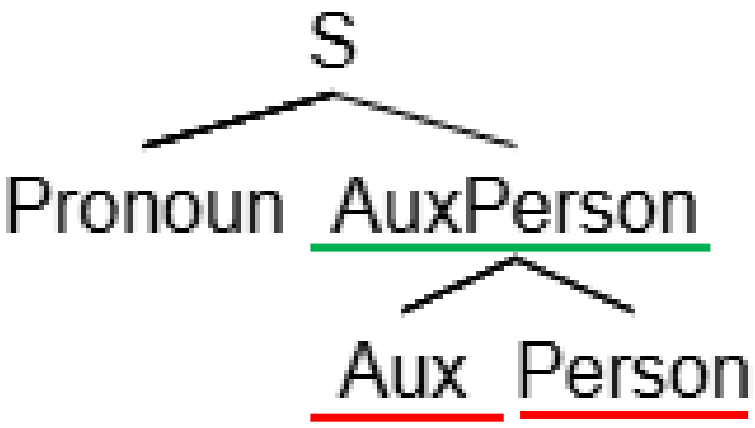
[**<pronoun>, [<AUX>, <person>]**]



[**<AUX><person>, <AUX>, <person>**]

Tokenization – Special Token

[**<pronoun>, [<AUX>, <person>]**]



[<pronoun>, <AUX><person>, <AUX>, <person>]

[<CLS> (BOS)]

[<SEP> (EOS)]

Input Tensor – Padding and Mapping

[<CLS>, <pronoun>, <AUX><person>, <AUX>, <person>, <SEP>(, <PAD>)]

<PAD> (if needed) to meet max_length, usually 6

Input Tensor – Padding and Mapping

[<CLS>, <pronoun>, <AUX><person>, <AUX>, <person>, <SEP>(, <PAD>)]

<PAD> (if needed) to meet max_length, usually 6

```
{
  "10" : "<PAD>",
  "20" : "<CLS>",
  "16" : "<AUX>",
  "30" : "<SEP>" ,
  "37" : "<person>",
  "39" : "<pronoun>",
}
```

[20, 39, 1637, 16, 37, 30]

Input Tensor – Second Mapping

1. Prevent overflow
2. Smaller numbers

[20, 39, 1637, 16, 37, 30]

```
{
   “6”   : “16”,
   “10” : ”20”,
   “19” : “30”,
   “26” : “37” ,
   “28” : “39”,
   “255” : “1637”,
}
```

```
{'attention_mask': tensor([[1, 1, 1, 1, 1, 1]]),
 'input_ids': tensor([[ 10,  28, 255,   6,  26,  19]]),
 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0]])}
```

‘input_ids’ : tensor([[10, 28, 255, 6, 26, 19]])

Input Tensor – attention mask

1. <PAD> is not context
2. *0 during sdpa

```
{'attention_mask': tensor([[1, 1, 1, 1, 1, 1]]),
 'input_ids': tensor([[ 10,  28, 255,   6,  26,  19]]),
 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0]])}
```

Input Tensor – token type

1. Mark next seq in NSP, Bert2Bert
2. [0, 0, 0, <SEP>, 1 , 1, 1]

```
{'attention_mask': tensor([[1, 1, 1, 1, 1, 1]]),
 'input_ids': tensor([[ 10,  28, 255,   6,  26,  19]]),
 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0]])}
```

For Bert2Bert:

input_id: [<CLS>, tw_pos, tw_ pos, tw_ pos, <SEP>, en_ pos, en_ pos, en_ pos, <SEP>]
token_type:[0, 0, 0, 0, 0, 1, 1, 1, 1]

Masking

‘input_ids’ : tensor([[10, 28, 255, 6, 26, 19]])

Each token has 15% chance being masked
80% will be replaced with mask_id (1829)
10% will be replaced with random id

‘input_ids’ : tensor([[10, 1829, 255, 6, 26, 19]])
‘labels’ : tensor([[-100, 28 , -100, -100, -100, -100]])

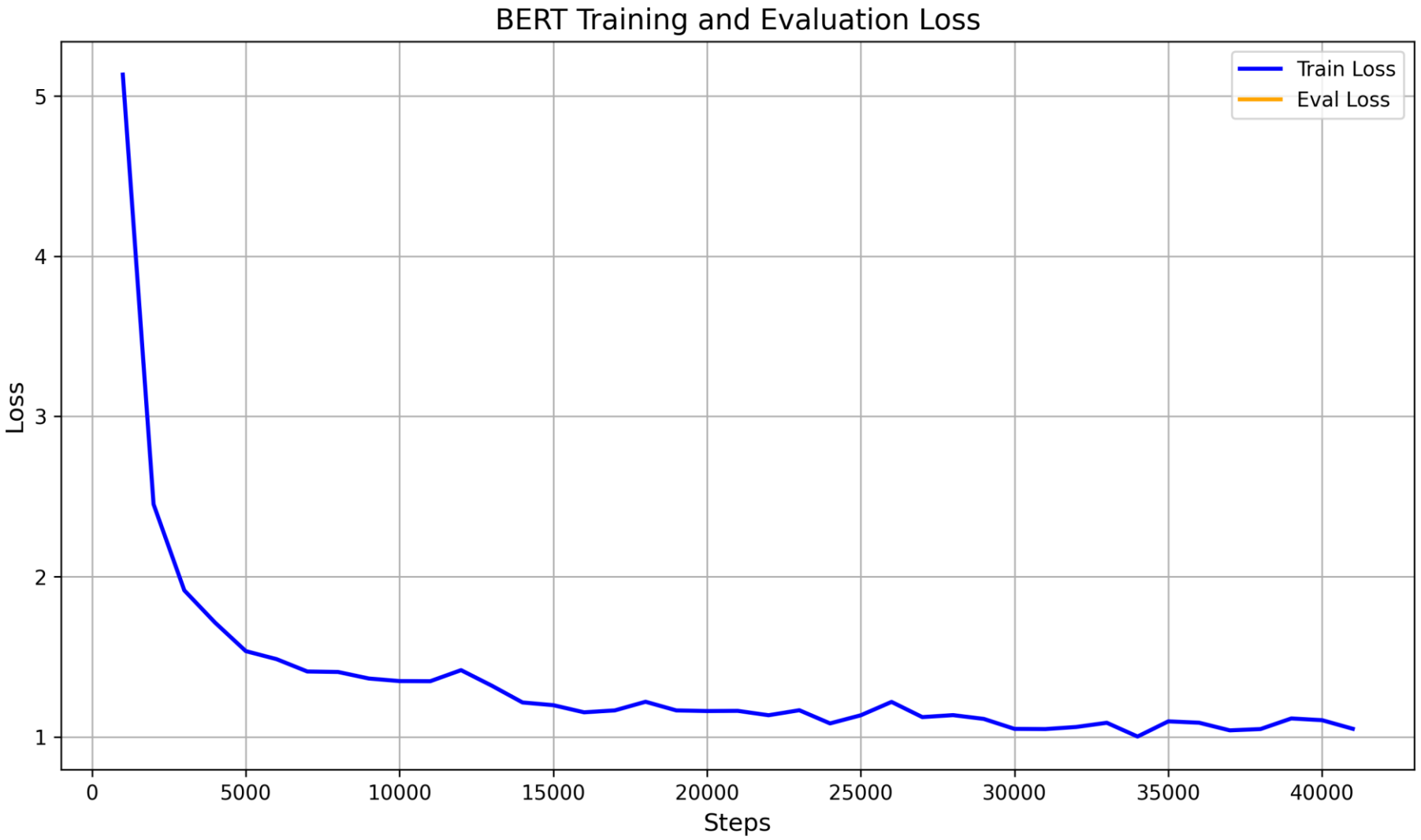
Training

‘input_ids’ : tensor([[10, 1829, 255, 6, 26, 19]])
‘attention_mask’: tensor([[1, 1, 1, 1, 1, 1]])
‘token_type_ids’: tensor([[0, 0, 0, 0, 0, 0]])
‘labels’ : tensor([[-100, 28, -100, -100, -100, -100]])

Training

On 3* NVIDIA RTX A5000
About 3~3.5 hr

Tw: 100k training data
En: 100k training data



```
BertForMaskedLM(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(1830, 576, padding_idx=0)
      (position_embeddings): Embedding(6, 576)
      (token_type_embeddings): Embedding(2, 576)
      (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
```

X-bar BERT vs BERT

	X-bar BERT	BERT base
vocab_size	1830 (POS combination)	30522 (token)
hidden_size	576	768
parameters	59 million	108 million
size/memory	229 MB	4 GB

```
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=576, out_features=576, bias=True)
              (key): Linear(in_features=576, out_features=576, bias=True)
              (value): Linear(in_features=576, out_features=576, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=576, out_features=576, bias=True)
              (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=576, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=576, bias=True)
            (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
```

```
  (cls): BertOnlyMLMHead(
    (predictions): BertLMPredictionHead(
      (transform): BertPredictionHeadTransform(
        (dense): Linear(in_features=576, out_features=576, bias=True)
        (transform_act_fn): GELUActivation()
        (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
      )
      (decoder): Linear(in_features=576, out_features=1830, bias=True)
```

```
BertForMaskedLM(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(1830, 576, padding_idx=0)
      (position_embeddings): Embedding(6, 576)
      (token_type_embeddings): Embedding(2, 576)
      (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
```

X-bar BERT vs BERT (under same layer config)

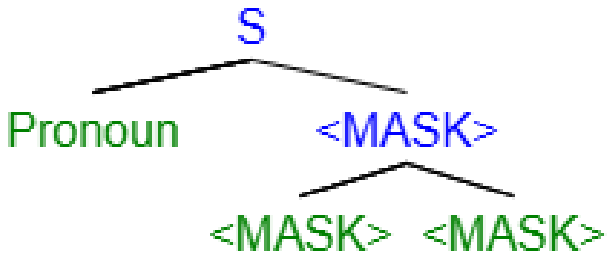
	X-bar BERT	BERT base
vocab_size	1830 (POS combination)	30522 (token)
hidden_size	576	768
parameters	59 million	108 million
size/memory	229 MB	4 GB

```
(encoder): BertEncoder(
  (layer): ModuleList(
    (0-11): 12 x BertLayer(
      (attention): BertAttention(
        (self): BertSdpaSelfAttention(
          (query): Linear(in_features=576, out_features=576, bias=True)
          (key): Linear(in_features=576, out_features=576, bias=True)
          (value): Linear(in_features=576, out_features=576, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=576, out_features=576, bias=True)
          (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=576, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=576, bias=True)
        (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
```

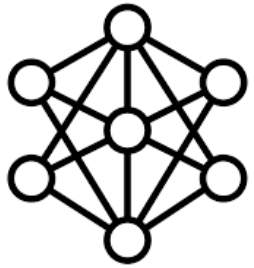
```
(cls): BertOnlyMLMHead(
  (predictions): BertLMPredictionHead(
    (transform): BertPredictionHeadTransform(
      (dense): Linear(in_features=576, out_features=576, bias=True)
      (transform_act_fn): GELUActivation()
      (LayerNorm): LayerNorm((576,), eps=1e-12, elementwise_affine=True)
    )
    (decoder): Linear(in_features=576, out_features=1830, bias=True)
```

Evaluation

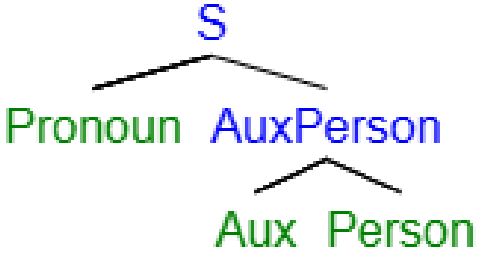
input



TW_xBERT



output



'input_ids' : tensor([[10, 28, 1829, 1829, 1829, 19]])
'labels' : tensor([[-100, -100, 255, 6, 26, -100]])

→ **[255, xx, xx]**

Evaluation

‘input_ids’ : tensor([[10, 28, 1829, 1829, 1829, 19]])
‘labels’ : tensor([[-100, -100, 255, 6, 26, -100]])

predict

[255, xx, xx]

The model does not always have to predict the inner structure correctly.

decode

[1637, 16, 37]

The decoder recovers inner structure with predicted phrase.

decode

[<AUX><person>, <AUX>, <person>]

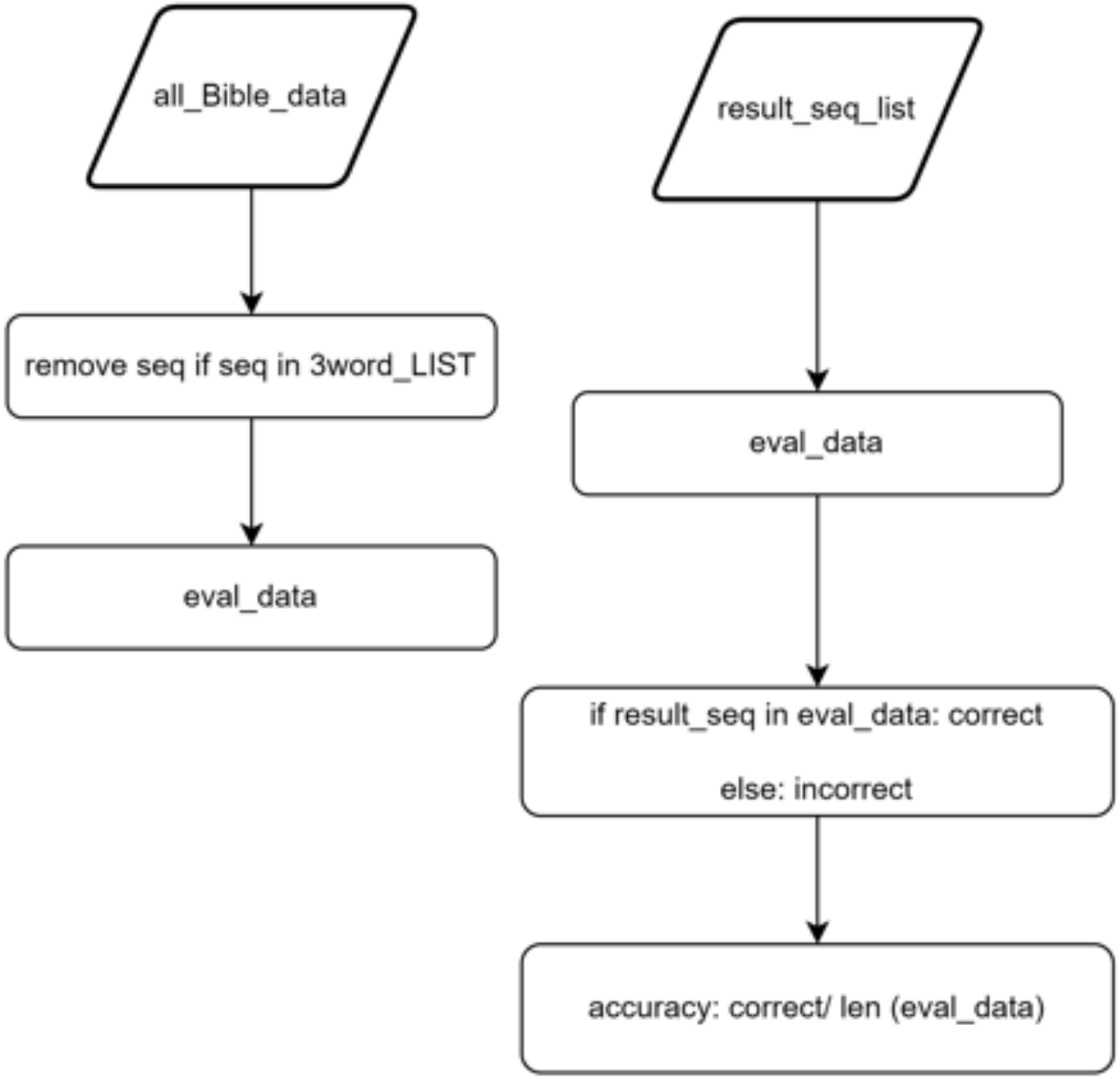
Fill in

[<pronoun>, <AUX>, <person>]

Evaluate

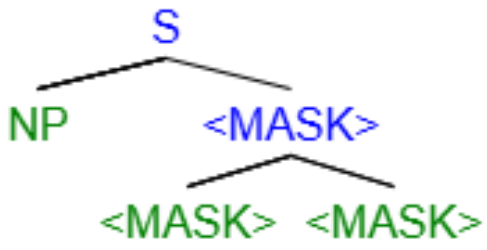
1. Seq is reasonable?
2. Seq exist in data?

Evaluation

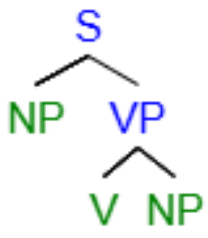

all_Bible_data
remove seq if seq in 3word_LIST
eval_data
result_seq_list
eval_data
if result_seq in eval_data: correct
else: incorrect
accuracy: correct/ len (eval_data)

Accuracy

Tw: 100k training data
En: 100k training data



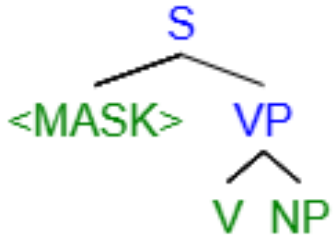
Given Head



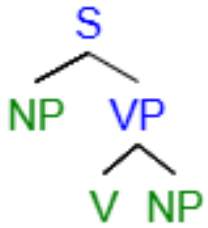
Predict complement

$$\frac{existPredSeqCount}{totalPredSeqCount}$$

Tw 99.6%
En 99.5%



Given complement



Predict Head

$$\frac{existPredSeqCount}{totalPredSeqCount}$$

Tw 97.9%
En 99.2%

Accuracy

Tw: 100k training data
En: 100k training data

Without Built-in Syntactic Unpacking:

- Out of **1682 predictions**:
  → **723** (43%) correctly match the structure `phrase = concat(tag, tag)`
  → **959** (57%) do **not** match
- Among the **723 correct** structures:
  → **719** (99.4%) are **real sequences** found in the Bible data
  → This accounts for **42.7% of all sentences**

Accuracy

Tw: 100k training data
En: 100k training data

With Built-in Syntactic Unpacking:

- The decoder **automatically fills in** the `phrase = concat(tag, tag)` structure
  → Makes the structural match effectively **100%**
- Among these:
  → **99.6%** are real sequences (from Bible data)
- For English (En):
  → Baseline real-sequence rate: **19%**
  → With syntactic unpacking: **99.5%**

Roadmap

1. PHASE 1:
    1. Phrase structure prediction
    2. twStructure2enStructure: (tw structure predicts en structure)
2. PHASE 2:
    1. Structure-based word filling
3. Application
    1. Translation

Thank You ~