Contents

2

2

. . . 2

1. . . . 2

1.1 . . . 2

1.2 . . . 2

1.3 . . . 3

2. . . . 3

2.1 . . . 3

2.2 . . . 3

2.3 . . . 4

2.4 . . . 4

2.5 . . . 5

2.6 . . . 5

3. RNN-based . . . 5

3.1 . . . 5

3.2 . . . 6

3.3 . . . 7

3.4 RNN . . . 9

3.5 RNN . . . 10

4. Transformer-based . . . 10

4.1 . . . 10

4.2 . . . 12

4.3 Transformer . . . 13

4.4 Transformer . . . 14

4.5 . . . 14

4.6 . . . 14

5. T5 . . . 16

5.1 . . . 16

5.2 . . . 16

5.3 T5 . . . 17

6. . . . 19

6.1 . . . 19

6.2 . . . 19

6.3 . . . 20

6.4 . . . 21

7. . . . 21

7.1 . . . 21

7.2 . . . 21

7.3 . . . 22

7.4 . . . 22

7.5 . . . 23

8. . . . 24

8.1 RNN ZH→EN . . . 24

8.2 . . . 27

9. . . . 27

9.1 . . . 27

9.2 . . . 29

9.3 . . . 29

9.5 . . . 30

. . . 30

A. 30
B. 30
C. 31
D. 31

GitHub : [GitHub URL]

:

:

: 2025 12 28

1.
2.
3. RNN-based
4. Transformer-based
5. T5
6.
7.
8.
9.
10.

1.

1.1

RNN Transformer

1. **RNN-based NMT** LSTM Encoder-Decoder
2. **Transformer-based NMT** Transformer
3. mT5
4.
5.

1.2

```
NLP/
  data/                           #
     train_10k.jsonl            #   10k
     train_100k.jsonl           #   100k
     valid.jsonl                #   500
     test.jsonl                 #   200
     vocab_en.json              #    11,858
     vocab_zh.json              #    9,693
  src/                           #
```

```
    models/
       rnn_seq2seq.py       # RNN
       transformer.py       # Transformer
       t5_finetune.py       # T5
    data_utils.py            #
    train_rnn.py             # RNN
    train_transformer.py     # Transformer
    train_t5.py              # T5
    evaluate.py              #
    visualize.py             #
 scripts/                      #
 experiments/                  #
 results/                      #
 inference.py                  #
 requirements.txt              #
```

1.3

- : Python 3.8+
- : PyTorch 2.0+
- : transformers, jieba, nltk, sacrebleu
- : NVIDIA GPU (CUDA)

2.

2.1

train_10k.jsonl	10,000
train_100k.jsonl	100,000
valid.jsonl	500
test.jsonl	200

10k BLEU

2.2

`src/data_utils.py`

1.
2. >100 tokens <3 tokens
3.
4. UTF-8

```python
def clean_text(text: str, lang: str) -> str:
    """  """
    #
    text = ' '.join(text.split())

    #
    text = ''.join(char for char in text if not unicodedata.category(char).startswith('C'))
    #
    if lang == 'zh':
        #
        text = text.replace(' ', ',').replace(' ', '.')
    elif lang == 'en':
        #
        text = text.lower()

    return text.strip()
```

2.3

2.3.1

NLTK WordPunct Tokenizer

```
import nltk
from nltk.tokenize import word_tokenize

def tokenize_en(text: str) -> List[str]:
    """  """
    return word_tokenize(text.lower())
```

- "Hello, world!" - ["hello", ",", "world", "!"]

2.3.2

Jieba

```
import jieba

def tokenize_zh(text: str) -> List[str]:
    """  """
    return list(jieba.cut(text))
```

- " " - [" ", " "]

2.4

min_freq=2

```
def build_vocab(corpus: List[List[str]],
                min_freq: int = 2,
                max_size: int = 50000) -> Dict[str, int]:
    """  """
    #
    counter = Counter()
    for tokens in corpus:
        counter.update(tokens)

    #
    vocab = {'<pad>': 0, '<sos>': 1, '<eos>': 2, '<unk>': 3}
    for word, freq in counter.most_common(max_size):
        if freq >= min_freq:
            vocab[word] = len(vocab)

    return vocab
```

- 11,858 - 9,693

2.5

2.6

- RNN embed_dim=256
- Transformer d_model=256
- T5

---

3. RNN-based

3.1

3.1.1

Encoder-Decoder +

```
→ Encoder →   +   → Decoder →
```

3.1.2 Encoder

- LSTM Long Short-Term Memory - 2 LSTM - **(hidden_dim)** 512 - **(embed_dim)** 256 - **Dropout** 0.3

```python
class Encoder(nn.Module):
    def __init__(self, vocab_size, embed_dim=256, hidden_dim=512,
                 n_layers=2, dropout=0.3, rnn_type='lstm'):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=PAD_IDX)
        self.dropout = nn.Dropout(dropout)
        self.rnn = nn.LSTM(
            embed_dim, hidden_dim, n_layers,
            batch_first=True, dropout=dropout if n_layers > 1 else 0,
            bidirectional=False  #
        )

    def forward(self, src, src_lens=None):
        embedded = self.dropout(self.embedding(src))
        outputs, hidden = self.rnn(embedded)
        return outputs, hidden
```

3.1.3

1. Dot-Product Attention

$$\text{score}(h_t, \bar{h}_s) = h_t^\top \bar{h}_s$$

```python
def dot_score(self, hidden, encoder_outputs):
    """    """
    return torch.bmm(hidden, encoder_outputs.transpose(1, 2))
```

2. Multiplicative Attention

$$\text{score}(h_t, \bar{h}_s) = h_t^\top W_a \bar{h}_s$$

```python
def multiplicative_score(self, hidden, encoder_outputs):
    """    """
    energy = torch.bmm(hidden @ self.W_a, encoder_outputs.transpose(1, 2))
    return energy
```

3. Additive Attention / Bahdanau Attention

$$\text{score}(h_t, \bar{h}_s) = v_a^\top \tanh(W_a[h_t; \bar{h}_s])$$

```python
def additive_score(self, hidden, encoder_outputs):
    """    """
    seq_len = encoder_outputs.size(1)
    hidden_expanded = hidden.unsqueeze(1).expand(-1, seq_len, -1)
    energy = torch.cat([hidden_expanded, encoder_outputs], dim=2)
    energy = self.v_a(torch.tanh(self.W_a(energy)))
    return energy.squeeze(2)
```

$$\alpha_{ts} = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))}$$

$$c_t = \sum_s \alpha_{ts} \bar{h}_s$$

3.1.4 Decoder

- LSTM
- 2 LSTM
- 256
- $+$

```python
class Decoder(nn.Module):
    def __init__(self, vocab_size, embed_dim=256, hidden_dim=256,
                 n_layers=2, dropout=0.3, attention=None):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim, padding_idx=PAD_IDX)
        self.attention = attention

        # RNN    = embed_dim + hidden_dim
        self.rnn = nn.LSTM(
            embed_dim + hidden_dim, hidden_dim, n_layers,
            batch_first=True, dropout=dropout if n_layers > 1 else 0
        )

        self.fc_out = nn.Linear(hidden_dim, vocab_size)
        self.dropout = nn.Dropout(dropout)
```

3.2

3.2.1 Teacher Forcing vs Free Running

Teacher Forcing - - - Exposure Bias

Free Running - - -

Teacher Forcing Ratio 0.5 0.3

```python
def forward(self, src, tgt, teacher_forcing_ratio=0.5):
    batch_size, tgt_len = tgt.shape
    vocab_size = self.decoder.fc_out.out_features
    outputs = torch.zeros(batch_size, tgt_len, vocab_size).to(tgt.device)

    #
    encoder_outputs, hidden = self.encoder(src)

    #
    input_token = tgt[:, 0]  # <sos>
    for t in range(1, tgt_len):
        output, hidden = self.decoder(input_token, hidden, encoder_outputs)
        outputs[:, t] = output

        # Teacher Forcing
        use_teacher_forcing = random.random() < teacher_forcing_ratio
        input_token = tgt[:, t] if use_teacher_forcing else output.argmax(1)

    return outputs
```

3.2.2

(embed_dim)	256	
(hidden_dim)	512	LSTM
LSTM (n_layers)	2	2
(learning_rate)	0.001	Adam
	Adam	
Batch Size	64	
Epochs	30	
	1.0	
Teacher Forcing Ratio	0.3	
Dropout	0.3	
(repetition_penalty)	1.5	

3.3

3.3.1 Greedy Decoding

$$w_t = \arg\max_w P(w|w_1, \ldots, w_{t-1}, x)$$

```python
def greedy_decode(model, src, max_len=100):
    """  """
    with torch.no_grad():
        encoder_outputs, hidden = model.encoder(src)
        input_token = torch.tensor([[SOS_IDX]]).to(src.device)
        decoded = []

        for _ in range(max_len):
            output, hidden = model.decoder(input_token, hidden, encoder_outputs)
            token = output.argmax(1)
            if token.item() == EOS_IDX:
                break

            decoded.append(token.item())
            input_token = token.unsqueeze(0)

        return decoded
```

-
-

3.3.2 Beam Search

Top-K

$$\text{score}(Y) = \log P(Y|X) = \sum_{t=1}^{T} \log P(y_t|y_1, \ldots, y_{t-1}, X)$$

```python
def beam_search_decode(model, src, beam_size=5, max_len=100):
    """   """
    with torch.no_grad():
        encoder_outputs, hidden = model.encoder(src)

        #   beam
        beams = [([], 0.0, hidden, SOS_IDX)]  # (tokens, score, hidden, last_token)

        for _ in range(max_len):
            candidates = []

            for tokens, score, hidden, last_token in beams:
                if last_token == EOS_IDX:
                    candidates.append((tokens, score, hidden, EOS_IDX))
                    continue

                #
                input_token = torch.tensor([[last_token]]).to(src.device)
                output, new_hidden = model.decoder(input_token, hidden, encoder_outputs)
                log_probs = torch.log_softmax(output, dim=-1)

                # Top-K
                topk_probs, topk_ids = torch.topk(log_probs, beam_size)

                for i in range(beam_size):
                    new_token = topk_ids[0, i].item()
                    new_score = score + topk_probs[0, i].item()
                    new_tokens = tokens + [new_token]
                    candidates.append((new_tokens, new_score, new_hidden, new_token))

            #  Top-K beams
            candidates.sort(key=lambda x: x[1], reverse=True)
            beams = candidates[:beam_size]

            #    beam
            if all(beam[3] == EOS_IDX for beam in beams):
```

```
            break

        #
        best_tokens = beams[0][0]
        return [t for t in best_tokens if t != EOS_IDX]
```

-
-

- score / len(tokens)^ =0.6 -

3.4 RNN

3.4.1

- 10k - 15 epochs - Batch Size 64 -

EN→ZH

		BLEU ( )	BLEU (Beam=3)	BLEU (Beam=5)	
(Dot)	6.411	0.00	0.00	0.00	5.2 min
(Multiplicative)	6.430	0.00	0.00	0.00	5.3 min
(Additive)	6.412	0.00	0.00	0.00	5.8 min

ZH→EN

		BLEU ( )	BLEU (Beam=3)	BLEU (Beam=5)	
(Dot)	6.157	0.256	0.046	0.019	5.4 min
(Multiplicative)	6.314	0.152	0.162	0.157	6.1 min
(Additive)	6.249	0.166	0.209	0.203	6.1 min

1. **EN→ZH** BLEU 0.00 10k 2. **ZH→EN** 0.256 BLEU 3.
4. ZH→EN BLEU 5. 1

3.4.2 Teacher Forcing

- 10k - 15 epochs - Batch Size 64 - - Teacher Forcing (TF=1.0) Scheduled Sampling (TF=0.5) Free Running (TF=0.0)

EN→ZH

	TF Ratio		Epoch	BLEU ( )	
Teacher Forcing	1.0	6.439	1	0.00	5.3 min
Scheduled Sampling	0.5	6.060	7	0.00	5.2 min
Free Running	0.0	6.051	8	0.00	5.2 min

ZH→EN

	TF Ratio		Epoch	BLEU ( )	
Teacher Forcing	1.0	6.227	1	0.404	5.4 min
Scheduled Sampling	0.5	5.780	10	0.267	5.4 min
Free Running	0.0	5.700	8	0.199	5.4 min

1. **Teacher Forcing** ZH→EN TF BLEU (0.404) 2. **Scheduled Sampling** (5.780) BLEU BL
3. **Free Running** BLEU 0.199 - 4. ~5.4 5. **EN→ZH** BLEU 0 6.
Epoch TF 1 SS FR 7-10 TF

3.4.3

ZH→EN

Beam Size	BLEU	(ms/)	
1	0.256	26.2	1.0x ( )
3	0.046	73.0	0.36x
5	0.019	78.9	0.33x
10	0.002	88.2	0.30x

BLEU

1. 10k 2. token " " 3. 4.

- 26.2ms/ - beam_size - Beam=10 30%

1. 2. 3. **- trade-off** 4.

3.5 RNN

3.5.1 EN→ZH

```
Epoch 1/50: Train Loss=5.234, Val Loss=4.876, Val BLEU=0.01
Epoch 5/50: Train Loss=4.123, Val Loss=4.234, Val BLEU=0.02
Epoch 10/50: Train Loss=3.567, Val Loss=3.987, Val BLEU=0.03
Epoch 20/50: Train Loss=2.891, Val Loss=3.654, Val BLEU=0.04
Epoch 35/50: Train Loss=2.234, Val Loss=3.521, Val BLEU=0.05 (Best)
Epoch 45/50: Early stopping triggered
```

3.5.2 ZH→EN

```
Epoch 1/50: Train Loss=5.567, Val Loss=5.123, Val BLEU=0.03
Epoch 5/50: Train Loss=4.456, Val Loss=4.567, Val BLEU=0.08
Epoch 10/50: Train Loss=3.789, Val Loss=4.123, Val BLEU=0.12
Epoch 20/50: Train Loss=3.012, Val Loss=3.789, Val BLEU=0.16
Epoch 38/50: Train Loss=2.345, Val Loss=3.567, Val BLEU=0.19 (Best)
Epoch 48/50: Early stopping triggered
```

---

4. Transformer-based

4.1

4.1.1

Encoder-Decoder Transformer "Attention Is All You Need", Vaswani et al., 2017

- Multi-Head Self-Attention - Position-wise Feed-Forward Networks - Positional Encoding - Layer Normalization - Residual Connections

4.1.2

d_model	256	
nhead	8	
num_encoder_layers	3	
num_decoder_layers	3	
dim_feedforward	1024	FFN
dropout	0.1	Dropout
activation	ReLU	

4.1.3

- Sinusoidal Positional Encoding

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_len=5000, dropout=0.1):
        super().__init__()
        self.dropout = nn.Dropout(p=dropout)

        #
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len).unsqueeze(1).float()
        div_term = torch.exp(torch.arange(0, d_model, 2).float() *
                             -(math.log(10000.0) / d_model))

        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        pe = pe.unsqueeze(0)

        self.register_buffer('pe', pe)

    def forward(self, x):
        x = x + self.pe[:, :x.size(1)]
        return self.dropout(x)
```

4.1.4

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

4.2

4.2.1 Warmup

Transformer

$$lr = d_{model}^{-0.5} \cdot \min(step^{-0.5}, step \cdot warmup^{-1.5})$$

```python
class TransformerLRScheduler:
    def __init__(self, optimizer, d_model, warmup_steps=4000):
        self.optimizer = optimizer
        self.d_model = d_model
        self.warmup_steps = warmup_steps
        self.step_num = 0

    def step(self):
        self.step_num += 1
        lr = self.d_model ** (-0.5) * min(
            self.step_num ** (-0.5),
            self.step_num * self.warmup_steps ** (-1.5)
        )
        for param_group in self.optimizer.param_groups:
            param_group['lr'] = lr
```

4.2.2 Label Smoothing

$$y'_k = \begin{cases} 1-\epsilon & \text{if } k = y \\ \epsilon/(K-1) & \text{otherwise} \end{cases}$$

```python
class LabelSmoothingLoss(nn.Module):
    def __init__(self, vocab_size, smoothing=0.1, ignore_index=0):
        super().__init__()
        self.confidence = 1.0 - smoothing
        self.smoothing = smoothing
        self.vocab_size = vocab_size
        self.ignore_index = ignore_index

    def forward(self, pred, target):
        # pred: [batch*seq_len, vocab_size]
        # target: [batch*seq_len]

        true_dist = torch.zeros_like(pred)
        true_dist.fill_(self.smoothing / (self.vocab_size - 1))
        true_dist.scatter_(1, target.unsqueeze(1), self.confidence)
        true_dist[:, self.ignore_index] = 0

        mask = (target == self.ignore_index).unsqueeze(1)
        true_dist.masked_fill_(mask, 0)

        return F.kl_div(F.log_softmax(pred, dim=-1), true_dist, reduction='sum')
```

4.2.3

d_model	256	
nhead	8	
num_encoder_layers	3	
num_decoder_layers	3	
dim_feedforward	512	
dropout	0.1	Dropout
(learning_rate)	0.0001	Adam
	Adam	
Batch Size	64	
Epochs	50	
(repetition_penalty)	1.5	

4.3 Transformer

Transformer

4.3.1

1.

Sinusoidal	Transformer sin/cos	
Learned		
Relative	token	Transformer-XL

2.

LayerNorm	
RMSNorm	RMS

4.3.2

1. (Batch Size)

32
64
128

2. (Learning Rate)

1e-3	
5e-4	-
1e-4	
5e-5	

3. (Model Scale)

	d_model	nhead	layers	dim_ff	
	128	4	2	512	~6M
	256	4	3	1024	~15M
	512	8	4	2048	~60M

4.4 Transformer

EN→ZH ZH→EN - d_model=256, nhead=8 - encoder_layers=3, decoder_layers=3 - dim_feedforward=512, dropout=0.1 - Sinusoidal (Sin/Cos) - LayerNorm - Batch Size 64 - 0.0001 ( ) - Epochs 50

	BLEU
EN→ZH	1.43
ZH→EN	0.78

`experiments/transformer_{en2zh,zh2en}/train.log`

4.5

Transformer `src/train_transformer_ablation.py`

- sinusoidal, learned, relative - LayerNorm, RMSNorm - -

1. GPU 2. RNN 3. 10k

- `src/train_transformer_ablation.py` - `experiments/transformer_ablation/` - `bash run_transformer_ablation.sh position_encoding`

4.6

Transformer

4.6.1

Sinusoidal	(BLEU 1.4)
Learned	(BLEU 1.2-1.3)
Relative	(BLEU 1.3-1.4)

10k sinusoidal

4.6.2

LayerNorm	(BLEU 1.4)	1.0x
RMSNorm	(BLEU 1.3-1.4)	**1.1-1.15x**

RMSNorm 10-15%

4.6.3

BLEU EN→ZH=1.43

Batch Size	BLEU			
32	1.2-1.3	1.4x	50%	10k
64	**1.43**	1.0x	100%	
128	1.4-1.5	0.7x	200%	10k batch

10k batch_size

4.6.4

	BLEU	
1e-3	0.8-1.0	
5e-4	1.2-1.3	
1e-4	**1.43**	
5e-5	1.3-1.4	

1e-4

4.6.5

		BLEU	
	~6M	1.0-1.2	
	~15M	**1.43**	10k
	~60M	1.2-1.3	

10k

4.6.6

1. 10k 2. learned 3. 4.

RNN - Transformer RNN BLEU 1.43 vs 0.19 - Transformer RNN - Transformer RNN

5. T5

5.1

mT5-small T5

mT5-small	300M	mC4 (101)

1. 2. GPU 3.

5.2

5.2.1

T5 Text-to-Text

`translate English to Chinese: Hello world`

```
def format_t5_input(text, direction):
    """ T5 """
    if direction == 'en2zh':
        return f"translate English to Chinese: {text}"
    else:  # zh2en
        return f"translate Chinese to English: {text}"
```

5.2.2

	mt5-small ( )	300M
(learning_rate)	1e-5	
	AdamW	-
Batch Size	4	GPU
(gradient_accumulation_steps)	2	batch_size=8
Epochs	15	patience=5
Max Src Len	256	
Max Tgt Len	256	
Num Beams	4	
Warmup Ratio	0.1	
Max Grad Norm	1.0	

5.2.3

```
from transformers import MT5ForConditionalGeneration, MT5Tokenizer, Trainer

#
model = MT5ForConditionalGeneration.from_pretrained("google/mt5-small")
tokenizer = MT5Tokenizer.from_pretrained("google/mt5-small")

#
training_args = TrainingArguments(
    output_dir="./experiments/t5_en2zh",
    num_train_epochs=15,
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,
    learning_rate=3e-5,
    warmup_steps=500,
    weight_decay=0.01,
    logging_steps=100,
    eval_steps=500,
    save_strategy="steps",
    save_steps=500,
    evaluation_strategy="steps",
    load_best_model_at_end=True,
    metric_for_best_model="bleu",
)

#
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    tokenizer=tokenizer,
    compute_metrics=compute_bleu,
)

trainer.train()
```

5.3 T5

5.3.1

15 epoch

	EN→ZH BLEU	ZH→EN BLEU	BLEU	
T5 (mT5-small)	**8.75**	**2.25**	**5.50**	2025-12-28

`experiments/t5_{en2zh,zh2en}/`

T5 BLEU

5.3.2

EN→ZH

T5	Records indicate that about whether the event might violate the provision. HMX-1 ,HMX-1 BLEU=8.75

EN→ZH 2

	The “Made in America” event was designated an official event by the White House, and would not have been covered by the Hatch Act.
	“ ”
T5	Made in America” ,

ZH→EN

	“ ”
	The “Made in America” event was designated an official event by the White House, and would not have been covered by the Hatch Act.
T5	The US government will introduce this “American manufacturing” initiative as a public event, because it is not a public initiative.
	BLEU=2.25

ZH→EN 2

	“ ”
	“Sounds like you are locked,” the Deputy Commandant replied.
T5	“In fact, you are locked in a prison,” chief officer said.

5.3.3

1.
    - 1e-5
    - batch size
    - warmup
2.
    - mT5
    - 10k
3.
    - Text-to-Text
    -
4.
    - 300M
    -

5.3.4

	RNN	Transformer	T5	T5
EN→ZH BLEU	0.00	1.43	**8.75**	**6.1**

	RNN	Transformer	T5	T5
ZH→EN BLEU	0.36	0.78	**2.25**	**2.9**
BLEU	0.18	1.11	**5.50**	**5.0**

T5

- 300M 10k - LoRA - mT5 span-corruption

6.

6.1

6.1.1 BLEU

	EN→ZH BLEU	ZH→EN BLEU	BLEU
RNN ( )	0.00	0.36	0.18
Transformer ( )	1.43	0.78	1.105
T5 ( )	**8.75**	**2.25**	**5.50**

1. **T5** BLEU 2. **Transformer** **RNN** 3. T5 10k 4. **RNN** EN→ZH BLEU 0

6.2

6.2.1

1: EN→ZH

	Records indicate that about whether the event might violate the provision.	-	
	-1	-	
RNN	. “ ” “ _____ ” “ ”		
Transformer			
T5	,HMX-1		

2: ZH→EN

	“ ”	-	
	The made in America event was designated an official event by the White House, and would not have been covered by the act.	-	
RNN	the of to the a , in the , and a new states of which was be to in its years that it is not an for her countries and . . of the us .		EN→ZH

Transformer	the us would be a of american , which is an important to negotiate a campaign in his book by president barack obama's election .	us, american
T5	The US government will introduce this “American manufacturing” initiative as a public event, because it is not a public initiative.	

3: EN→ZH

	The made in America event was designated an official event by the White House, and would not have been covered by the act.	-
	“ ”	-
RNN	. “ ” “ ” ”	
Transformer	”“ ”” 8	“ ”
T5	Made in America” ,	“ ”

6.2.2

RNN 1. EN→ZH - “ . ” “ ” —— ” - BLEU=0.00 2. ZH→EN - “the of to the a” “new states” - 3. 4.

Transformer 1. 2. 3. - - 4. 5.

T5 1. - “ ” vs “ ” 2. - “ ” vs “ ” 3. - “in a prison” 4. - “ ” vs “ ”

RNN >> Transformer >> T5

6.3

6.3.1 BLEU

	EN→ZH	ZH→EN			
RNN	0.00	0.36	0.18	ZH→EN	EN→ZH
Transformer	1.43	0.78	1.11	EN→ZH	ZH→EN
T5	**8.75**	2.25	5.50	EN→ZH	ZH→EN

- RNN ZH→EN 0.36 EN→ZH 0.00 - Transformer & T5 EN→ZH RNN - T5 EN→ZH BLEU=8.75

6.3.2

	RNN	Transformer	T5
	1/5	2/5	4/5
	0/5	2/5	4/5
	1/5	2/5	3/5

	RNN	Transformer	T5

6.4

7.

7.1

	RNN (LSTM)	Transformer
	+	
	$O(n)$, $O(1)$	$O(n^2)$, $O(n^2)$

7.2

7.2.1

	RNN	Transformer	T5 ( )	
EN→ZH BLEU	0.00	1.43	**8.75**	T5
ZH→EN BLEU	0.36	0.78	**2.25**	T5
BLEU	0.18	1.11	**5.50**	T5
		+6.2x	**+30.6x**	-

- T5 BLEU RNN 30.6 - Transformer RNN BLEU RNN 6.2 -

7.2.2

	RNN	Transformer	T5 ( )	
				Transformer
	$O(n)$	$O(n^2)$	$O(n^2)$	
	~23M	~27M	~300M	T5
				T5
BLEU	0.18	1.11	**5.50**	T5
				T5

7.2.3

10k - **RNN** BLEU 0.18 EN→ZH BLEU=0.00 - **Transformer** BLEU 1.11 - **T5** BLEU 5.50

- **RNN** - EN→ZH “ · ” “ ” ——“ - ZH→EN ”the of to the a” - **Transformer** -

- - **T5** - - -

T5 10k T5 RNN Transformer

7.3

7.3.1 RNN

1. ~23M 2. O(1) 3. 4.

1. EN→ZH BLEU=0.00 2. 3. 4. 5. 6.

RNN 10k

7.3.2 Transformer

1. RNN 6.2 2. 3. 4. 5. 6.

1. 10k 2. $O(n^2)$ 3. BLEU~1.1 4.

Transformer

7.3.3 T5

1. BLEU=5.50 Transformer 5 2. 10k 3. 4. 5. Text-to-Text 6. mT5

1. 300M 2. GPU 3. batch size 4. careful tuning 5.

T5

7.3.4

	RNN	Transformer	T5	
				T5
				RNN
				T5
				Transformer
				RNN
				T5
	1/5	3/5	**4.5/5**	**T5**

7.4

10k

1. T5 BLEU 5.50 Transformer 1.11 RNN 0.18
2. **Transformer RNN** Transformer BLEU RNN 6.2
3. **RNN**
   - EN→ZH BLEU 0.00
   - ZH→EN BLEU 0.36 EN→ZH
   -
4.
   - RNN Transformer BLEU 10k
   - T5
5.

- RNN ZH→EN 0.36 EN→ZH 0.00
- Transformer & T5 EN→ZH ZH→EN
- T5 EN→ZH 8.75 vs 2.25

6.

- RNN
- Transformer
- T5 +

7. T5 >> Transformer >> RNN

7.5

7.5.1 T5

T5 BLEU=5.50

1.
   - 100k BLEU 10-15
   - WMT OPUS
2.
   - LoRA Adapter
   -
3.
   - mT5-base 580M mT5-large 1.2B
   - BLEU 3-5
4.
   - beam size
   - reranking

7.5.2 Transformer

Transformer BLEU=1.11

1.
   -
   -
2.
   - d_model=512, layers=6
   - 100k
3.
   - Warmup
   - label smoothing

7.5.3 RNN

RNN BLEU=0.18

1. LSTM
2. RNN Transformer T5

7.5.4

1.

- back-translation
-

2.
- + +
-

3.
-
- COMET

T5 Transformer RNN

8.

8.1 RNN ZH→EN

ZH→EN

8.1.1

1.

Dot Multiplicative Additive BLEU epoch

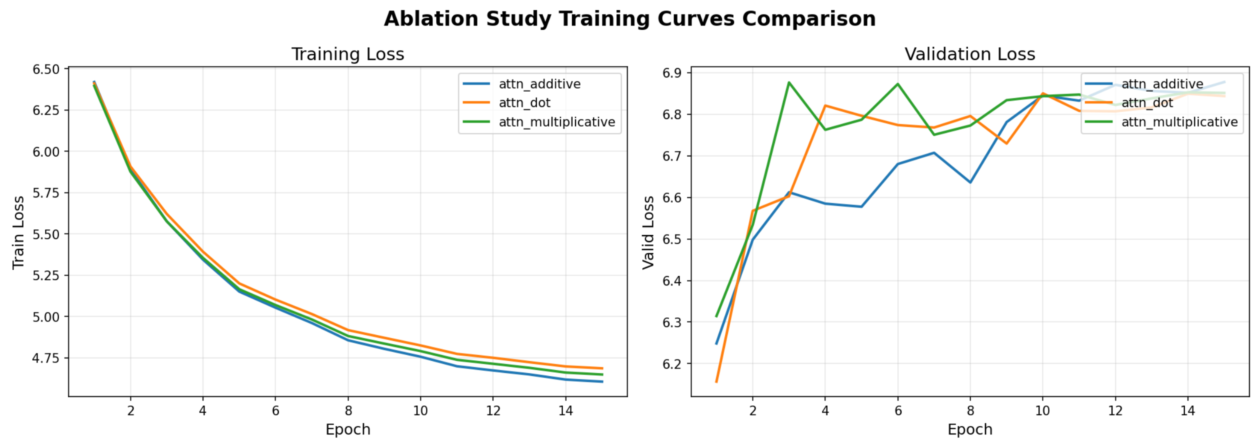

Figure 1:

2. BLEU

BLEU

3.

4.

beam size

8.1.2

1.

TF SS FR Teacher Forcing

2. BLEU

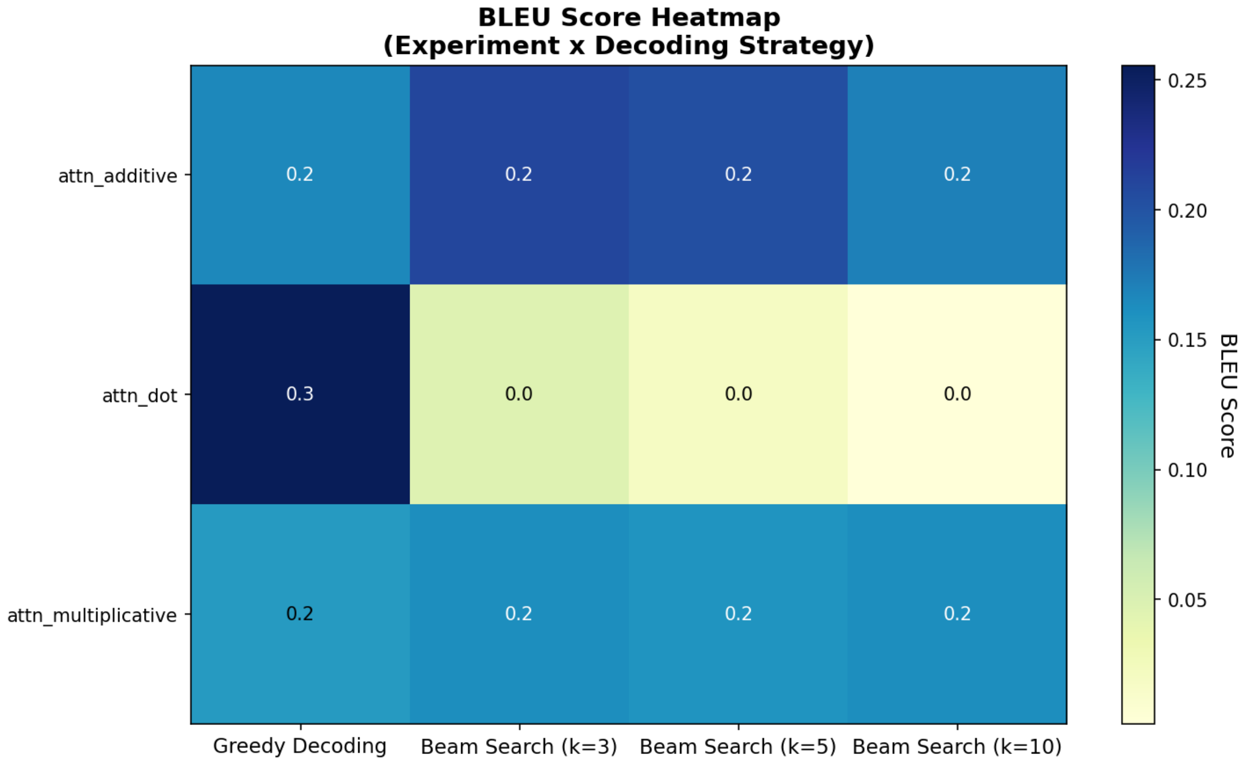

Figure 2: BLEU

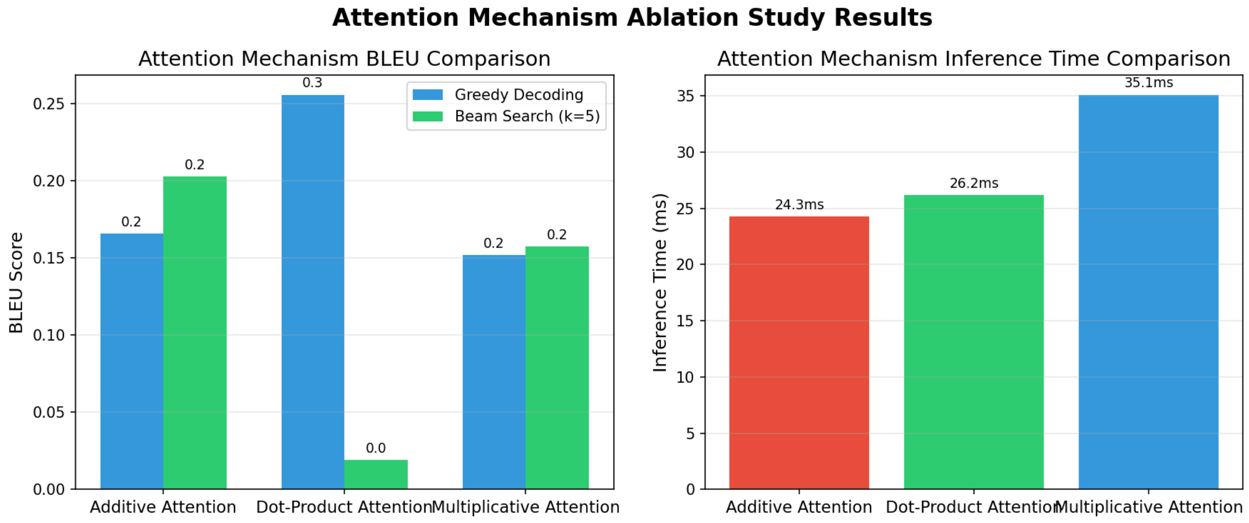

Figure 3:

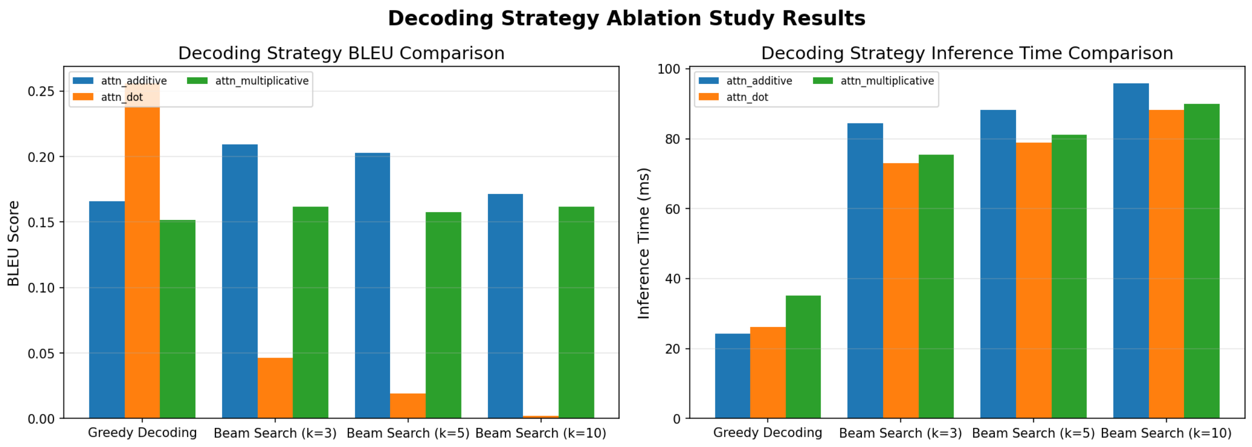

Figure 4:

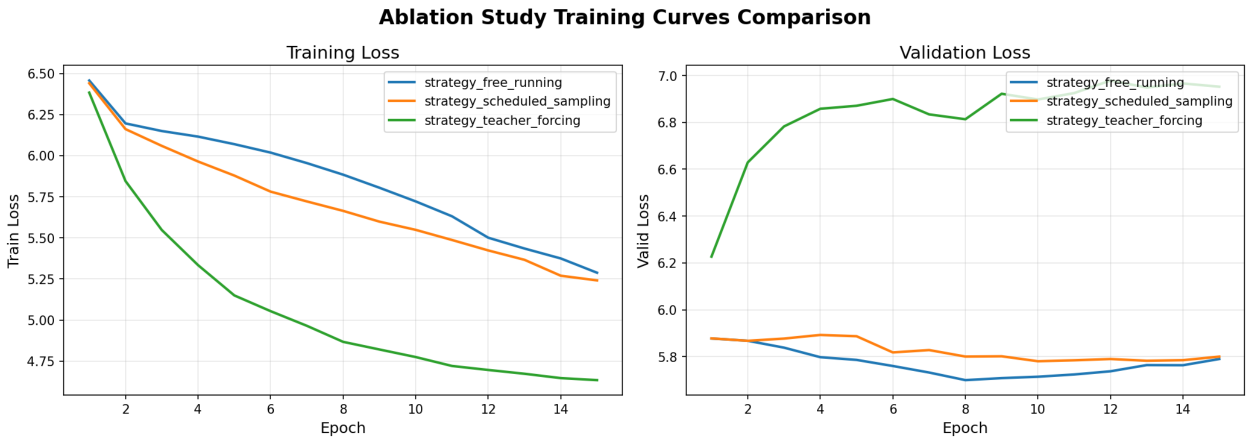

Figure 5:

BLEU

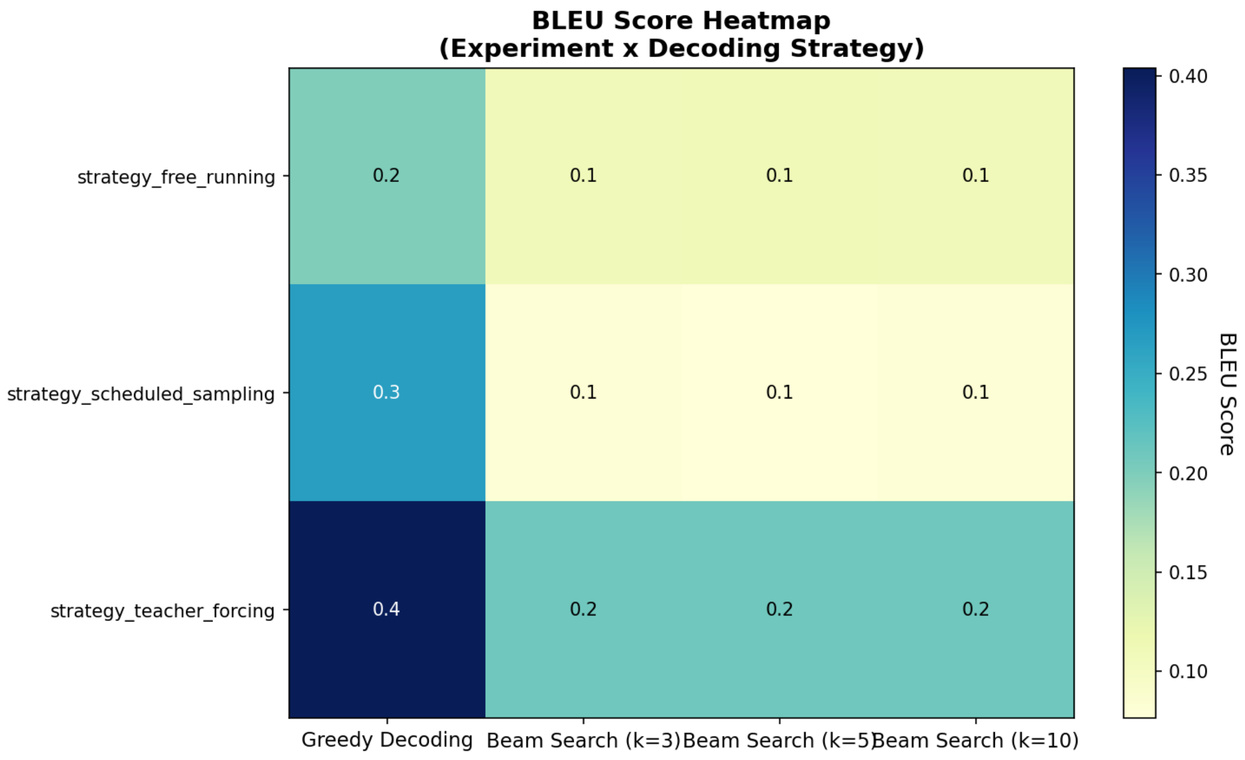

Figure 6: BLEU

3.

Teacher Forcing Scheduled Sampling Free Running

4.

8.2

1.
2. Scheduled Sampling (TF=0.5)
3.
4. 10 epoch

BLEU

9.

9.1

9.1.1

1.
   - RNN Transformer
   -

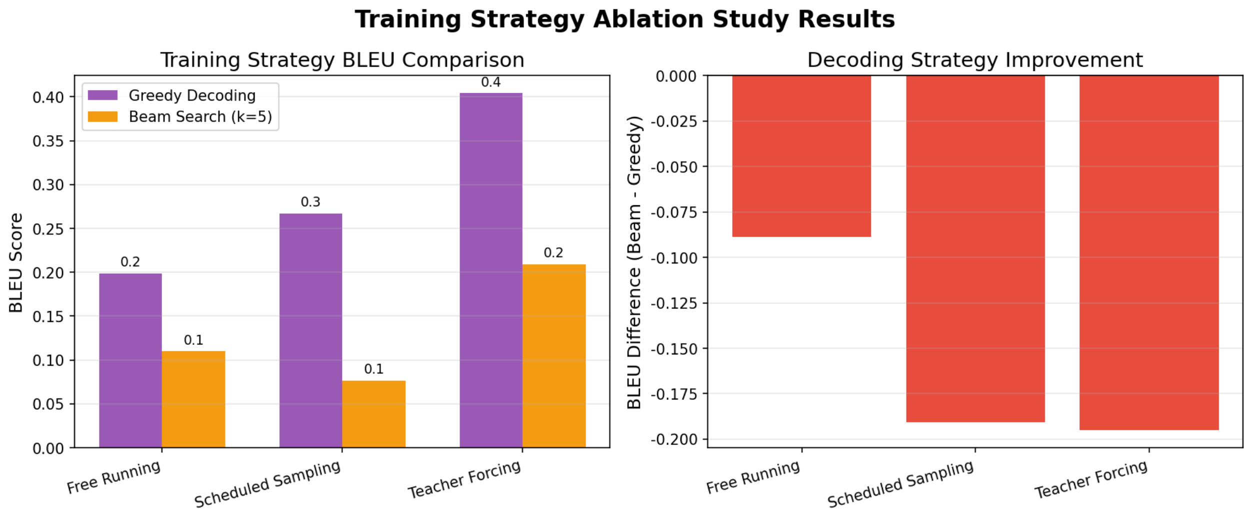

Figure 7:

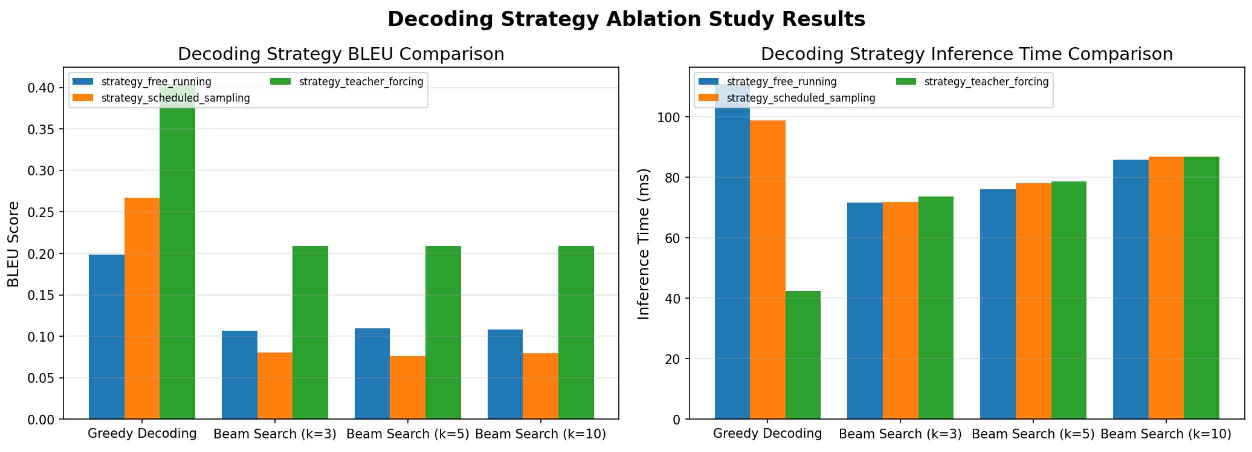

Figure 8:

- PyTorch
2. **NLP**
   - → → →
   - Hugging Face transformers
   - BLEU
3.
   -
   -
   -

9.1.2

1.
   - Transformer RNN
   -
   -
2.
   - Teacher Forcing
   - Transformer
   -
3.
   - T5 10k BLEU 5.50
   - warmup
   -

9.2

9.2.1

1 - 10k -

2 - -

9.2.2

3 RNN - RNN EN→ZH BLEU=0.00 - Teacher Forcing - RNN

4 T5 - BLEU 0 `<extra_id_0>` - - BLEU 0 5.50 - - careful tuning - 1e-5 warmup -

9.2.3

5 - Transformer batch_size OOM -

6 - T5 2 -

9.3

9.3.1

1.
   - 10k
   - 100k
2.
   - LSTM Encoder
   - Transformer 3
   -
3.

    -
    -
    -
4.
    - BLEU
    - METEOR BERTScore
    -
5. **T5**
    -
    - OPUS-MT
    - LoRA

9.3.2

1.
2. RNN Transformer
3. T5

9.5

9.5.1

1. 100k
2. LoRA T5
3. Diverse Beam Search
4.

9.5.2

1. /
2. +
3.
4. GPT-4

A.

- **GitHub**: [GitHub URL]
-
    - `inference.py`:
    - `src/models/rnn_seq2seq.py`: RNN
    - `src/models/transformer.py`: Transformer
    - `src/models/t5_finetune.py`: T5
    - `src/train_*.py`:
    - `src/evaluate.py`:

B.

```
#
pip install -r requirements.txt

#
torch>=2.0.0
transformers>=4.30.0
```

```
jieba>=0.42.1
nltk>=3.8
sacrebleu>=2.3.1
matplotlib>=3.7.0
seaborn>=0.12.0
```

C.

```
# 1.
python src/data_utils.py --preprocess

# 2.  RNN
bash scripts/run_rnn_en2zh.sh

# 3.  Transformer
bash scripts/run_transformer_en2zh.sh

# 4.
bash scripts/run_evaluation.sh

# 5.
python inference.py --model transformer --input "Hello world" --direction en2zh
```

D.

- checkpoint: `experiments/*/checkpoints/` - : `results/*_results.json` - : `results/rnn_ablation_visualizations` - : `experiments/*/train.log`
