NLP：lab2

序号	学号	专业班级	姓名	性别
/	3220102157	图灵2201	张祎迪	女

1.Project Introduction

Development Environment:

- ModelArts Ascend Notebook： mindspore1.7.0-cann5.1.0-py3.7-euler2.8.3

SeqtoSeq with Transformers

Sequence-to-Sequence (Seq2Seq) Model

- **Seq2Seq Model**: A type of neural network architecture used for sequence prediction tasks, such as machine translation, text summarization, and speech recognition.
- **Encoder-Decoder Architecture**: Consists of an encoder network that processes the input sequence and a decoder network that generates the output sequence.
- **Attention Mechanism**: Allows the model to focus on different parts of the input sequence during the decoding process.

This project aims to implement a Seq2Seq model using the Transformer architecture for machine translation tasks. The Transformer model is a state-of-the-art architecture that has been widely adopted for sequence-to-sequence tasks due to its parallelism and scalability.

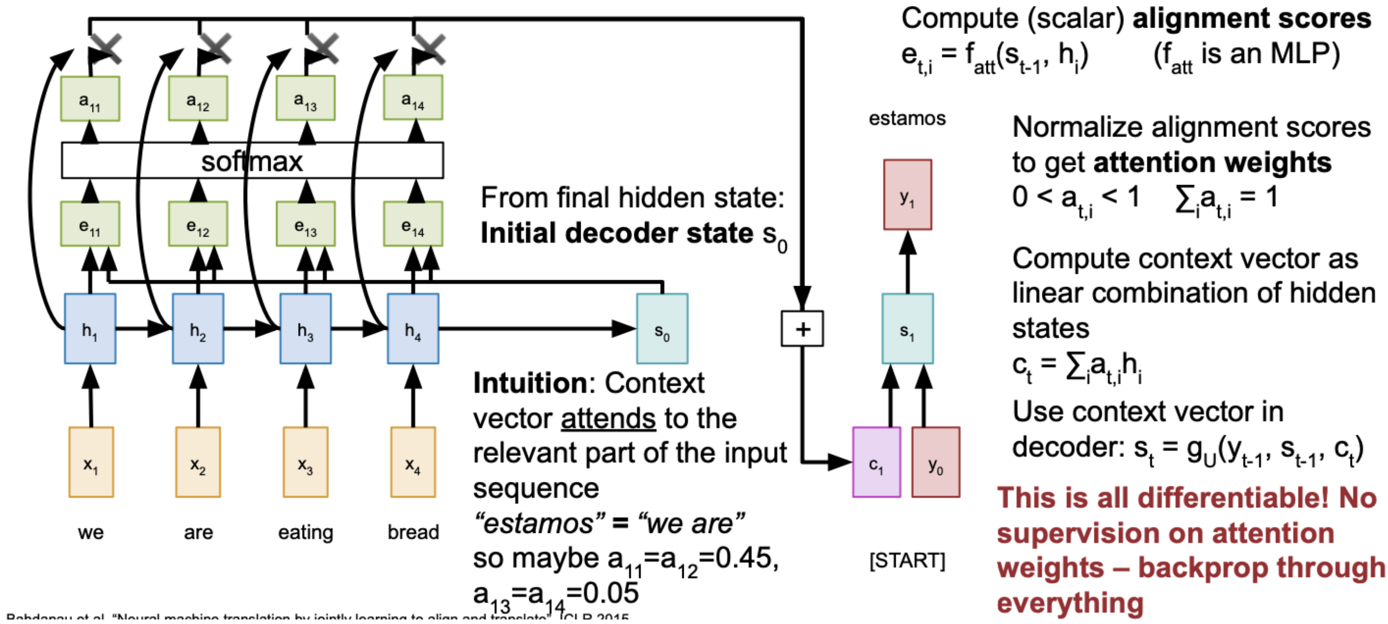
Steps:

1. **Data Preprocessing**: Prepare the training and validation datasets for the machine translation task.
2. **Model Architecture**: Implement the Transformer model architecture for sequence-to-sequence tasks.
3. **Training**: Train the model on the training dataset and evaluate its performance on the validation dataset.
4. **Inference**: Use the trained model to generate translations for new input sequences.

2.Technical Details

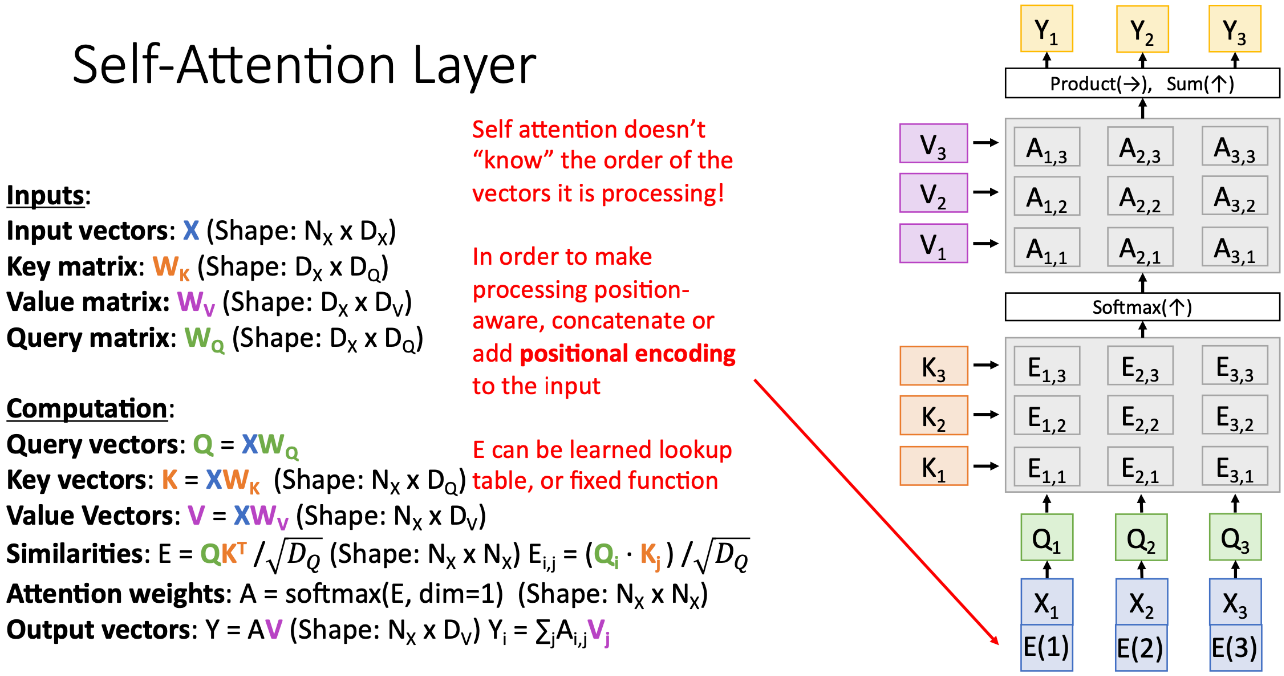
Theoretically Elaboration

Attention Mechanism
Basic Idea: The attention mechanism allows the model to focus on different parts of the input sequence during the decoding process. It assigns weights to the input elements based on their relevance to the current output element, enabling the model to capture long-range dependencies and improve the quality of the generated sequences.

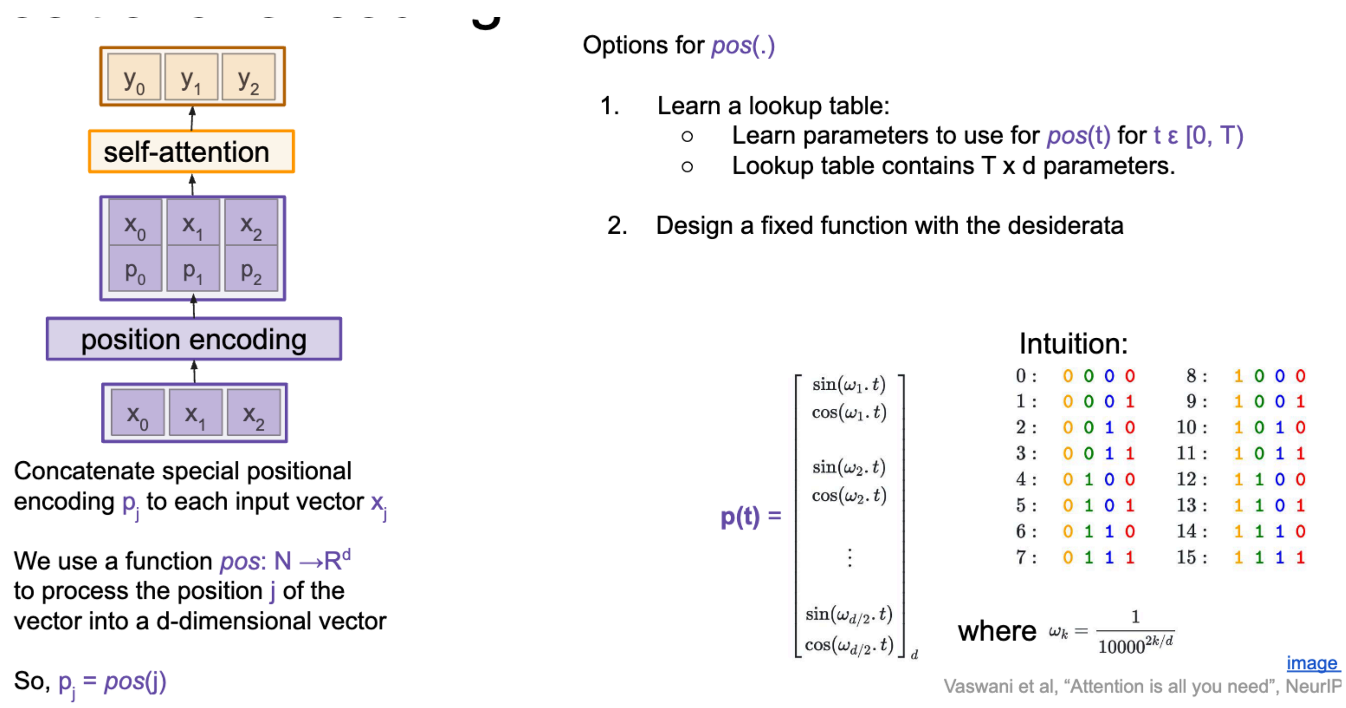


From the above figure, we can see that each s_i is computed based on the weighted sum of the input elements $h_1, h_2, \ldots, h_n$, where the weights are determined by the attention scores a_{ij}.

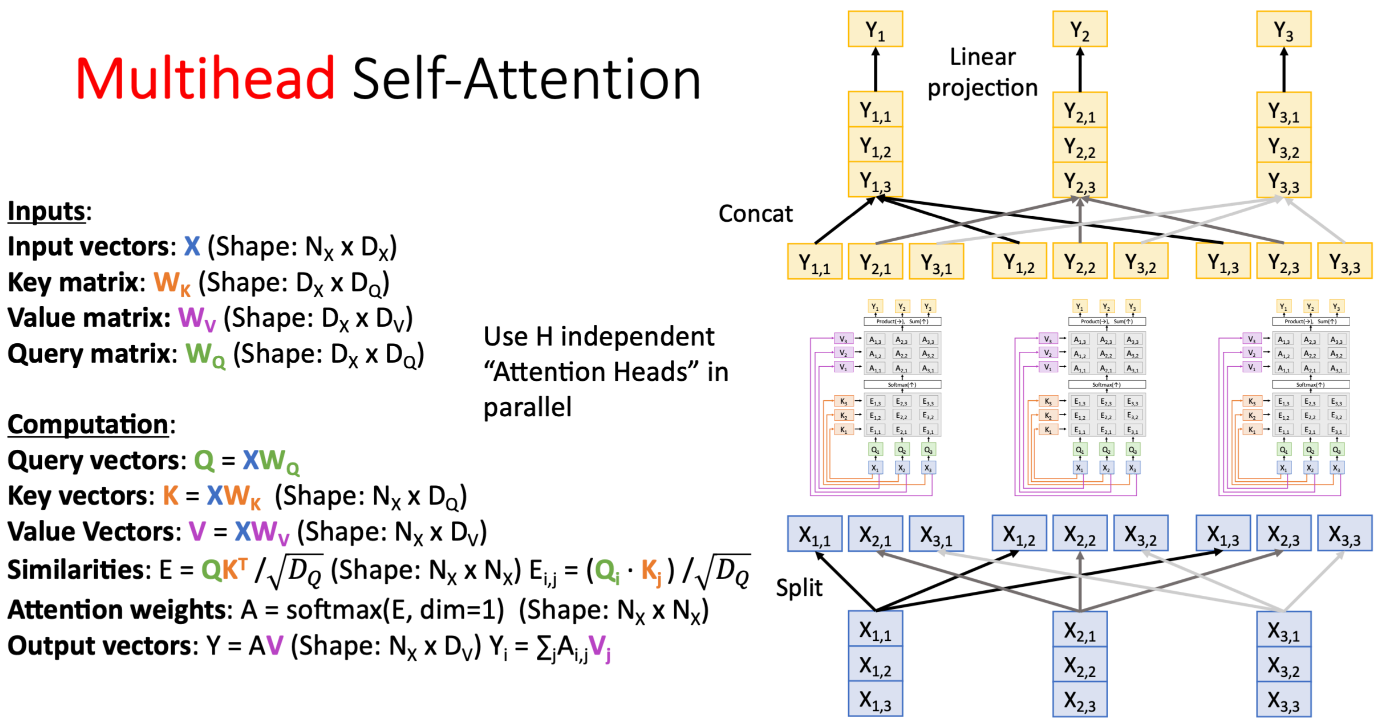
Improved Attention Mechanisms



- **Self-Attention**: A mechanism that allows the model to focus on different parts of the input sequence during the decoding process.
- **Scaled Dot-Product Attention**: A type of self-attention mechanism that scales the dot-product attention scores by the square root of the dimensionality of the key vectors.

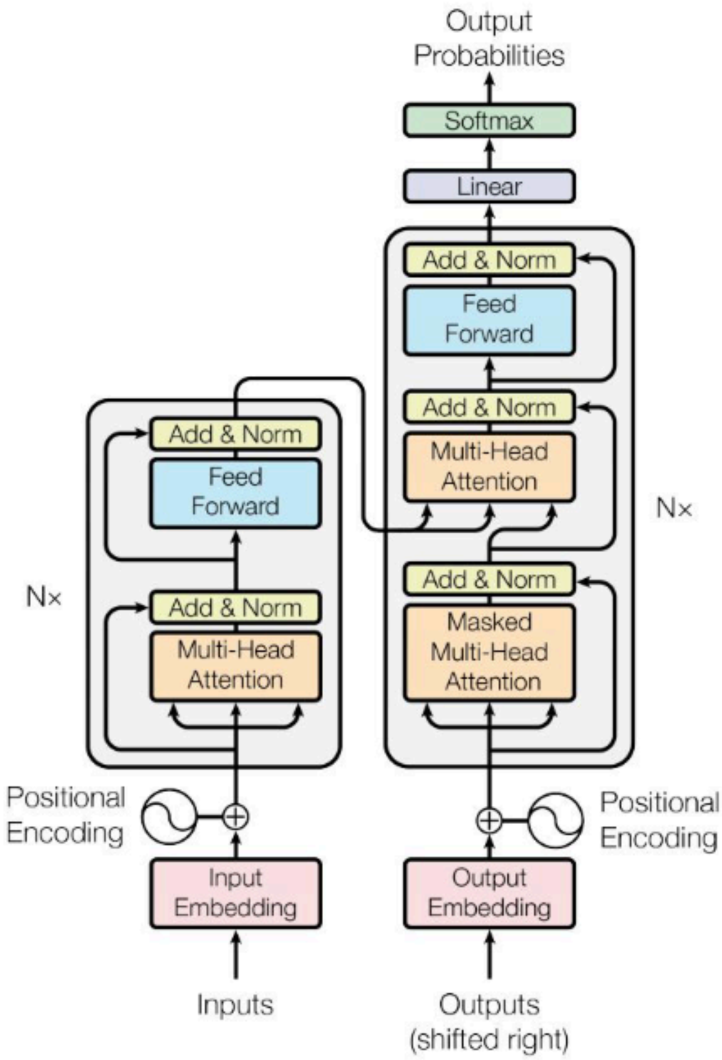


- **Positional Encoding**: A technique used to inject positional information into the input embeddings to capture the sequential order of the input sequence.

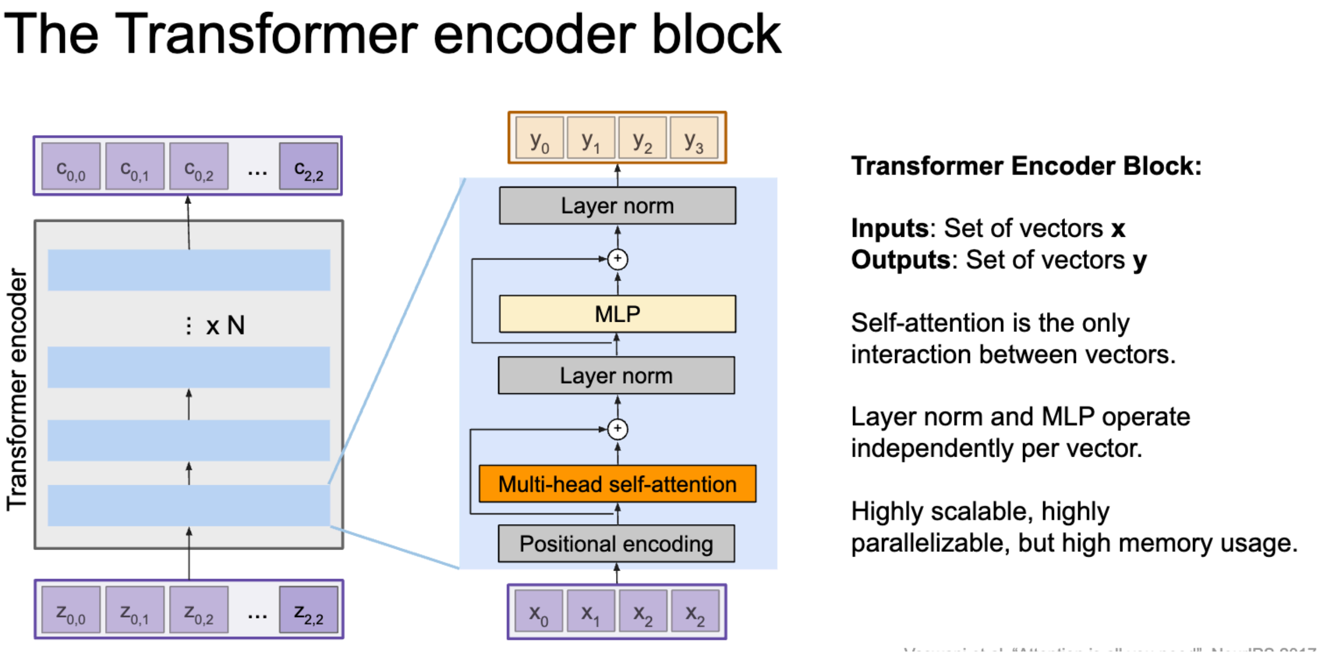


- **Multi-Head Attention**: A mechanism that allows the model to jointly attend to information from different representation subspaces.

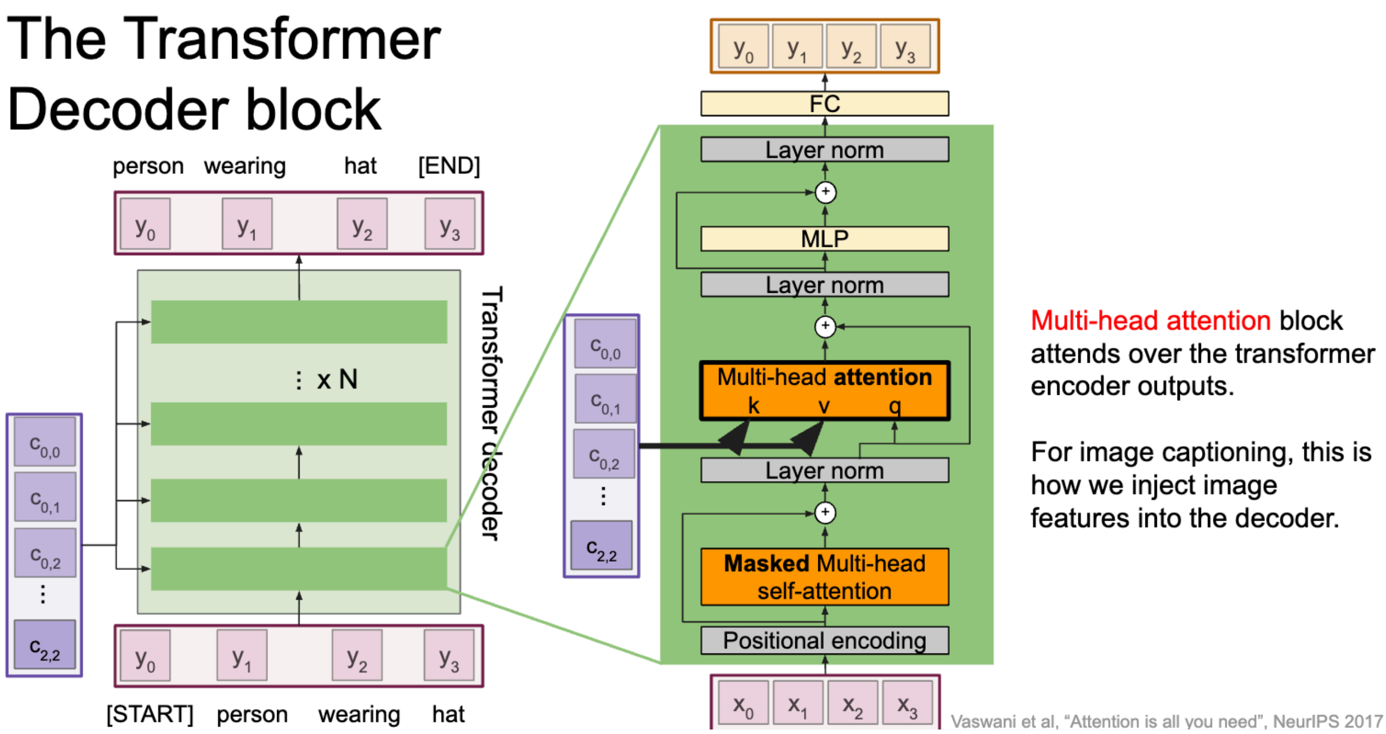
The Transformer Architecture



- **Encoder**: Processes the input sequence and generates a sequence of hidden representations.



- **Decoder**: Generates the output sequence based on the encoder's hidden representations and the previous output elements.



Algorithm Implementation

I did not implement the algorithm myself in the project, but I used the MindSpore library to build the Transformer model for machine translation tasks. The MindSpore library provides a high-level API for building and training deep learning models, making it easy to implement complex architectures such as the Transformer model.

1. **Data Preprocessing**

```
sample_num = 23607
eval_idx = np.random.choice(sample_num, int(sample_num*0.2), replace=False)
data_prepare(data_cfg, eval_idx)
```

> Choose 20% of the data as training set

The `data_prepare` function preprocesses input data, tokenizes it, filters instances based on length constraints, and writes the preprocessed data to separate files for training and evaluation in a format suitable for training a transformer-based model.

2. **Training**

- **Main part of the parameters** (in the `train_cfg` dictionary):

```
'''
two kinds of transformer model version
'''
if train_cfg.transformer_network == 'base':
    transformer_net_cfg = TransformerConfig(
        batch_size=train_cfg.batch_size,
        seq_length=train_cfg.seq_length,
        vocab_size=train_cfg.vocab_size,
        hidden_size=512,
        num_hidden_layers=6,
        num_attention_heads=8,
        intermediate_size=2048,
        hidden_act="relu",
        hidden_dropout_prob=0.2,
        attention_probs_dropout_prob=0.2,
        max_position_embeddings=train_cfg.max_position_embeddings,
        initializer_range=0.02,
        label_smoothing=0.1,
        input_mask_from_dataset=True,
        dtype=mstype.float32,
        compute_type=mstype.float16)
elif train_cfg.transformer_network == 'large':
    transformer_net_cfg = TransformerConfig(
        batch_size=train_cfg.batch_size,
        seq_length=train_cfg.seq_length,
        vocab_size=train_cfg.vocab_size,
        hidden_size=1024,
        num_hidden_layers=6,
        num_attention_heads=16,
        intermediate_size=4096,
        hidden_act="relu",
        hidden_dropout_prob=0.2,
        attention_probs_dropout_prob=0.2,
        max_position_embeddings=train_cfg.max_position_embeddings,
        initializer_range=0.02,
        label_smoothing=0.1,
        input_mask_from_dataset=True,
        dtype=mstype.float32,
        compute_type=mstype.float16)
else:
    raise Exception("The src/train_config of transformer_network must base or large. Change the str/train_config file and try ag
```

Specifications

`batch_size`: The number of instances in each batch.

`seq_length`: The maximum sequence length for input and output sequences.

`vocab_size`: The size of the vocabulary used for tokenization.

`hidden_size:` The size of the hidden layers in the Transformer model.

`num_hidden_layers`: The number of hidden layers in the Transformer model.

`num_attention_heads`: The number of attention heads in the Transformer model.

`hidden_act`: The activation function used in the hidden layers.

`hidden_dropout_prob`: The dropout probability for the hidden layers.

`attention_probs_dropout_prob`: The dropout probability for the attention probabilities.

`max_position_embeddings`: The maximum number of positions for positional embeddings.

`initializer_range`: The range for random weight initialization.

`label_smoothing:` The label smoothing factor for the loss function.

`input_mask_from_dataset:` Whether to use input masks from the dataset.

```
train(train_cfg)
```

The `train()` function trains the model. It first loads the dataset and initializes the Transformer network, loss function, learning rate, and optimization method. If pre-trained model parameters exist, it loads the parameters from the specified cfg.checkpoint_path path into the network. If model saving is enabled in the configuration, it also adds a ModelCheckpoint callback function to save the model. Additionally, it adjusts the optimizer based on whether enable_lossscale is enabled. Finally, it wraps everything into a model using the Model class, sets it to training mode, and calls the `train()` method to start training.

3. **Evaluation**

```
evaluate(eval_cfg)
```

- Specific parameters in the `eval_cfg` dictionary
- More details can be found in the `mt_transformer_mindspore.ipynb` file.

3.Experiment Results

- All test results can be found in the `mt_transformer_mindspore.ipynb` file.

```
...
result: 他 有 可 能 再 迟 到 了 。
source: He is lying through his teeth . 他 明 显 在 撒 谎 。
result: 他 穿 过 衣 服 的 牙 齿 穿 衣 服 在 牙 齿 。
source: He painted a picture of roses . 他 画 了 一 幅 玫 瑰 的 画 。
result: 他 画 了 一 幅 画 的 照 片 。
source: He put on an air of innocence . 他 摆 出 一 副 无 辜 的 样 子 。
result: 他 在 空 空 空 空 空 空 空 空 气 上 发 现 了 空 气 。
source: He studies history at college . 他 在 大 学 修 读 历 史 。
result: 他 在 大 学 校 学 的 历 史 学 习 历 史 上 学 习 历 史 书 。
source: He threw a rock into the pond . 他 扔 一 块 石 头 到 池 塘 里 。
result: 他 扔 了 一 块 岩 石 头 后 扔 了 。
source: He was busy with his homework . 他 忙 于 做 功 课 。
result: 他 忙 于 做 他 的 家 庭 作 业 后 做 作 业 。
source: He was hurt in a car accident . 他 在 一 次 车 祸 中 受 伤 了 。
result: 他 在 一 场 车 祸 中 发 生 了 意 外 的 意 外 面 。
source: He ' ll be back by five o ' clock .    他 五 点 左 右 会 回 来 。
result: 他 五 点 之 前 会 五 点 回 来 。
source: He ' s a friend of my brother ' s .    他 是 我 哥 哥 的 朋 友 。
result: 他 是 我 弟 弟 弟 弟 弟 弟 的 一 个 朋 友 。
source: His wife is one of my friends . 他 的 妻 子 是 我 的 一 个 朋 友 。
result: 他 的 妻 子 是 我 的 一 个 朋 友 都 是 我 的 朋 友 。
source: How about going to the movies ? 我 们 去 电 影 院 怎 么 样 ?
result: 去 看 电 影 怎 么 样 ?
source: How are you going to get home ? 你 打 算 怎 么 回 家 ?
result: 你 要 怎 么 去 家 里 ?
source: How much is this handkerchief ? 请 问 这 个 手 帕 多 少 钱 ?
result: 今 天 的 手 帕 怎 么 样 ?
source: I am interested in this story . 我 对 这 个 故 事 感 兴 趣 。
```

- Some typical results are shown below:

> Lack in learning slang

```
source: He is lying through his teeth . 他 明 显 在 撒 谎 。
result: 他 穿 过 衣 服 的 牙 齿 穿 衣 服 在 牙 齿 。

source: He put on an air of innocence . 他 摆 出 一 副 无 辜 的 样 子 。
result: 他 在 空 空 空 空 空 空 空 空 气 上 发 现 了 空 气 。
```

> Lack in caputure features of different languages.
> For example, Chinese tends to use less words to express the same meaning than English, but the model tends to generate the same length of sentences, so there is many repeated words in the generated sentences.

```
source: He ' s a friend of my brother ' s . 他 是 我 哥 哥 的 朋 友 。
result: 他 是 我 弟 弟 弟 弟 弟 弟 的 一 个 朋 友 。
```

> Redundant words : Hard time finding end signal

```
source: You may injure yourself if you don ' t follow safety procedures . 如 果 你 不 按 照 安 全 手 续 来 的 话 ， 你 可 能 会 受 伤 的 。
result: 如 果 你 分 之 前 ， 你 不 能 为 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 止 不 能 不 能

source: This plane flies between Osaka and Hakodate . 这 架 飞 机 往 返 于 大 阪 和 函 馆 之 间 。
result: 这 架 飞 机 和 大 之 间 之 间 之 间 之 间 之 间 之 间 飞 机 之 间 之 间 之 间 之 前 坐 飞 机 之 间 之 间 之 间 之 间 之
```

References

- https://www.mindspore.cn/