NLP（自然语言处理）

1、课程大纲

备注：都是用来处理时序相关的序列模型，比如：文本分类，摘要生成，机器翻译，语音识别，视频识别

- RNN
- LSTM
- GRU
- BiLSTM

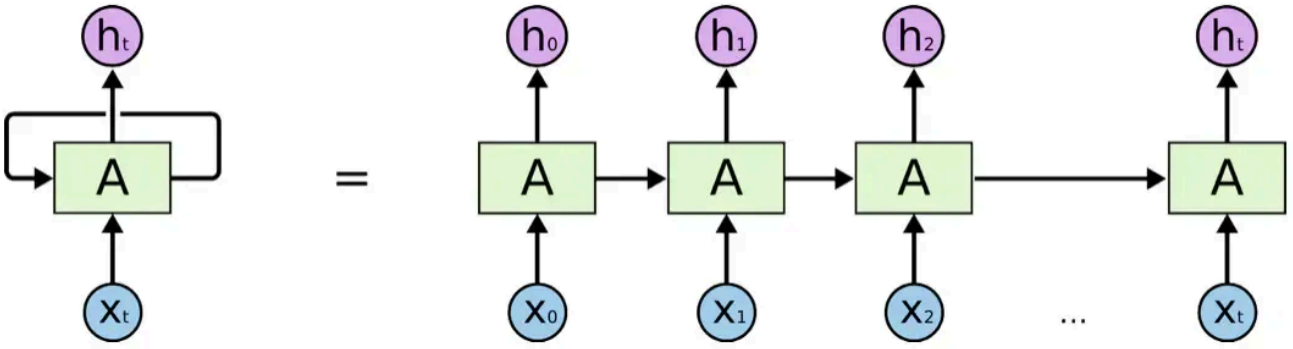
==备注：前馈神经网络（CNN）与 RNN网络的区别？==

- 应用场景：CNN主要应用在图像识别，目标检测，它的核心是核心是**卷积层**，用来实现特征提取的，结合池化，全连接形成前馈神经网络

  而RNN主要用来处理时序序列模型，比如：文本分析/分类/生成，自然语言处理等这种长时序相关的**序列数据**，核心是**循环结构**
- 网络结构：前馈神经网络它的输入只有一个，而RNN循环神经网络它的每个时间步都有2个输入，一个是当前时刻的输入，第二个是上一个时间步的输出作为下一个时间步的输入

2、网络结构

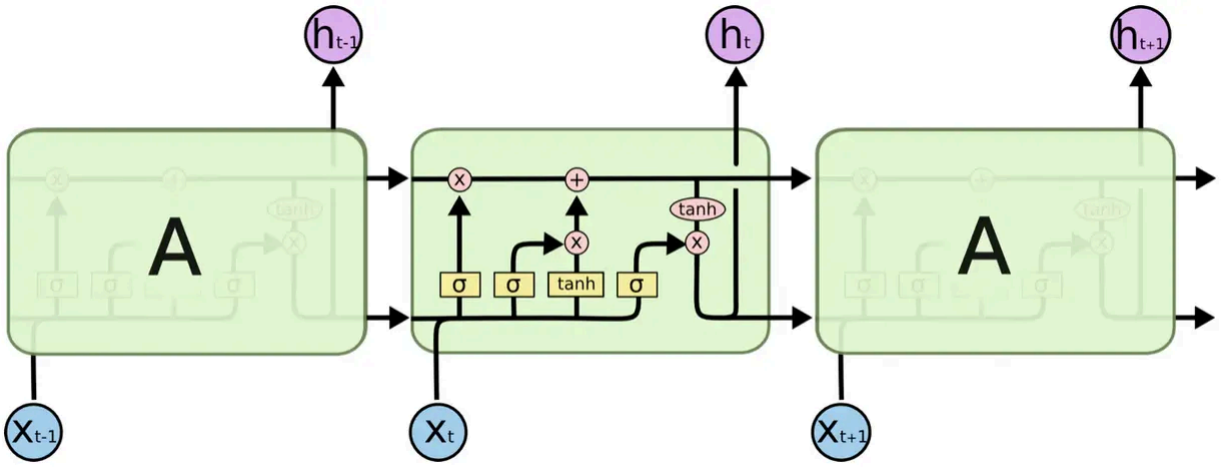
2.1、RNN



==缺点：==

- 当时间序列长度过长的时候，会出现记忆退化
- 存在梯度消失，梯度爆炸的问题

1.2、LSTM

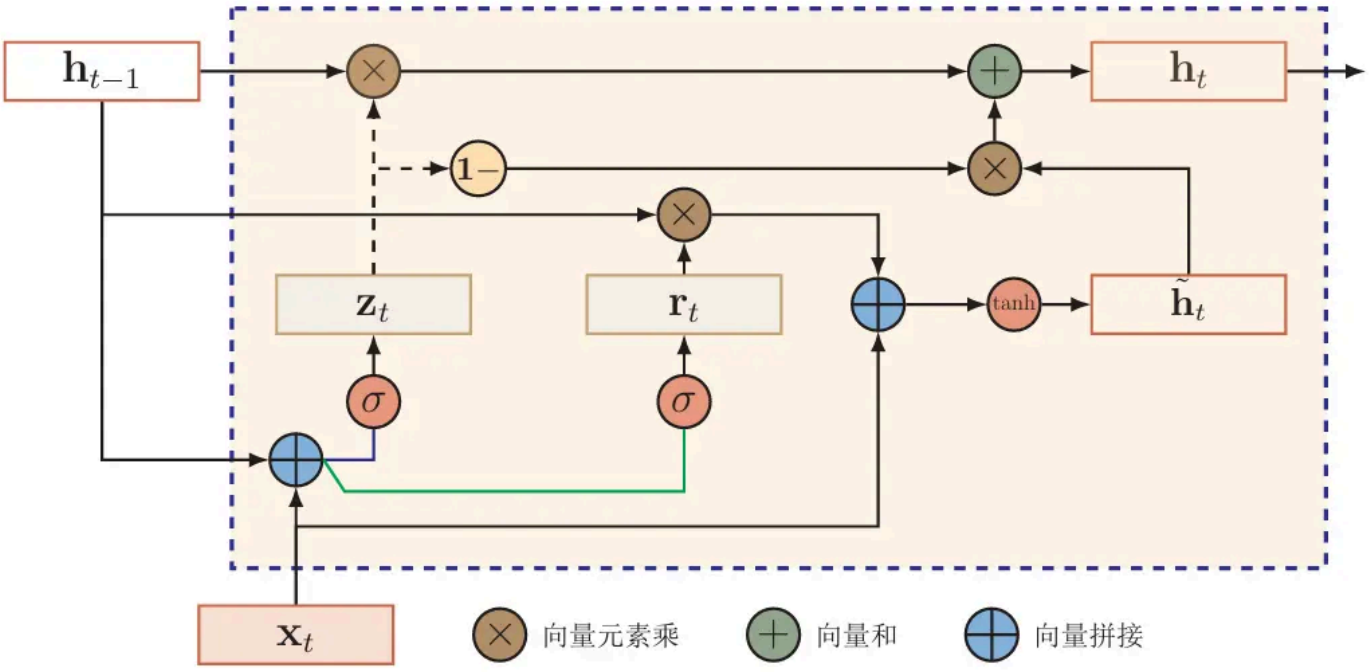


核心：3个门（输入门，输出门，遗忘门）+ 1个细胞更新单元

特点：相对于RNN网络记忆时间更长久，从某种程度上能解决那种长时序的依赖问题，相对RNN来讲，它能处理更长的时序序列数据，缓解了RNN网络的梯度消息，梯度爆炸的问题。

为什么能解决：引入记忆单元与门控机制，能够选择性地保存或丢弃信息

1.3、GRU

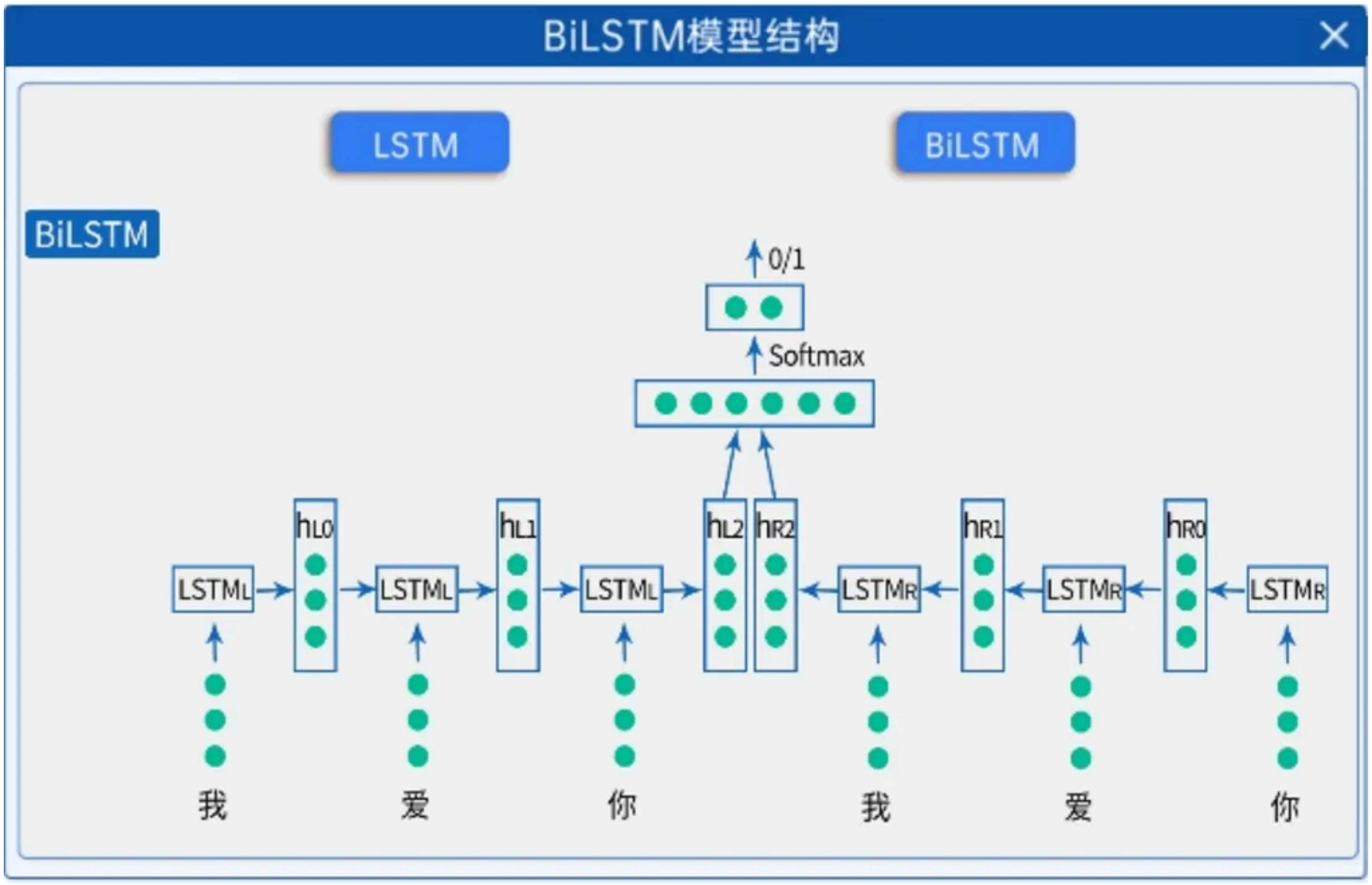


核心：2个门组成：更新门，重置门

重置门作用：决定如何将过去的信息与当前输入相结合，换句话讲就是==重置门决定了保留或遗忘上一时间步信息的程度。==

更新门作用：决定如何结合当前输入和更新后的上一时间步信息来计算新的隐藏状态。

1.4、BiLSTM的结构



2、面试问题

1、简单给我介绍下RNN网络

1. RNN 通过循环结构处理序列数据，每一时刻的隐藏状态依赖于当前输入和前一时刻的隐藏状态，能够捕捉序列中的时序依赖关系。
2. 相对CNN来讲，适合处理变长序列数据，如文本、时间序列。结构简单，易于实现。
3. 不过，它存在梯度消失和梯度爆炸问题，难以捕捉长距离依赖。
4. 训练效率较低，对长序列建模能力有限。
5. 不过后期LSTM 和 GRU 通过引入门控机制，解决了 RNN 的梯度问题，提升了长序列建模能力。

2、简单给我介绍下LSTM网络

1. LSTM叫长短期记忆网络
2. 例外，LSTM主要由3个门控结构(输入门,输出门,遗忘门)和一个细胞更新单元组成
3. 它核心机制就是通过引入细胞状态和门控结构（输入门、遗忘门、输出门），解决了传统 RNN 的梯度消失问题，适合长序列建模。
4. 能够捕捉长距离依赖关系，适合处理长序列数据。
5. 在时间序列预测、文本生成、机器翻译等任务中表现优异。

3、简单给我介绍下BiLSTM网络?

BiLSTM（双向长短期记忆网络）是一种改进的RNN结构，通过两个独立的LSTM分别处理正向和反向序列信息，能够捕捉上下文依赖。其核心特点包括：
1. 双向结构：同时学习过去和未来的上下文信息，能够捕捉到输入序列中每个位置的过去和未来信息，更全面地捕捉序列中的上下文信息。
2. LSTM单元：解决梯度消失问题，适合长序列建模。
3. 相对LSTM来说，因为它的模型参数数量大约是 LSTM 的两倍，在一定程度上增加了计算量和训练时间。
4. 应用场景：文本分类、命名实体识别、机器翻译等。
优点：上下文信息捕捉能力强，适合序列数据建模。

4、简单给我介绍下GRU门控网络

1. 从结构上来说，GRU 是 LSTM 的变体，通过减少门控机制（将 LSTM 的输入门、遗忘门合并为更新门，并移除输出门），降低了模型复杂度。
2. 它的主要核心组件就是,更新门(这个是用来控制历史信息和当前信息的融合比例),重置门(这个用来决定是否忽略历史信息，捕捉短期依赖)
3. 它主要有几个优点吧：
   3.1 参数更少，计算效率更高，适合资源有限场景
   3.2 保留了 LSTM 解决梯度消失问题的能力，适合长序列建模
4. 对比 LSTM：GRU 性能接近 LSTM，但训练更快，适合对效率要求较高的任务。

5、大概给我讲下RNN网络与LSTM网络的区别?

1. RNN是最基础的循环神经网络，通过循环结构处理序列数据，它的核心思想是当前时刻的输出依赖于前一时刻的隐藏状态。结构相对比较简单，但存在梯度消失问题，难以捕捉长距离依赖关系。
2. LSTM（长短期记忆网络）是RNN的改进版本，引入了门控机制（输入门、遗忘门、输出门），能够更好地控制信息的流动。它的核心思想是通过记忆单元（Cell State）来保存长期信息，从某种程度上解决了RNN的梯度消失问题，适合处理长序列数据。
3. 例外，适合处理短序列任务，比如字符级语言模型或简单的分类任务。由于RNN结构简单，计算效率高，适合资源受限的场景。而**LSTM**适合处理长序列任务，比如机器翻译、文本生成、语音识别等。LSTM的门控机制能够有效捕捉长距离依赖关系，适合复杂任务。

6、为什么LSTM能解决梯度消失问题?

LSTM通过引入记忆单元和门控机制，能够选择性地保存或丢弃信息，从而避免了梯度在长序列中的衰减。

7、什么情况下你会选择RNN而不是LSTM或GRU?

当任务非常简单（如短序列分类）且计算资源有限时，RNN是一个不错的选择，因为它结构简单，计算效率高。

8、在实际项目中，如何选择RNN或LSTM?

如果是短序列任务且对计算资源要求不高，可以选择RNN；如果是长序列任务且需要捕捉复杂依赖关系，建议选择LSTM。

9、LSTM网络与GRU门控网络的区别?

1. LSTM（长短期记忆网络）通过引入三个门控机制（输入门、遗忘门、输出门）和记忆单元，解决了RNN的梯度消失问题，能够更好地捕捉长距离依赖关系。适合处理长序列复杂任务，比如机器翻译、文本生成、语音识别等。
2. GRU（门控循环单元）是LSTM的简化版本，通过两个门控机制（更新门、重置门）来控制信息的流动，减少了参数数量，计算效率更高。适合处理中等长度序列任务，尤其是对计算效率要求较高的场景，比如实时语音处理或中等长度的文本分类。GRU的性能接近LSTM

10、LSTM和GRU的门控机制有什么本质区别?

LSTM有三个门（输入门、遗忘门、输出门），而GRU只有两个门（更新门、重置门）。GRU通过合并输入门和遗忘门，简化了结构。

3、LSTM文本情感分析系统

使用LSTM/RNN/GRU来构建神经网络模型的步骤，重点，核心

- 构建词汇表(文本向量化/Embedding)
    - 分词（jieba分词）
    - 构建词汇表（字典/词典），做文本映射（文本到数字的映射，数字到文本的映射）

      词汇表构建完成之后，一般都会保存，格式：.json, .pk, .pt, .vec
    - 进行wordEmbeding,构建词向量矩阵
    - 将输入数据进行索引表示，I love you -> [6, 10, 200]
    - 不够的要进行填充，多的截取（根据你的设定的序列长度）
    - 对输入数据进行向量化（根据我们索引去向量矩阵中去找对应词向量）
- 构建网络模型
    - Embedding层->训练的, 矩阵大小（vocab_size(词汇表大小)，Embedding_dim(维度)）
    - 构建LSTM层(输入序列数据的维度, 隐藏层大小, 层数)
    - 全连接层（输入维度，输出维度）
- 数据划分，构建dataloader, 构建优化器, 损失函数

3.1、具体流程

==1、数据预处理==

- 从 CSV 文件中加载数据
- 提取数据中的唯一标签和词汇表
- 将句子拆分为字符，构建字符级别的词汇表。
- 将字符和标签映射为索引，方便模型处理。
- 将文本数据转换为模型可处理的格式（索引序列），并进行填充。
- 数据持久化
    - word_dict.pk保存
    - label_dict.pk保存

==2、构建网络模型==

网络结构定义，这里主要包括3层

- Embedding层（vocab_size+1处理）
- LSTM层（hidden_dim=100）
- 全连接层（output_dim=2）
- 前向传播逻辑
    - 将输入序列进行向量化
    - 将嵌入向量导入到lstm层，这里要注意lstm层输出的有3个数据，分别为

        - output: LSTM 在每个时间步的隐藏状态（hidden state）。
        - h_n: LSTM 在最后一个时间步的隐藏状态（hidden state）。
        - c_n: LSTM 在最后一个时间步的细胞状态（cell state）
    - 将其导入全连接层
    - 取最后时间步输出（output[-1]）

==3、数据集划分==

==4、训练配置==

- Adam优化器（lr=0.003）
- 交叉熵损失函数
- 前向传播，梯度清零，反向传播，计算梯度并更新参数
- 保存模型

==5、预测推理==

- 加载字典文件
- 模型权重加载
- 文本预处理：字符转索引，动态填充处理
- 预测执行：转置张量（transpose），argmax取预测结果

3.2、核心重点

- ==1. 数据预处理关键==
    - 字符级处理：解决OOV问题
    - 后置填充策略：统一输入维度
    - 字典持久化：保证一致性
- ==2. 模型结构核心==
    - Embedding设计：vocab_size+1预留位
    - LSTM配置：3层堆叠结构
    - 输出处理：末时间步特征提取
- ==3. 训练优化要点==
    - 设备管理：CPU/GPU统一处理
    - 梯度控制：zero_grad必要性
    - 模型保存：state_dict轻量化

提醒：

- 原理
- 网络结构
- 代码结构

4、基于seq2seq的机器翻译

4.1、什么是seq2seq? 给我简单介绍下?

Seq2Seq模型的核心思想是接受一个输入序列，通过编码（Encoder）将其映射到一个固定长度的表示，然后通过解码（Decoder）将这个表示映射回输出序列。这使得Seq2Seq模型适用于处理不定长输入和输出的任务。

1、Seq2Seq 是一种用于将一个序列映射到另一个序列的模型，通常由**编码器（Encoder）和解码器（Decoder）**两部分组成。
2、编码器：将输入序列（如句子）编码为一个固定长度的上下文向量（Context Vector），捕捉序列的语义信息。
解码器：基于上下文向量逐步生成输出序列（如翻译结果）。
3、它的编码器,解码器可以是RNN、LSTM 或 GRU网络
4、如果引入注意力机制（Attention），解决长序列信息丢失问题，提升模型性能。
5、能够处理变长输入和输出序列

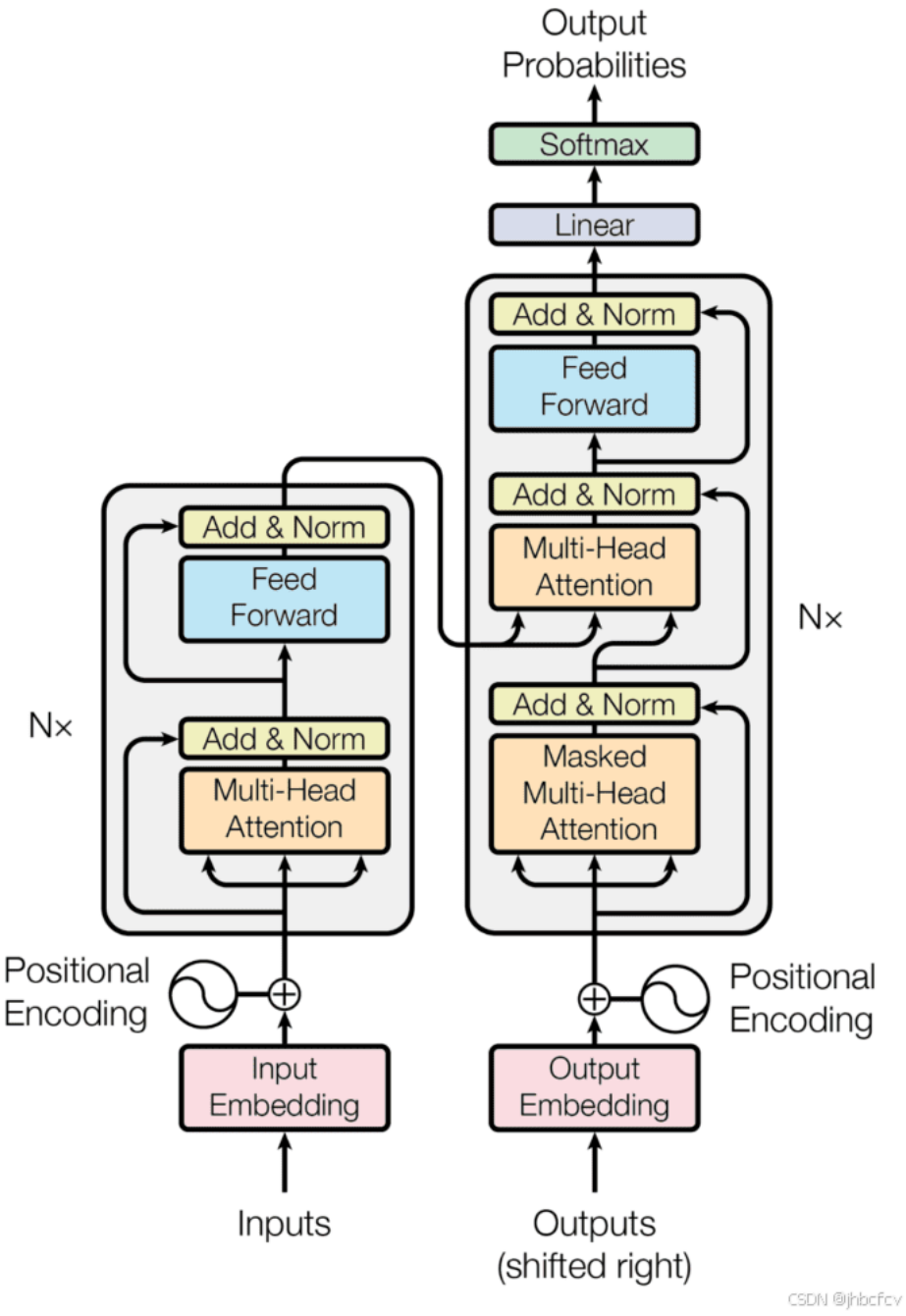
机器翻译的代码结果/逻辑：

1. 对文本进行分词处理，构建词汇表，这里需要注意, 需要设置最大序列长度,不够需要进行填充,多的需要进行截取。
2. 根据词汇表的大小，构建word2Embedding词嵌入向量矩阵
3. 构建网络模型
    1. 网络模型结构主要包括,Encoder，Decoder2个部分
    2. Encoder主要采用LSTM/GRU网络
    3. Decoder主干循环网络也采用LSTM/GRU, 在Decoder层当时有加入注意力机制,编码器的输出作为解码器的输入

       例外, 在训练过程中，采用教师压迫模式进行的。
4. 之后就是构建损失函数,优化器,设置epoch进行训练
5. 其实这里的核心重点还是一下几个方面：
    - 词汇表的构建
    - Word2Embedding过程
    - 网络结构的搭建，包括涉及的注意力机制, 教师压迫模式

5，基于transformer的谣言检测

5.1、Transformer结构



5.2、Tranformer模型的基本步骤：

- 输入嵌入：

  首先将输入句子转换为称为嵌入的数字表示。它们捕获输入序列中标记的语义含义。对于单词序列，这些嵌入可以在训练期间学习，也可以从预训练的单词嵌入中获得。

- 位置编码：

  位置编码通常作为一组附加值或向量引入，这些值或向量在将它们输入转换器模型之前添加到令牌嵌入中。这些位置编码具有对位置信息进行编码的特定模式。

- 多头注意力：

  自注意力在多个“注意力头”中运行，以捕获令牌之间不同类型的关系。Softmax函数是一种激活函数，用于计算自注意力机制中的注意力权重。

- 层归一化和残差连接：该模型使用层归一化和残差连接来稳定和加速训练。

- 前馈神经网络：自注意力层的输出通过前馈层传递。这些网络将非线性变换应用于标记表示，使模型能够捕获数据中的复杂模式和关系。
- 堆叠层：转换器通常由堆叠在一起的多个层组成。每一层都处理前一层的输出，逐渐细化表示。堆叠多个图层使模型能够捕获数据中的分层和抽象特征。
- 输出层：在神经机器翻译等序列到序列任务中，可以在编码器顶部添加单独的解码器模块来生成输出序列。
- 训练：Transformer 模型使用监督学习进行训练，在监督学习中，它们学习最小化损失函数，该函数量化模型的预测与给定任务的基本事实之间的差异。训练通常涉及优化技术，如 Adam 或随机梯度下降 （SGD）。
- 推理：训练后，模型可用于对新数据进行推理。在推理过程中，输入序列通过预训练模型传递，模型为给定任务生成预测或表示。

5.3、核心流程

- 数据加载与预处理
    - 从 TSV 文件中加载文本数据和标签。
    - 提取数据中的唯一标签和词汇表，并构建字符级别的词汇表。
    - 将字符和标签映射为索引，方便模型处理。
    - 将文本数据转换为模型可处理的格式（索引序列），并进行填充。
- 模型定义
    - 位置编码：为输入序列添加位置信息
    - 定义 Transformer 模型结构
        - **嵌入层（Embedding Layer）**：将输入的字符索引转换为稠密的向量表示
        - **位置编码（Positional Encoding）**：为输入序列添加位置信息。
        - **Transformer 编码器（Transformer Encoder）**：通过多头自注意力机制和前馈神经网络提取特征
        - **全连接层（Fully Connected Layer）**：将 Transformer 的输出映射到分类结果。
    - 前向传播
    - 将字符索引转换为稠密的嵌入向量
        - 为嵌入向量添加位置信息，
        - **Transformer 编码器**：通过多头自注意力机制和前馈神经网络提取特征
        - 对序列维度取均值，将每个样本的序列信息汇总为一个向量
        - **全连接层**：将汇总后的向量映射到分类结果
- 模型训练
    - 将数据加载为 DataLoader。
    - 构建损失函数(交叉熵)，优化器（Adam）
    - 前向传播，计算损失，梯度清零，反向传播，更新优化
- 使用训练好的模型进行预测