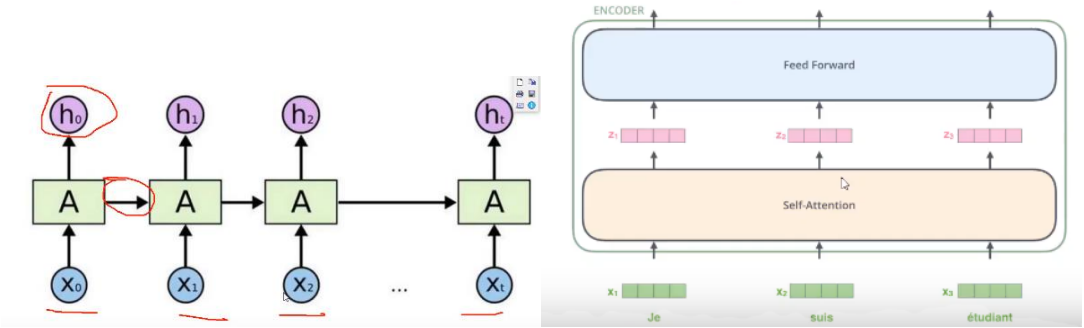
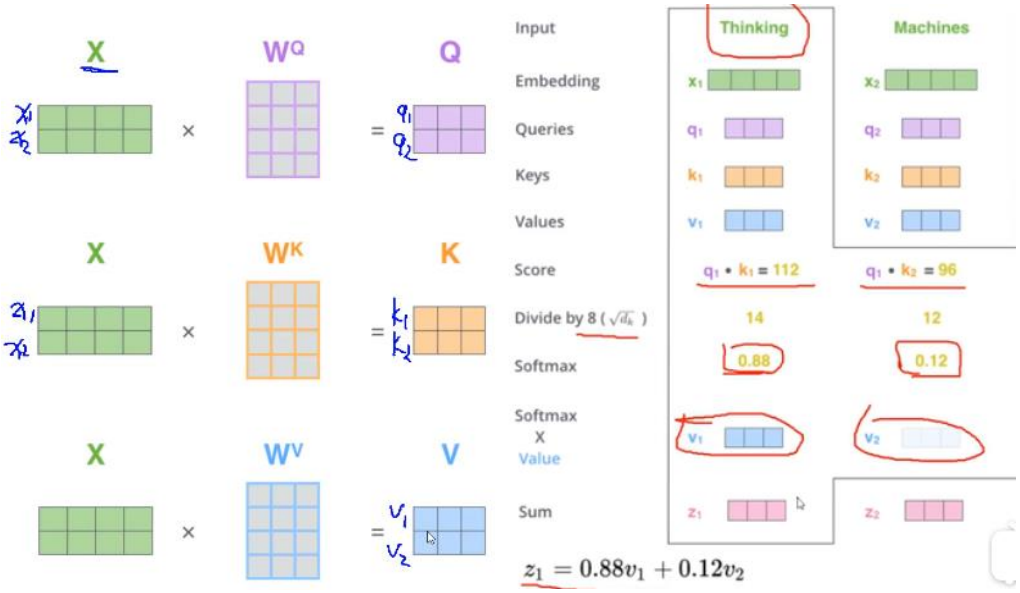
视频：https://edu.csdn.net/course/play/26498/334600

BERT

BERT 基本组成为 Seq2Seq 网络：输入和输出都为序列的网络，中间为 Transformer
为什么要 Transformer? 因为传统 RNN 网络每个循环体都需要上一个循环体的中间结果，无法并行运算（不独立）；采用 Self-Attention 机制并行计算，取代 RNN
为什么不用 word2vec? 相同的词表达含义一样，实际上不是这样的



Self-Attention 机制：在不同的语境下不同的词有不同的权重。
方法：词->向量编码 x1->权重向量编码 z1
预先定义三个矩阵（权重）Wq、Wk、Wv，分别表示要被查询的、等待被查的和实际的特征信息，使输入词向量分别与三个矩阵进行运算。然后用 softmax 来进行归一化，求得每个词在当前句子中占有的权重（影响程度）
若要查询第一个词与每个词的关系，则用 Wq1*Wk1（内积：若无关系则内积为 0）表示与第一个词的关系，Wq1*Wk2 表示与第二个词的关系，以此类推



Multi-headed 机制：提取多种词向量特征（上述只是提取一种特征）
通过多个头机制（一般 8 个）得到多个特征表达，然后将所有特征拼接起来，再加一层全连

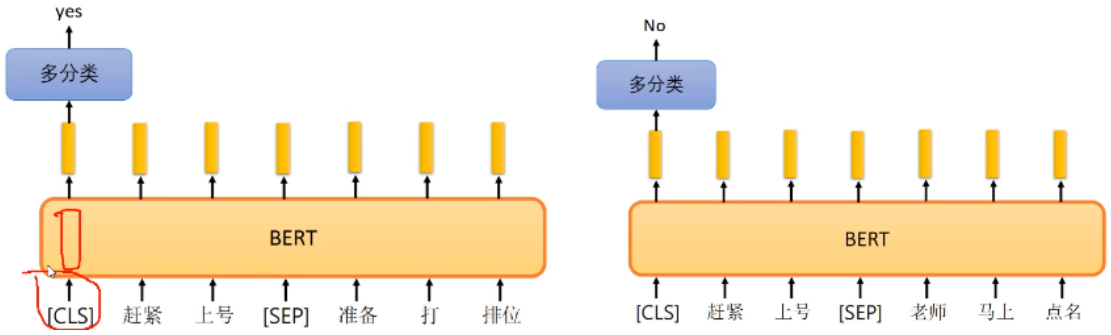
接来降维
注：通过上述过程得到词特征向量，一般情况下还要堆叠多层来得到最终的词特征向量

位置信息表达：相对于上述得到的词特征向量再加上位置信息编码，一般是周期信号（正余弦等）

LayerNormalize：对每个词的所有特征进行归一化处理（区别于对每批数据进行归一化），为了使得训练更快，更稳定。其中词的特征向量由上述处理得到
连接：层归一化的同时，加入残差连接

Decoder：相对于 Encoder，Self-Attention 计算不同（用 q 来查），加入了 MASK 机制。

训练 BERT：将句子中 15%（替换为 mask，随机变为其他词或不变）的词随机 mask 掉，让模型去预测被 mask 的是什么；预测两个句子是否应该连在一起（[seq]连接符，[cls]分类向量）



源代码解析

准备数据：
根据数据生成 tf_record 数据文件（如下为具体制作数据文件过程）
->对标签列表添加标记（顺序标记），为 label_map
->对每个输入序列进行 wordpiece 分词（并控制所有序列长度和不超过最大长度，截断）
->对每个序列加入起始和分割等特殊标记（同时设置新列表 segment_ids 来标记每个词属于第几句话）
->从词表中查询每个词的 id 来生成 input_ids（方便后续做词嵌入）
->设置对应的 input_mask 并补全（即有词的位置标为 1，否则标为 0，0 表示后续不再参与计算）
->构建 feature 样本并序列化保存（需要对每个样本都做上述处理，从 label_map 开始）

创建模型：
（此时是批处理输入来构建模型）
构建 embedding 层
->输入查询 input_ids（batch_size, seq_len）在词表中对应的编码（batch_size, seq_len, hidden_size），得到词嵌入编码 input_ids

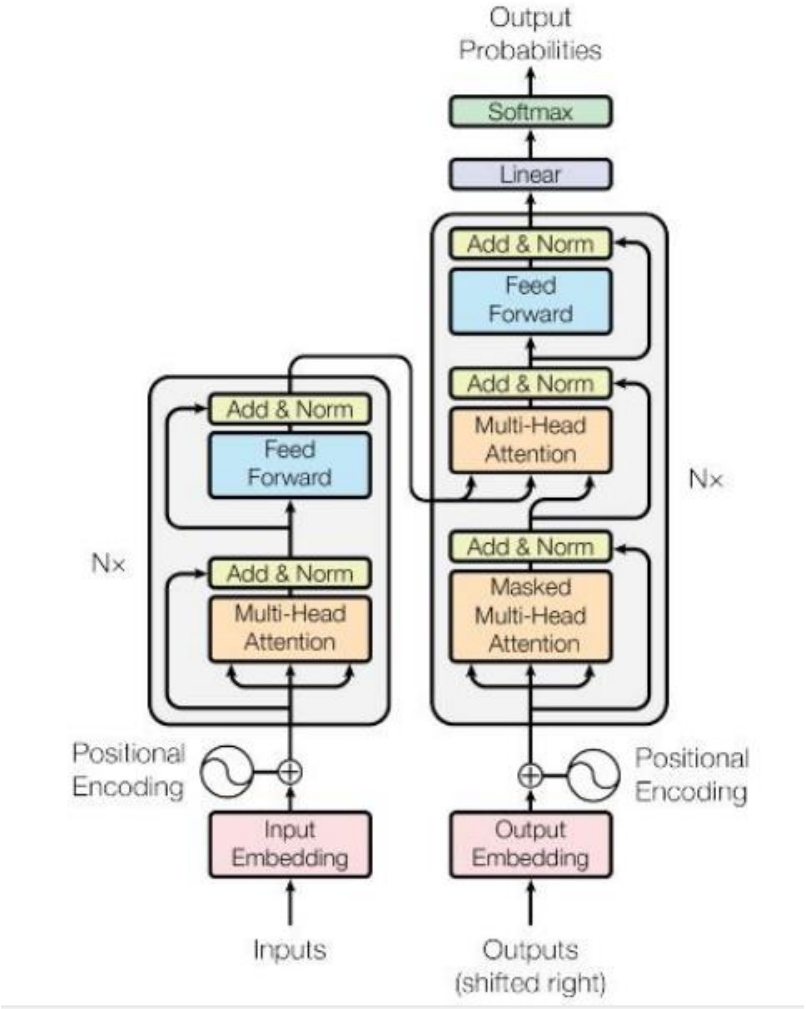
->对生成的 input_ids（batch_size*seq_len, hidden_size），首先对 segment_ids 进行编码生成 token_type_table，维度为（2, hidden_size）因为只有 0,1 需要编码；之后利用 segment_ids 来构建 one_hot_ids（batch_size*seq_len, 2）表示需要知道每个词对应的是 0 还是 1；接着使 one_hot_ids 和 token_type_table 相乘得到从属关系 output，然后与 input_ids 相加即可嵌入。初始化位置信息（1, batch_size*seq_len, hidden_size）然后相加嵌入上述结果中（1 表示每行信息都要加，因此没必要生成相同的多行）【概括来说就是嵌入语句从属关系和位置信息】

->加入层归一化（也可再加入 dropout）,记最后输出为 embedding_output

构建 Transformer（encoder）

->将 2D mask 变为 3D attention mask，表示每个序列的每个词应该和该序列的哪个词计算（当对应位置为 0 时表示当前词不与那个位置的词计算），刚开始每个序列的每个词 mask 对应的仅仅是一个 0/1，相当于每个序列的 mask 是 2D 的。

->将输入 embedding_output 结合起来，即将同 batch 组合由 3D 变为 2D（batch_size 那个维度合并），进入 multi_head 循环多次提取特征，对于每个循环创建一个 selt-attention。每个自注意力模型会创建若干 Q，K 和 V 矩阵/dense 层（每个头每个词都会有一类矩阵），其中 Q 由 from_tensor（嵌入层输出）生成，K 由 to_tensor 生成，V 由 to_tensor 生成。接着 QK 相乘计算内积并通过 softmax 得到自注意头的矩阵信息，最后将所有头矩阵拼接经过全连接层输出【概括来说就是构建 wordpiece】



新数据集制作：

情感分析数据集展示（0 表示特别满意，1 表示满意，2 表示不满意）

test_sentiment.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

1 于山宾馆服务非常差,是我从未见过的恶劣.柜台动作慢,态度凶,脸色臭,房间没空调又冷又潮,餐厅更差.可能是政府公营,不在乎吧 2

2 酒店设施陈旧，估计已经10年没有更新。壁纸、浴缸漆面剥落，马桶盖已损坏，空调系统基本瘫痪，噪音较大暖风几乎没有，入住酒店其内部停车场仍要收费。建议偕程应定期对推荐酒店进行筛选，合格酒店进行保留，不合格酒店进行剔除，这样才能保证服务的总体品质。 2

3 楚辞文化知识的学习和研究是大学语文普遍涉及的领域屈原所代表的时代精神值得后人深思和学习此书是目前对楚辞注解最全面，最权威的著作。其所涉及的知识层面无论广度或深度而言，是很符合一般具普通文言知识读者的。中华书局历来在编排文字及版本上尊重原著作者的 1

4 总体感觉还不错。房间很干净、简洁。网上所披露的位于海河最美的一段，特意要了6层的房间，可是却没看到海河的影子 1

5 linux改xp花了一下午时间。散热不好，cpu温度就没下过50，玩游戏能上70，比较吓人。触摸板关不掉，打字经常碰到。 2

6 12月1日又入住了瑞吉红塔，服务还是很好，但感觉不到以前的会心微笑了。不过，瑞吉红塔的游泳池还是一如既往地好 1

7 1完美屏上帝保佑2散热没有那么多问题3安装系统还好了，我用的是新萝卜系统3.0，然后大家可以参考 http:wuyilala.comindex.php20090609140803，注意一下第四步，一切就ok啦！cpu真的很强悍很强1

8 从来不喝蒙牛伊利 2

9 我怀孕生儿子时看的就是这本书,现在儿子已经14岁了特别健康,朋友的女儿结婚,我买了送给她做礼物,希望她也生个健健康康的宝贝.是日本专家写的孕期指导,特别体贴有实际操作价值.每个月妈妈的身体变化,注意事项,宝贝在肚里的变化和注意事项,大小都提醒.特别是:告诉 1

->根据预设的数据读取方法定义属于自己的数据读取方法

```
class DataProcessor(object):...


class MyProcessor(DataProcessor):
    """Base class for data converters for sequence classification data sets."""

    def get_train_examples(self, data_dir):
        """Gets a collection of `InputExample`s for the train set."""
        raise NotImplementedError()

    def get_dev_examples(self, data_dir):
        """Gets a collection of `InputExample`s for the dev set."""
        raise NotImplementedError()

    def get_test_examples(self, data_dir):
        """Gets a collection of `InputExample`s for prediction."""
        raise NotImplementedError()

    def get_labels(self):
        """Gets the list of labels for this data set."""
        raise NotImplementedError()


class XnliProcessor(DataProcessor):...


class MnliProcessor(DataProcessor):...


class MrpcProcessor(DataProcessor):...
```

```
class MyProcessor(DataProcessor):
    """Base class for data converters for sequence classification data sets."""

    def get_train_examples(self, data_dir):
        """Gets a collection of `InputExample`s for the train set."""
        file_path = os.path.join(data_dir, 'train_sentiment.txt')
        f = open(file_path, 'r', encoding='utf-8')  # 读取训练数据
        train_data = []
        index = 0
        for line in f.readlines():
            guid = 'train-%d' % (index)
            line = line.replace('\n', '').split('\t')
            text_a = tokenization.convert_to_unicode(str(line[1]))  # 输入数据在第2列
            label = str(line[2])  # 数字不需要转换为unicode
            train_data.append(
                InputExample(guid=guid, text_a=text_a, text_b=None, label=label))  # 生成与预置方法一样的对象
            index += 1
        return train_data
```

```python
# 自制情感分析数据集读取方法
class MyProcessor(DataProcessor):
  """Base class for data converters for sequence classification data sets."""

  def get_train_examples(self, data_dir):...

  def get_dev_examples(self, data_dir):...

  def get_test_examples(self, data_dir):
    """Gets a collection of `InputExample`s for prediction."""
    import pandas as pd
    file_path = os.path.join(data_dir, 'test.csv')
    test_df = pd.read_csv(file_path, encoding='utf-8')
    test_data = []
    index = 0
    for index, test in enumerate(test_df.values):
        guid = 'test-%d' % index
        text_a = tokenization.convert_to_unicode(str(test[0]))
        label = str(test[1])  # 数字不需要转换为unicode
        test_data.append(
            InputExample(guid=guid, text_a=text_a, text_b=None, label=label))  # 生成与预置方法一样的对象
    return test_data

  def get_labels(self):
    """Gets the list of labels for this data set."""
    return ['0', '1', '2']
```

->main 方法中设置对应关系，运行程序即可

```python
def main(_):
  tf.logging.set_verbosity(tf.logging.INFO)

  processors = {
      "cola": ColaProcessor,
      "mnli": MnliProcessor,
      "mrpc": MrpcProcessor,
      "xnli": XnliProcessor,
      'my': MyProcessor,
  }
```

--task_name=my # 模型名称（如上图）

\

--do_train=true

\

--do_eval=true

\

--data_dir=../GLUE/glue_data/mydata # 数据集路径

\

--vocab_file=../GLUE/BERT_BASE_DIR/chinese_L-12_H-768_A-12/vocab.txt # 词库位置

\

--bert_config_file=../GLUE/BERT_BASE_DIR/chinese_L-12_H-768_A-12/bert_config.json # 配置参数

\

```
--init_checkpoint=../GLUE/BERT_BASE_DIR/chinese_L-12_H-768_A-12/bert_model.ckpt    # 预训练模型参数
\
--max_seq_length=64   # 最大有效长度
\
--train_batch_size=1   # 批大小
\
--learning_rate=2e-3   # 学习率
\
--num_train_epochs=3.0   # 训练轮数
\
--output_dir=../GLUE/output   # 模型训练输入位置
```

命名实体识别 NER

对需要的“专有”名词进行识别的任务。对于数据集，若非实体名则标注为 O，否则该实体名第一个字标注为 B-XXX（X 自定义类别标识），其他字标注为 I-XXX。采用 BERT+CRF 模型共同完成。

BERT 模型：

->得到一个所有序列的输出 embedding（batch_size, seq_len, embedding_size）

CRF 模型：（BERT 模型可单独实现，但是加上 CRF 效果更好）

->添加 CRF 预处理层（也可以先加一个 BLSTM 再加 CRF 预处理层），传入的参数是 embedding。相当于加了一层全连接层，输出的维度是分类总数 num

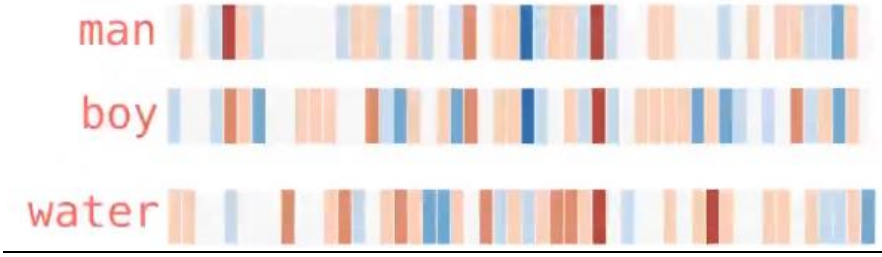
->添加 CRF 层，构建一个 num*num 的转移矩阵，目的是为了筛选结果，即如果第一个是 B-LOC，那么下一个词不能是 B-XXX 等，这样可以使得结果更加准确。最后计算与真实标签的似然函数值作为损失返回即可

总结：任何 NLP 任务，最重要的是提取特征，而 BERT 可以做到对特征的提取，因此可以实现对多种任务进行处理。同时也可以利用该模型额外处理来适应自己的 NLP 项目

词向量模型

词向量模型应该做到：与词数据表达相近、与词位置相关、与相近词表达相近

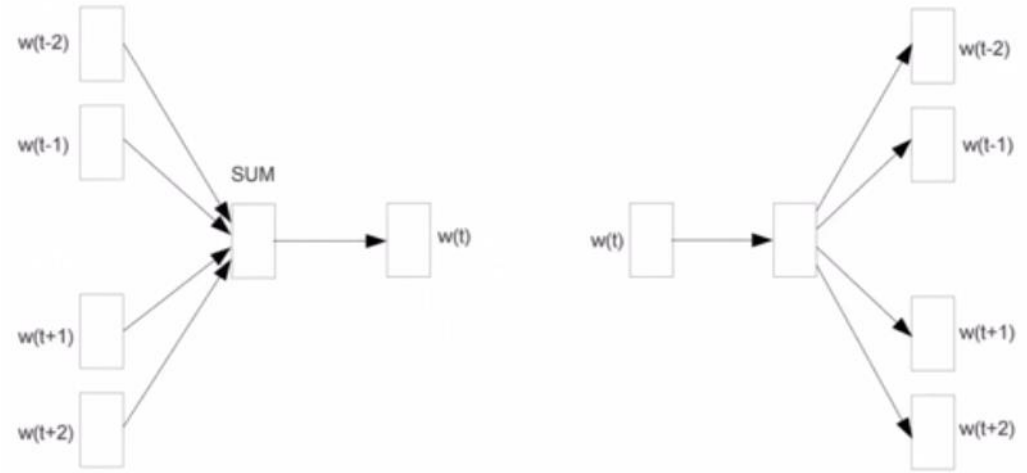
因此对于一个词，应该从多个维度（一般 50-300 维）进行描述，这样就可以对多个词计算相似度



词向量模型相当于一个神经网络模型，首先随机初始化一个词表（2D 向量表），然后输入是从中选出 N 个词向量（Look up embeddings），输出是计算得到的一个结果向量，然后通过反向传播更新词表，同时更新输入的词向量。通过多次训练从而得到更为准确地词表。

词向量训练模型有：CBOW 模型（左图）；Skip-gram 模型（右图）等。若输出是词向量，那么可能维度过大而计算复杂，因此改进为输入上文和下文，输出为 1/0，而由于语料本身几乎都是正样本，所以需要负样本采样辅助训练。

负样本模型：一个词对应一个正样本，对于负样本，可以随机从其他样本中抽取 N 个来作为负样本。正负样本组成一个向量，类似于 one-hot，从而替代 softmax 预测所有的词概率，提高运行效率



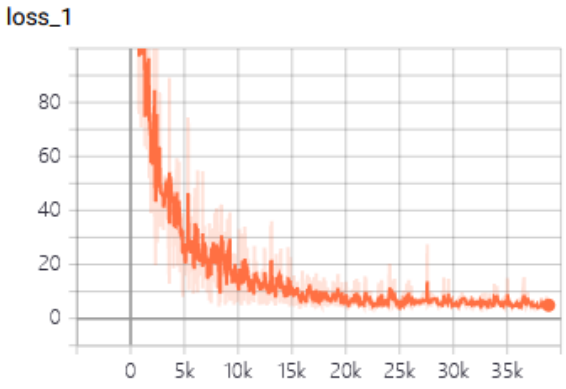
源代码解析：

->读取文本语料库（提前设置训练多少词，其余词会被标记为 UNK）

->对词进行编码，并求得编码-词对应表

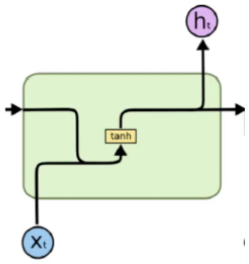
->生成一批数据，构造输入 batch 和输出数据 labels，其中一个 batch 对应两个 labels（因为有上文和下文）【skip-gram 模型数据】

->结合负样本模型进行训练（每次找到最近的 N 个词进行匹配和损失修正）

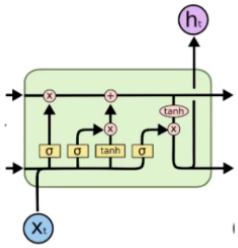


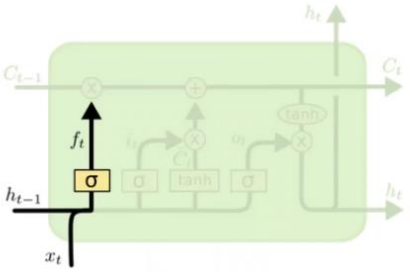
RNN 与 LSTM

RNN



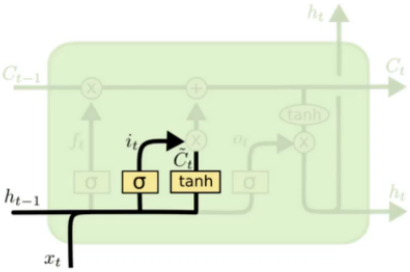
LSTM





$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

f_t 与C_{t-1}计算决定丢弃什么信息



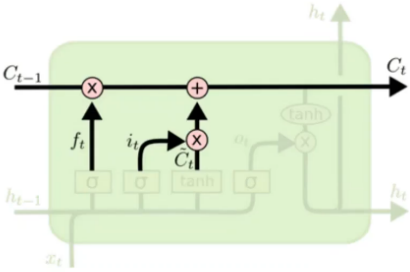
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t 要保留下来的新信息

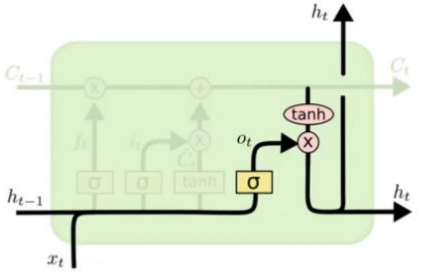
C_t 新数据形成的控制参数

确定更新的信息



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

更新细胞状态



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

利用新的控制参数产生输出

输出信息

一般取最后一个单元的输出

NLP 应用与影评分析

应用：对话系统、情感分析、图文映射、机器翻译和语音识别等

影评分析输入：（batch_size, time_step, vec）

数据集准备：训练集使用的是 IMDB 数据集。这个数据集包含 25000 条电影数据，其中 12500 条正向数据，12500 条负向数据。这些数据都是存储在一个文本文件中。正向数据包含在一个文件中，负向数据包含在另一个文件中。首先将数据集中的每个词得到其 id，接着拿 id 找到对应词的词向量，最后将每个文件（影评）的词向量下标 ids 记录出来，便于后续训练

训练与测试：将每个文件得到词向量矩阵，送入 LSTM 模型中进行训练，同时保存模型。训练完成后加载保存训练好的模型进行测试

命名实体识别

数据处理：向量长度设置+字典映射+数据填充到指定长度。获得训练数据、语料库和标签集

构建模型：嵌入层（输入语料库及维度）+双向 LSTM 层+CRF 后处理层