情感分析大作业技术报告

一、问题分析

情感分析也称为意见挖掘（ Opinion Mining），是自然语言处理(NLP)的一个领域，它构建的系统，用于在文本中识别和提取观点。 通常，除了识别观点之外，这些系统还提取描述的特征，例如：发言者表达积极或消极的意见、正在谈论的事情的主题、表达意见的个体或者实体等。

观点有明确和含蓄之分，明确的观点，例如“这部电影真好看”，这样的句子更加直白也更易分析，而含蓄的观点，例如“真该呆在家！”，这样的观点很难从字面上看出表达者的本意，需要结合上下文语境以及强大的先验知识才能判断出表达者的情感偏向。本次任务要对75000条电影评论分析，判定类别为正面评论或者负面评论。通过训练集训练出一个二分类判别器，然后利用训练好的判别器对要判定的数据集判断情感偏向，最后提交结果。

二、数据预处理

数据预处理十分重要，有时候数据预处理的质量对最后结果的影响是决定性的。

- **标点符号**:很多时候标点符号也会承载人们的各种情感，例如“！！！”，“:-(”。但是标点符号所隐含的情感与上下文有强烈的联系，而且不如文字直接，所以这里我们将标点符号去掉。
- **大小写转化**:很多时候文字的大小写并没有本质的区别，但是在数据处理的时候，同一文字的大小写对应的是不同的字符与编码，这会给 数据处理带来额外的影响，所以将文本中的大写字母转换成小写字母，并分割成一个个单词的形式，以便后续的分析。
- **停用词**：停用词是指那些经常出现但是又没有多大意义的词汇，在英文中比如说“a”,”and”,”is”,”the”等等。由于我们所用的模型很多需要根据词频或统计的方法进行文本分析，所以为了减少停用词对准确度的影响，常常把这些词去掉，有时候也会根据需求手工添加停用词。这里我们选择去掉停用词。
- **词形还原**：词性还原是把一个词汇还原为一般形式，用于表达完整语义，例如去掉单词的词缀，提取单词的主干部分，通常提取后的单词会是字典中的单词。比如，单词“cars”词形还原后的单词为“car”，单词“ate”词形还原后的单词为“eat”。

三、特征提取

我们需要处理的文本是句子级别的文本序列，我们需要将文本序列转化为模型能够接受的向量输入，即特征提取。在使用传统机器学习方法的时候，我们需要将获取句子级别的特征，这里我们使用的句子特征主要有两种，分别是词向量和句子向量。

- **词向量**：我们首先将经过数据预处理的数据集使用word2vec进行训练，获得每个单词的词向量，然后我们将该句子中所有单词的词向量进行求和并取平均值作为句子向量。
- **句子向量**：在使用了word2vec生成词向量，通过对句子的进行词向量简单求均值的方式得到句向量并使用逻辑回归进行分类后。我们思考后认为，也许简单求均值的方式并不能够很好地表达句向量。**因为忽略了词序与上下文**这些重要的信息，因此我们选择进一步尝试Doc2Vec。

  Doc2Vec是一种非监督式算法，基于 Word2Vec。这个方法在原有基础上添加 paragraph / document 向量，并存在两种方法：DM（Distributed Memory，分布式内存）和分布式词袋（DBOW）。

- DM 试图在给定前面部分的词和 paragraph 向量来预测后面单独的单词。即使文本中的语境在变化，但 paragraph 向量不会变化，并且能保存词序信息。
- DBOW 则在给定文档向量 paragraph 的情况下预测文档中一组随机单词的概率。

  首先读取csv文件中的信息。由于 Gensim 的 Doc2Vec 要求每个文档/段落包含一个与之关联的标签。此处使用 TaggedDocument进行处理。添加一个形如 “TRAIN_i” 或者 “TEST_i”的标签。

```
def labelizeReviews(reviews, label_type):
    for i,v in enumerate(reviews):
        label = '%s_%s'%(label_type,i)
        yield
gensim.models.doc2vec.TaggedDocument(gensim.utils.simple_preproc
ess(v,max_len=100), [label])
```

  训练时，我们采用官方教程中推荐使用的超参数，并同时训练两个模型。

```
# Doc2Vec(dbow,d100,n5,mc2,t4)

model_dm = gensim.models.Doc2Vec(min_count=2, window=10,
vector_size=size, sample=1e-3, negative=5, workers=3)
model_dbow = gensim.models.Doc2Vec(min_count=2, window=10,
vector_size=size, sample=1e-3, negative=5, dm=0, workers=3)
```

  在预测时，将待预测的段落输入 DM 和 DBOW 两个模型，并将得到的向量叠加，以得到更好的结果。

```
def getVecs(model, corpus):
    vecs = []
    for i in corpus:
        vec =
model.infer_vector(gensim.utils.simple_preprocess(i,max_len=300
))
        vecs.append(vec)
    return vecs

test_vecs_dm = getVecs(model_dm, test_content_list)
test_vecs_dbow = getVecs(model_dbow, test_content_list)
test_vecs = np.hstack((test_vecs_dm, test_vecs_dbow))
```

四、模型及模型融合

4.1 传统机器学习方法

在使用传统机器学习方法的时候，我们主要使用词向量平均值以及句子向量作为句子的特征，主要使用了以下三种分类模型。

4.1.1朴素贝叶斯

朴素贝叶斯分类器（Naive Bayes classifier）采用了属性条件独立性假设（attribute conditional independence assumption）：对所有已知类别，假设所有属性相互独立。

$$P(c|x)=\frac{P(c)P(x|c)}{P(x)}=\frac{P(c)}{P(x)}\prod_{i=1}^{d}P(x_i|c)$$

其中d为属性数目，x_i为x 在第i 个属性上的取值。因为对于所有类别的$P(x)$相同，因此贝叶斯判定准则有：

$$h_{nb}(x)=arg\max_{c\in\gamma}P(c)\prod_{i=1}^{d}P(x_i|c)$$

那么朴素贝叶斯分类器的训练过程就是基于训练集D来估计类先验概率，并为每个属性估计条件概率$P(x_i|c)$ 。

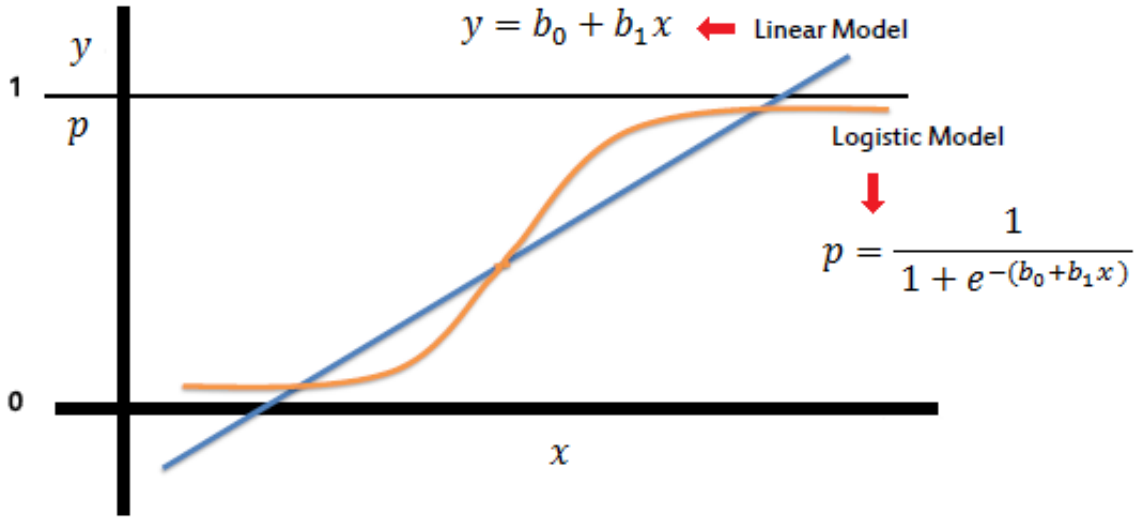
4.1.2支持向量机

支持向量机是一种二分类模型，它的目的是寻找一个最优分类面超平面来对样本进行分割，分割的原则是间隔最大化，最终转化为一个凸二次规划问题来求解。最优分类面就是要求分类线不但能将两类正确分开(训练错误率为0)，且使分类间隔最大。**SVM**考虑寻找一个满足分类要求的超平面，并且使训练集中的点距离分类面尽可能的远，也就是寻找一个分类面使它两侧的空白区域**(Margin)**最大。由简至繁的模型包括：

- 当训练样本线性可分时，通过硬间隔最大化，学习一个线性可分支持向量机
- 当训练样本近似线性可分时，通过软间隔最大化，学习一个线性支持向量机
- 当训练样本线性不可分时，通过核技巧和软间隔最大化，学习一个非线性支持向量机

4.1.3逻辑回归

逻辑回归是概率型非线性回归，但其本质是线性回归，只是在特征到结果的映射中加入了一层函数映射，即先把特征线性求和，然后使用函数sigmoid来预测。



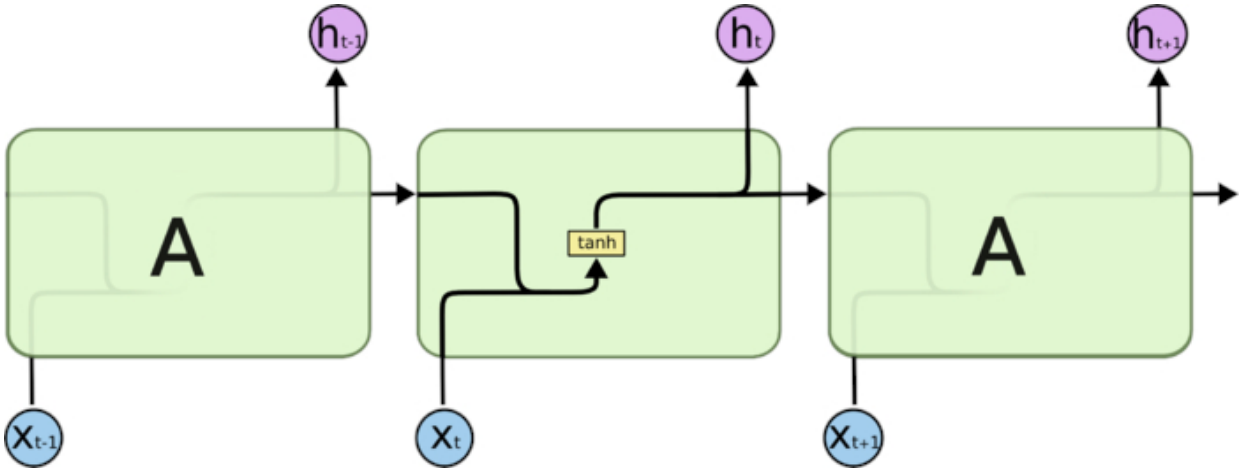
4.2神经网络方法

在使用神经网络的方法中，我们主要使用了RNN，LSTM以及BERT模型，在进行训练之前，我们首先将文本序列转化成模型所需要的输入，这里我们使用了同样的处理方法。

- 词嵌入：嵌入层需要传入整数类型的数据，因此我们需要将单词编码为整数类型。最简单的方法是创建一个从单词到整数的映射的字典，然后我们能将每条评论转换为整数传入网络。这里为了进一步提升结果，我们不使用随机初始化的词向量矩阵，而是经过word2vec训练后的词向量作为词向量矩阵。我们使用了影评语料训练的词向量模型和基于维基百科的Glove词向量，这里词向量均使用的是300维的词向量。
- 数据截断：在查看训练集中句子长度分布后，发现有的评论长度太长，对于RNN训练来说，需要太多的步骤，因此我们的处理方法是将每条评论的单词数控制为400个单词。这意味着对于不足400个单词的评论，将用0补上，对于超过400个单词的评论，我们只截取前400个使用。在BERT模型中受限于GPU限制，我们使用了256作为数据截断长度
- 训练集划分：在这里，我们定义一个划分比例，split_frac，代表数据保留到训练集中的比例，通常设置为0.8或0.9，然后剩余的数据为验证集。

4.2.1RNN

在这里我们将使用RNN(循环神经网络)对电影评论进行情感分析，结果为positive或negative，分别代表积极和消极的评论。至于为什么使用RNN而不是普通的前馈神经网络，是因为RNN能够存储序列单词信息，得到的结果更为准确。这里我们将使用一个带有标签的影评数据集进行训练模型。



- 模型构建

```
lstm_size = 256
lstm_layers = 1
batch_size = 500
learning_rate = 0.1


#加1是因为字典从1开始，我们用0来填充
# #创建图对象
graph = tf.Graph()
#像图中添加节点
with graph.as_default():
    inputs = tf.placeholder(tf.int32, [None, None], name = 'inputs')

    labels = tf.placeholder(tf.int32, [None, None], name = 'labels')
    keep_prod = tf.placeholder(tf.float32, name = 'keep_prod')


```

```
with graph.as_default():
    embedding = tf.Variable(tf.convert_to_tensor(all_word_vector))
    embed = tf.nn.embedding_lookup(embedding, inputs)


with graph.as_default():
    lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)
    drop = tf.contrib.rnn.DropoutWrapper(lstm,
output_keep_prob=keep_prod)
    cell = tf.contrib.rnn.MultiRNNCell([drop] * lstm_layers)
    initial_state = cell.zero_state(batch_size, tf.float32)

with graph.as_default():
    outputs, final_state = tf.nn.dynamic_rnn(cell, embed,
initial_state=initial_state)

with graph.as_default():
    predictions = tf.contrib.layers.fully_connected(outputs[:, -1], 1,
activation_fn=tf.sigmoid)
    cost = tf.losses.mean_squared_error(labels, predictions)

    optimizer =
tf.train.AdadeltaOptimizer(learning_rate).minimize(cost)

with graph.as_default():
    correct_pred = tf.equal(tf.cast(tf.round(predictions), tf.int32),
labels)
    accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

def get_batchs(x, y, batch_size = 100):
    n_batchs = len(x) // batch_size
    x, y = x[:n_batchs * batch_size], y[:n_batchs * batch_size]
    for ii in range(0, len(x), batch_size):
        yield x[ii:ii+batch_size], y[ii:ii+batch_size]


epochs = 80

with graph.as_default():
    saver = tf.train.Saver()
```

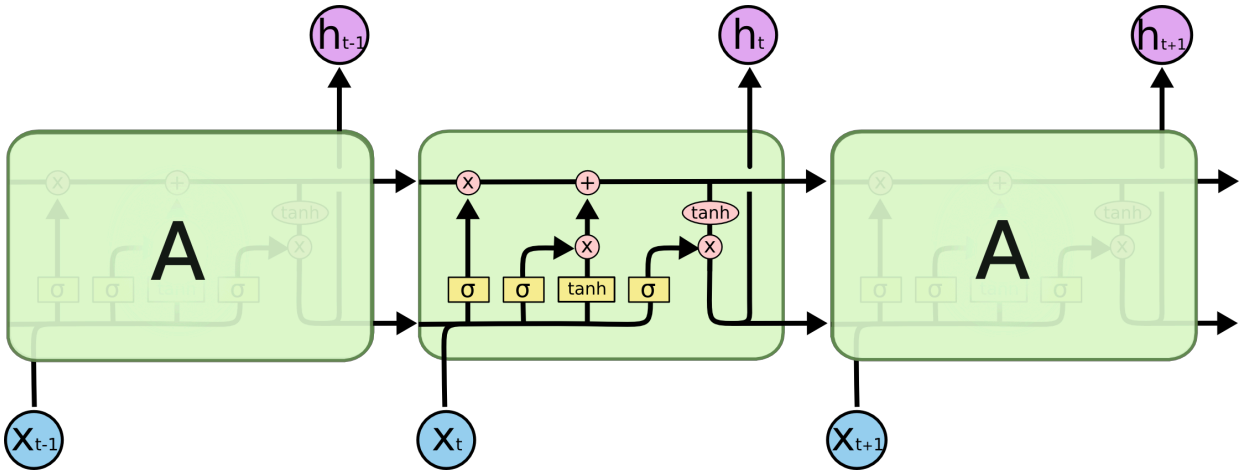
4.2.2LSTM

LSTM，**是为了解决长期依赖问题而专门设计出来的**，所有的RNN都具有一种重复神经网络模块的链式形式。在标准RNN中，这个重复的结构模块只有一个非常简单的结构，例如一个tanh层。LSTM 同样是这样的结构，但是重复的模块拥有一个不同的结构。不同于单一神经网络层，这里是有四个，以一种非常特殊的方式进行交互。LSTM模型主要包括输入门，输出门以及遗忘门三种结构，十分适合

处理序列问题。

LSTM由单元（cell）组成，通过门控单元控制信息的流通。

- 遗忘门：先前输出（即隐藏状态）的信息和来自当前输入的信息经sigmoid函数激活，输出介于0-1之间。越接近0意味着越容易被忘记，越接近1则越容易被保留。
- 输入门：
  - 先前输出和当前输入经sigmoid函数，计算出哪些值更重要；
  - 同时，把先前输出和当前输入给tanh函数，生成候选状态；
  - 最后，把tanh的输出与sigmoid的输出相乘，生成更新状态
- 状态更新：
  - 先前cell状态和遗忘门输出的向量点乘，由于越不重要的值越接近0，原隐藏状态中不重要的信息被丢弃。
  - 新的输出，与当前cell的候选状态相加，输出更新后的cell状态。
- 输出：
  - 输出建立在cell状态基础上
  - 先前输出与当前输入经过sigmod，决定哪一部分cell状态需要被输出-输出门
  - 状态经过tanh后，与输出门相乘，只输出想要输出的。



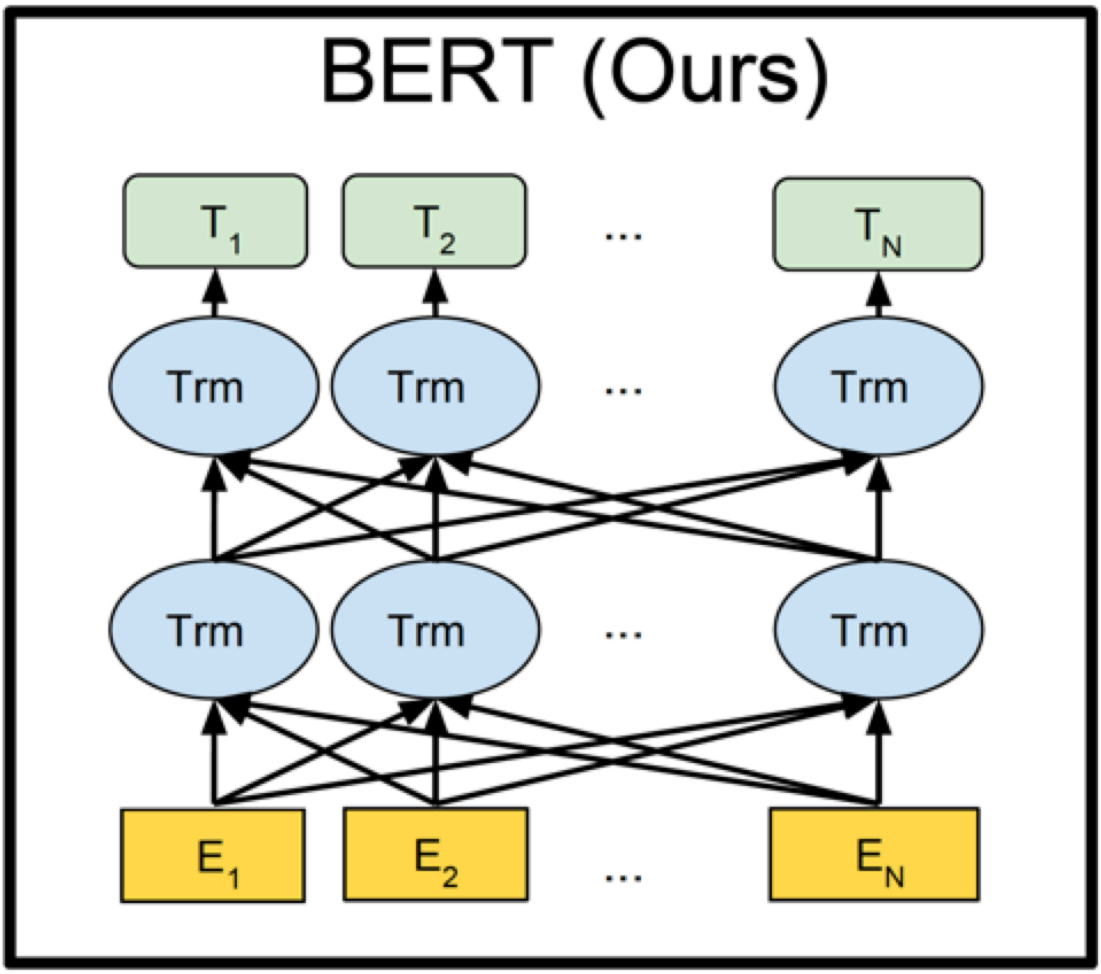

```
embedding_layer = Embedding(len(word_index) + 1,
                            EMBEDDING_SIZE,
                            weights=[embedding_matrix],
                            input_length=800,
                            trainable=True)
model=Sequential()
model.add(embedding_layer)
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(HIDDEN_LAYER_SIZE,dropout=0.2,recurrent_dropout=0.2),merge_mode='concat'))
#分类问题，分类结果的输出纬度为1
model.add(Dense(1))
#二分类，使用sigmoid作为分类函数
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
15 model.fit(Xtrain,ytrain,batch_size=BATCH_SIZE,epochs=NUM_EPOCHS,validat
   ion_data=(Xtest,ytest),callbacks=callback_lists)
16 model.save("lstm_model.h5")
```

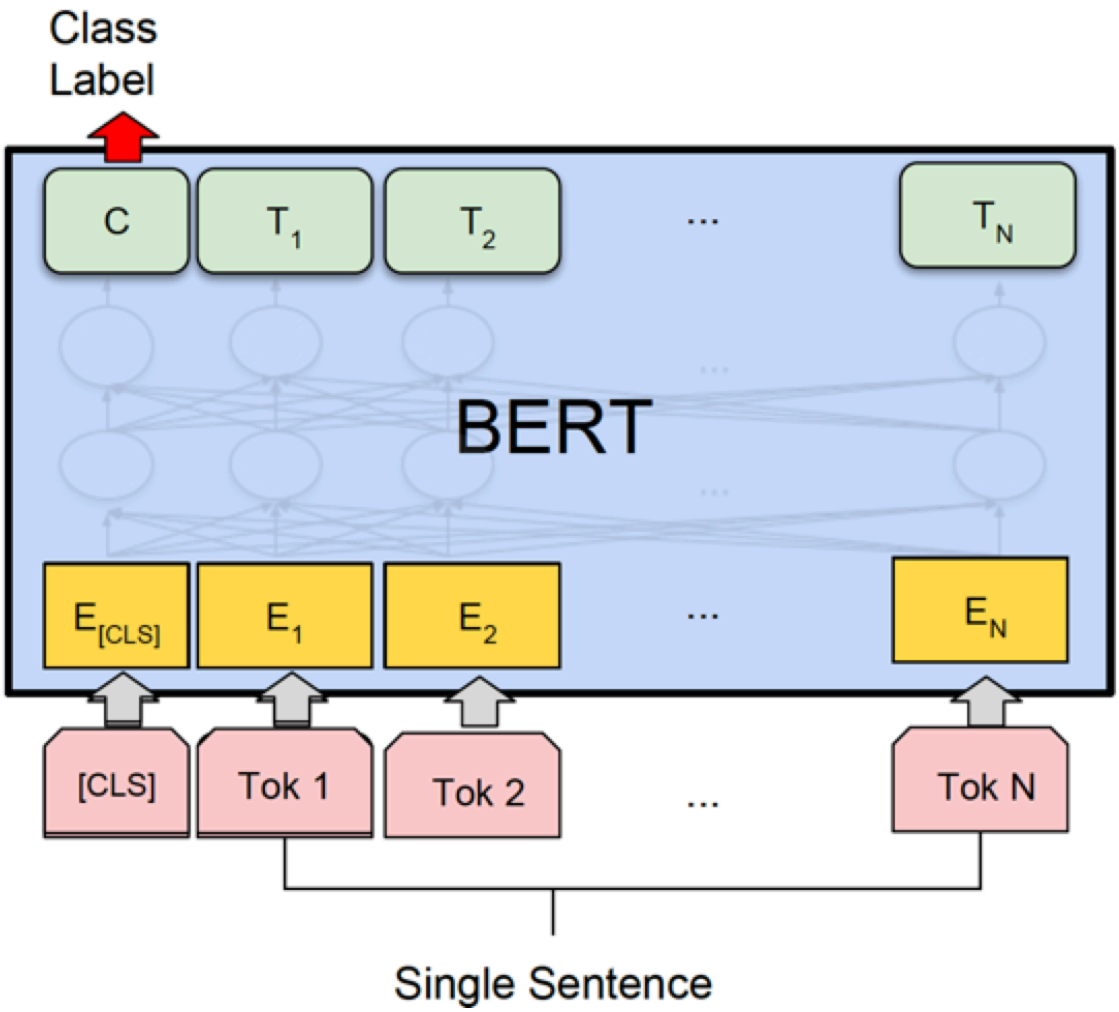
4.3.3BERT

BERT是基于双向LSTM的语言预训练模型，它定义了预测单词和下一句预测两个目标函数作为训练目标。

- 随机掩盖一些输入的单词，然后对这些被掩盖的单词进行预测。
- 随机给出一些句子对，判断句子对中的两句话是否是连续上下文。



- Fine-tuning:对于单句分类任务我们只需要在模型的输出后面接上一层softmax作为分类概率输出即可。



- 模型参数

```
{
  "attention_probs_dropout_prob": 0.1,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "type_vocab_size": 2,
  "vocab_size": 30522
}
```

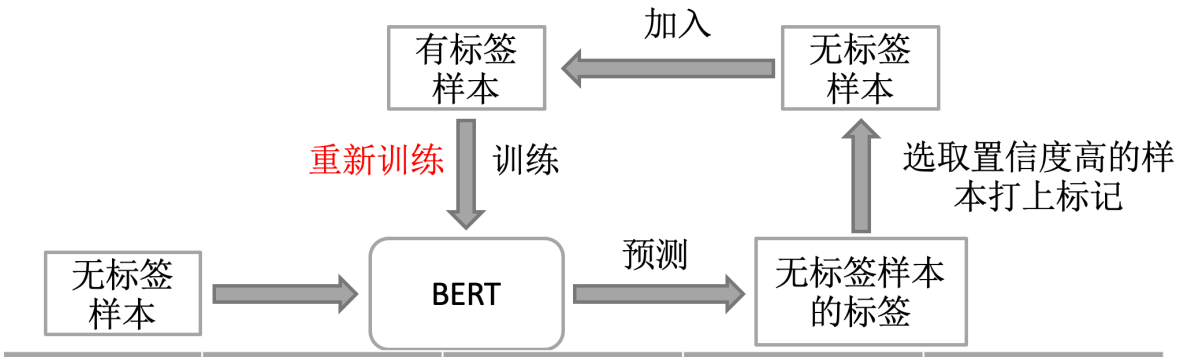
4.3 模型融合

4.3.1 集成

我们对传统机器学习方法以及LSTM和RNN模型的输出进行简单的投票法集成，得到了比单模型更好的结果。

4.3.2 半监督学习

我们针对单模型效果最好的BERT模型进行了半监督学习，算法流程如下：



实验序号	1	2	3	4
训练集	原训练集	原训练集	原训练集+预测后的无标签样本	3号实验数据去除标点符号
序列输入长度	128	256	256	256
结果	91.22	91.72	92.22	92.30

五、结果分析

5.1 不同模型结果对比

模型	评分
LSTM(glove)	86.94
BiLSTM(glove)	88.14
RNN	87.66
LR(D2V)	89.08
SVM	82.4
Naive Bayes Classifier	86.38
BERT	**92.30**
BiLSTM+RNN+LR+SVM+Naive Bayes Classifier（集成）	90.62

5.2 结果分析

- 对于单模型我们在BERT模型中得到了最优结果。
- 半监督学习对BERT模型有着略微的提升。
- 使用双层的LSTM模型比仅仅使用单向的LSTM模型有着较大的提升效果。
- SVM模型由于数据集较大，收敛较慢，结果也并不是很好。
- 逻辑回归模型在传统机器学习方法中表现优异。

- 不同模型的集成能起到比单模型更加好的效果。

六、总结与展望

这次情感分析作业，我们使用了传统机器学习模型也在此基础上使用了深度学习的方法。两种方法各有优点，传统机器学习方法计算速度快，而深度学习方法往往需要很长的时间才能够收敛。此外我们使用了集成学习和半监督学习方法，也对最后结果有着较好的提升。不过我们在数据预处理的时候没有很好的对数据进行分析，去除停用词也略显粗糙，希望在以后的任务中我们能够做的更好。