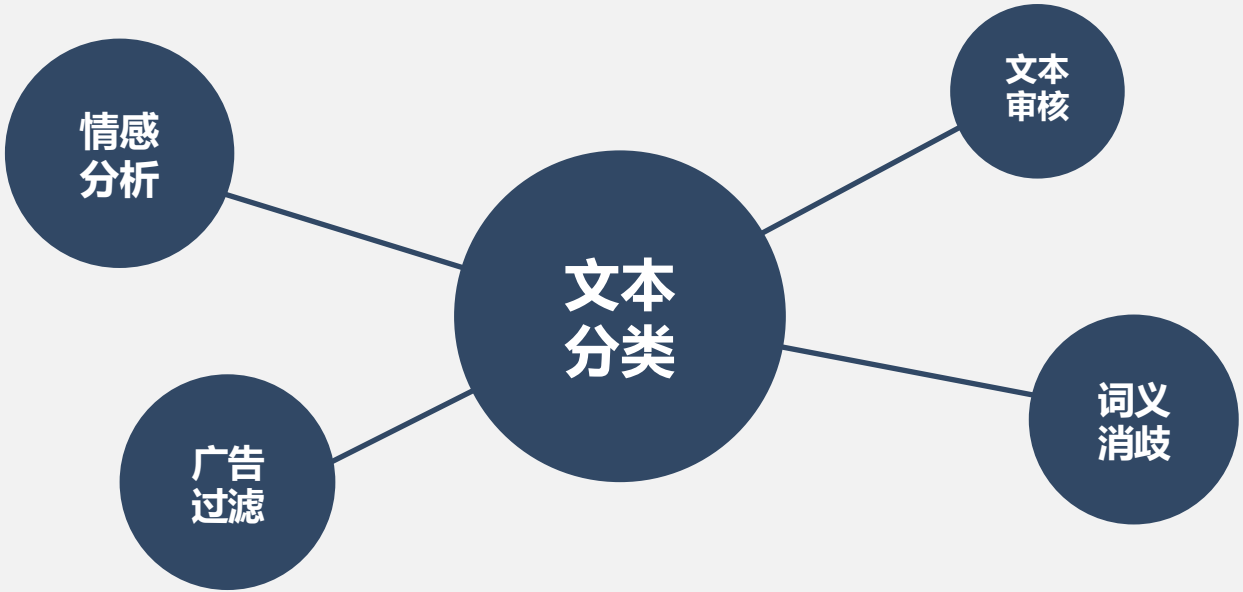
文本分类

汇报人　唐雨馨　李莹莹　黄宇婷　王铎　毛瑜琦

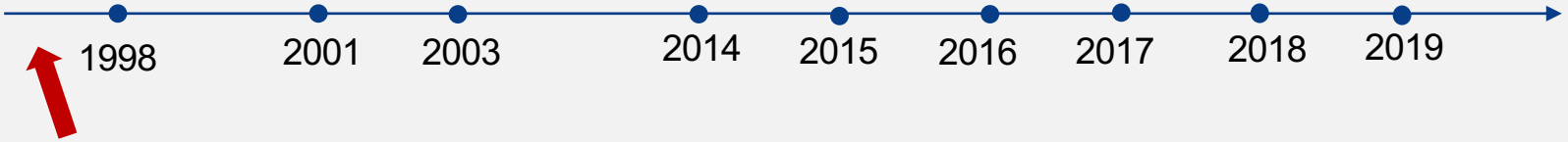
文本分类指的是计算机通过算法对输入的文本按照一定的类目体系进行自动化归类的过程。



简介

分类体系	新闻类别、评价好坏
分类对象	长文本、短文本
分类模式	二分类、多分类、多标签分类

发展历史



分类规则：人工定义

例句：毛毛喜欢踢足球

↓

规则：足球属于体育类

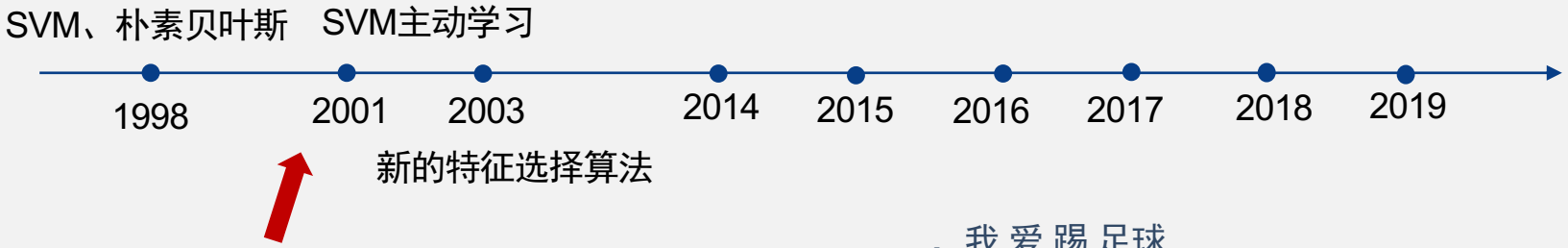
↓

类别：体育

文本特征：手工提取

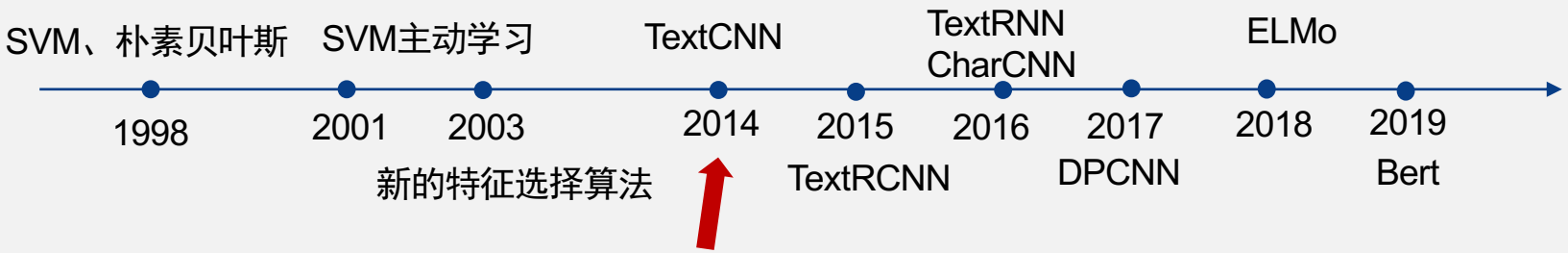
依赖专家系统，耗时耗力

发展历史



特征工程不能很好的表示语义信息

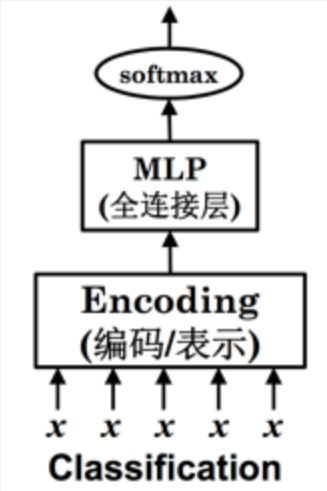
发展历史

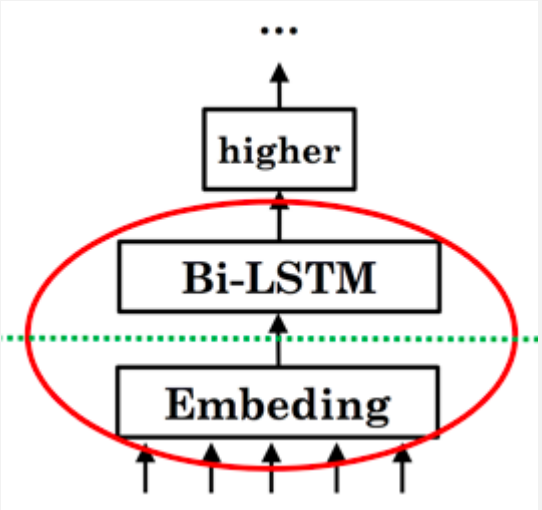


分布式表示

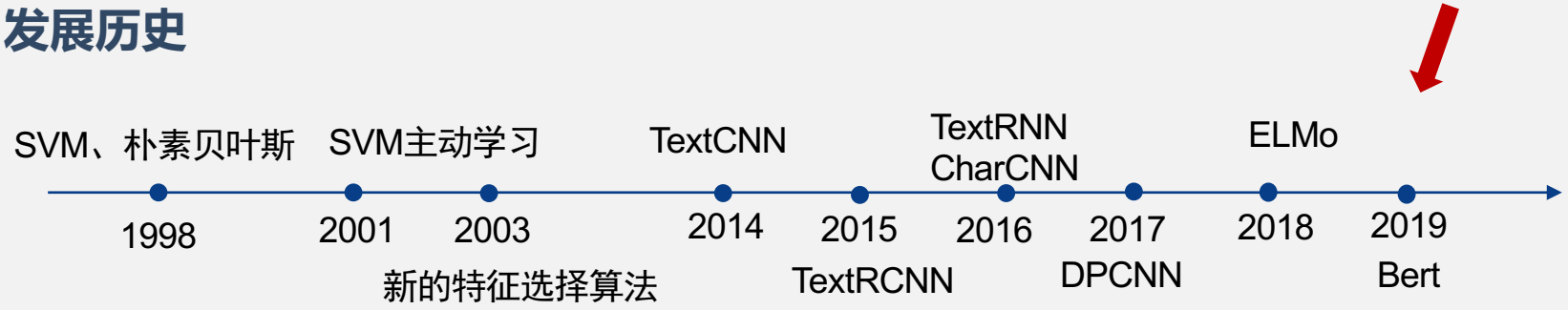
+

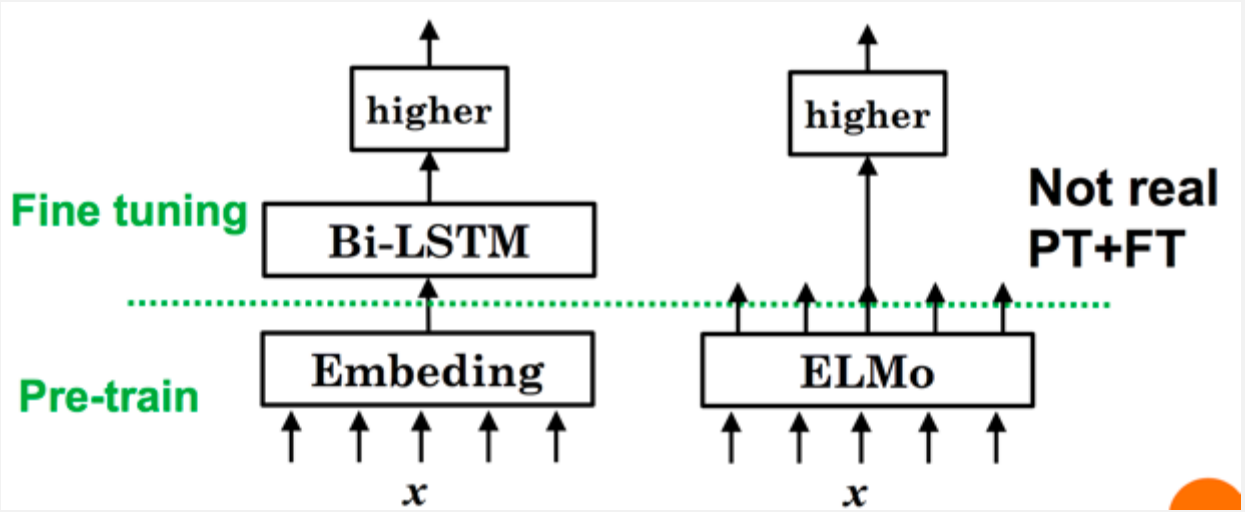
神经网络





发展历史


SVM、朴素贝叶斯
SVM主动学习
TextCNN
TextRNN
CharCNN
ELMo
1998
2001
2003
2014
2015
2016
2017
2018
2019
新的特征选择算法
TextRCNN
DPCNN
Bert


higher
Fine tuning
Bi-LSTM
Pre-train
Embeding
x
higher
Not real
PT+FT
ELMo
x

添加标题内容

01 传统的文本分类

文本表示

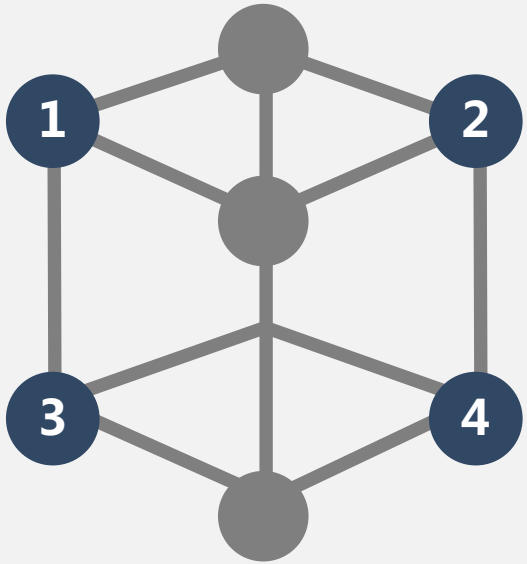
特征提取

02 深度学习文本分类

fasttext

TextCNN

TextRNN+Attention



03 对比实验

基于深度学习方法的对比

预训练语言模型

04 最新工作

可能也不是很新

传统的文本分类方法

In [1]:
```
import numpy as np
import sklearn
```

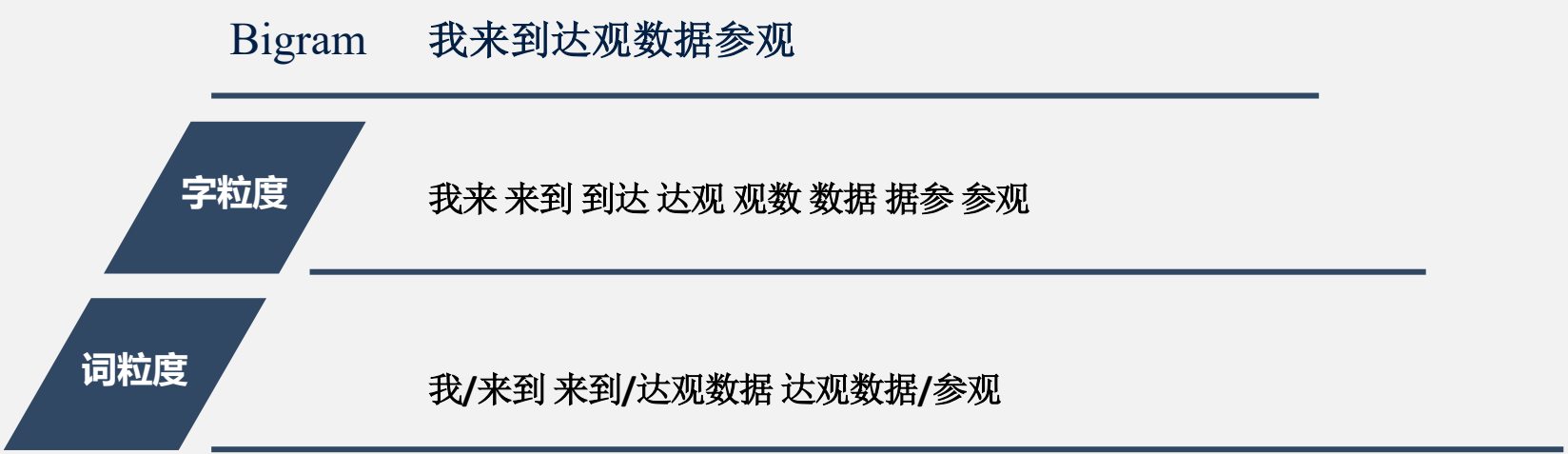
In [2]:
```
from sklearn.datasets import fetch_20newsgroups
train = fetch_20newsgroups(subset = 'train',shuffle=True)
test = fetch_20newsgroups(subset = 'test',shuffle=True)
train.target_names
```

Out[2]:
```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

N-gram 特征

在文本特征提取中，常常能看到n-gram的身影。它是一种基于语言模型的算法，基本思想是将文本内容按照字节顺序进行大小为N的滑动窗口操作，最终形成长度为N的字节片段序列。n-gram产生的特征只是作为文本特征的候选集。

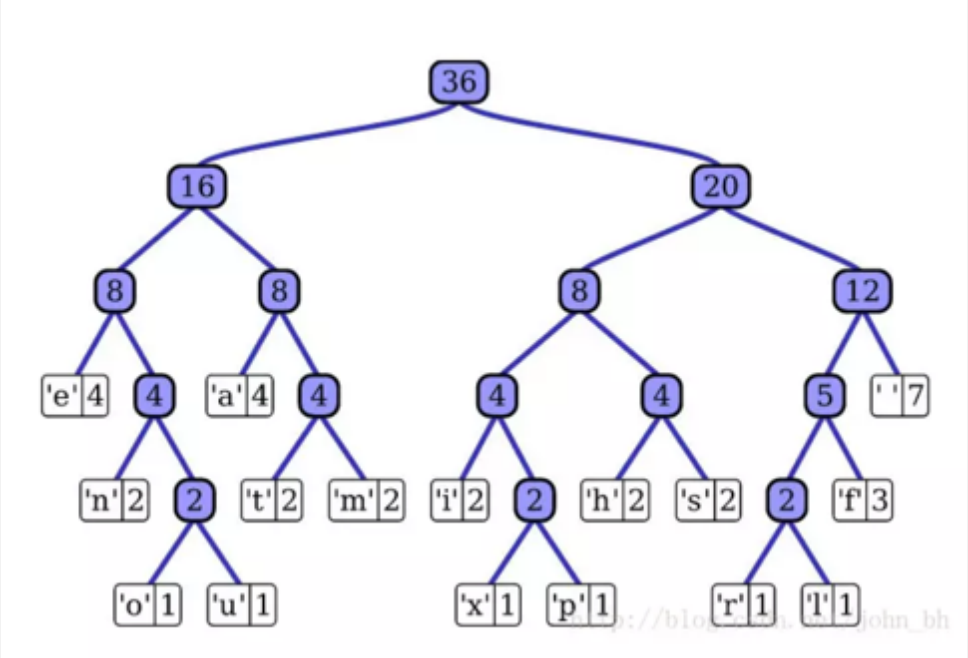
n-gram中gram根据粒度不同，有不同的含义。它可以是字粒度，也可以是词粒度。



层次 SoftMax

对于有大量类别的数据集，Fast Text使用了一种分层分类器（而非扁平式架构），即层次 SoftMax技巧。层次SoftMax技巧建立在哈弗曼编码的基础上，对标签进行编码，能够极大地缩小模型预测目标的数量。

Fast Text 也利用了类别不均衡这个事实（一些类别出现次数比其他的更多），通过使用 Huffman 算法建立用于表征类别的树形结构(Huffman树)。因此，频繁出现类别的树形结构的深度要比不频繁出现类别的树形结构的深度要小，这也使得进一步的计算效率更高。



Text CNN

卷积神经网络的核心思想是捕捉局部特征，对于文本来说，局部特征就是由若干单词组成的滑动窗口，类似于N-gram。卷积神经网络的优势在于能够自动地对N-gram特征进行组合和筛选，获得不同抽象层次的语义信息。

Text CNN 优势：模型简单，训练速度快，训练效果不错

Text CNN 缺点：模型可解释性不强，没有类似GBDT模型中特征重要度概念，在调优模型时，很难根据训练的结果去做针对性的调整具体。

DEMO

对弹幕数据进行正负评价分类

Demo背景： 随着自媒体短视频的盛行，我们常可以看到许多科技产品的评测视频。同时网友可以通过弹幕来表达对此产品的看法。通过网友的评价来判断产品在市场中的受欢迎度是非常重要的一个途径。

Demo目的： 给定一个耳机评测视频中弹幕的数据集，对弹幕进行正负评价分类，从而判断网友对此款耳机的喜爱程度。

DEMO 数据集

+1

307.946:AKG好评
326.529:这个好像比之前视频那个两百多的游戏耳机好点

437.721:有点超乎我的意料
443.865:死丢丢3可以的，beats里唯一能听的

446.322:???这不是挺好的吗
475.269:喜欢这个，干净清新不刺耳

-1

317.822:又蹂躏我的耳朵
332.257:突然干瘪 仿佛漏音

321.045:这是在闷包子啊？
411.106:抬走，一般，而且重的要死，低音炮水平

430.705:2899？感觉不太行阿，不如Sundaram。。
540.892:高音糊，低音渣。下一个


```
{ "label" : "1" , "text" : "AKG好评" }
{ "label" : "0" , "text" : "xelento已就绪" }
{ "label" : "0" , "text" : "SR60" }
{ "label" : "0" , "text" : "TFZKing就绪" }
{ "label" : "0" , "text" : "香格里拉洗脑，再无能入耳的声音" }
{ "label" : "0" , "text" : "MSR7戴上" }
{ "label" : "0" , "text" : "ar3bt" }
{ "label" : "0" , "text" : "没有人妻我自带了。" }
```


DEMO 配置项

```
数据的预处理
read_file(): 读取文件数据;
build_vocab(): 构建词汇表，使用字符级的表示，这一函数会将词汇表存储下来，避免每一次重复处理;
read_vocab(): 读取上一步存储的词汇表，转换为{词：id}表示;
read_category(): 将分类目录固定，转换为{类别：id}表示;
to_words(): 将一条由id表示的数据重新转换为文字;
preocess_file(): 将数据集从文字转换为固定长度的id序列表示;
batch_iter(): 为神经网络的训练准备经过shuffle的批次的数据。
```

```python
class TCNNConfig(object):
    """CNN配置参数"""

    embedding_dim = 64  # 词向量维度
    seq_length = 600  # 序列长度
    num_classes = 2  # 类别数
    num_filters = 256  # 卷积核数目
    kernel_size = 5  # 卷积核尺寸
    vocab_size = 5000  # 词汇表达小
    hidden_dim = 128  # 全连接层神经元
    dropout_keep_prob = 0.5  # dropout保留比例
    learning_rate = 1e-3  # 学习率
    batch_size = 24  # 每批训练大小
    num_epochs = 10  # 总迭代轮次
    print_per_batch = 50 #每多少轮输出一次结果
    save_per_batch = 10  # 每多少轮存入tensorboard
```

DEMO 训练和测试

对数据集进行训练

```
Training and evaluating...
Epoch: 1
Iter:      0,  Train Loss:    0.7,  Train Acc:  41.67%,  Val Loss:    0.7,  Val Acc:  38.59%,  Time: 0:00:01 *
Iter:     50,  Train Loss:   0.52,  Train Acc:  66.67%,  Val Loss:   0.44,  Val Acc:  82.32%,  Time: 0:00:07 *
Epoch: 2
Iter:    100,  Train Loss:   0.41,  Train Acc:  79.17%,  Val Loss:   0.19,  Val Acc:  92.59%,  Time: 0:00:12 *
Epoch: 3
Iter:    150,  Train Loss:   0.11,  Train Acc:  95.83%,  Val Loss:   0.12,  Val Acc:  96.58%,  Time: 0:00:18 *
Epoch: 4
Iter:    200,  Train Loss:   0.12,  Train Acc:  95.83%,  Val Loss:  0.056,  Val Acc:  98.67%,  Time: 0:00:24 *
Epoch: 5
Iter:    250,  Train Loss:  0.081,  Train Acc:  95.83%,  Val Loss:  0.032,  Val Acc:  99.62%,  Time: 0:00:29 *
Iter:    300,  Train Loss: 0.0098,  Train Acc: 100.00%,  Val Loss:   0.03,  Val Acc:  99.62%,  Time: 0:00:35
```

DEMO 训练和测试

对数据集进行测试

```
Test Loss:   0.28, Test Acc:  88.34%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

           1       0.83      0.92      0.87       241
          -1       0.93      0.86      0.89       325

avg / total       0.89      0.88      0.88       566

Confusion Matrix...
[[221  20]
 [ 46 279]]
```

真赞！ ： 1
抬走 ： -1
这个声场太好了，下潜足，而且氛围好前面的太多 ： 1
音质很不错，值得买 ： 1
太贵了，这是抢钱吗？ ： -1
森海真的好平衡啊，解析度也喜欢 ： 1
这个耳机就是森海的笑话 ： -1
哇,对beats失望了，还买的同款，还是接受现实吧，项链 ： -1
万元户！！！！！！ ： -1
说实话这个我有点失望，枪声部分还没脑放君的好 ： -1
好评，拜亚动力 ： 1
什么鬼？ ： -1

TextRNN

RNN

Recurrent Neural Network

中文名为循环神经网络
适合分析序列数据，比如预测股价等

Text RNN

在自然语言处理领域
RNN可以将句子、文档、语音作为输入，进行自动翻译、情感分析、语音转文字
还可以用来作曲、作文、生成标题

TextRNN

比如有这样一句话，需要判断下划线处最可能出现的词：

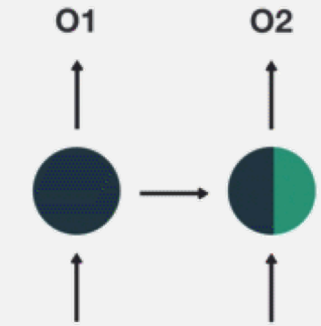
我昨天上学迟到了，老师批评了____。

对这句话进行分词：

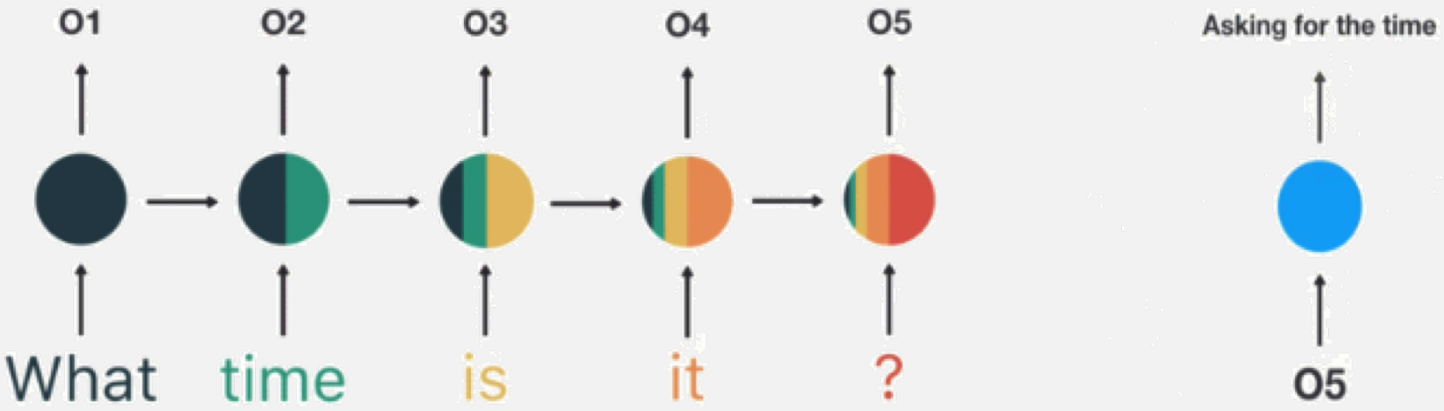
我 昨天 上学 迟到 了， 老师 批评 了 ____。

TextRNN用于文本分类

What time is it ?



TextRNN用于文本分类



Demo测试

预测值 / 真实值	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

precision = TP / (TP +FP)
recall = TP / (TP + FN)
f1-score = 2 / [(1 / precision) + (1 / recall)]

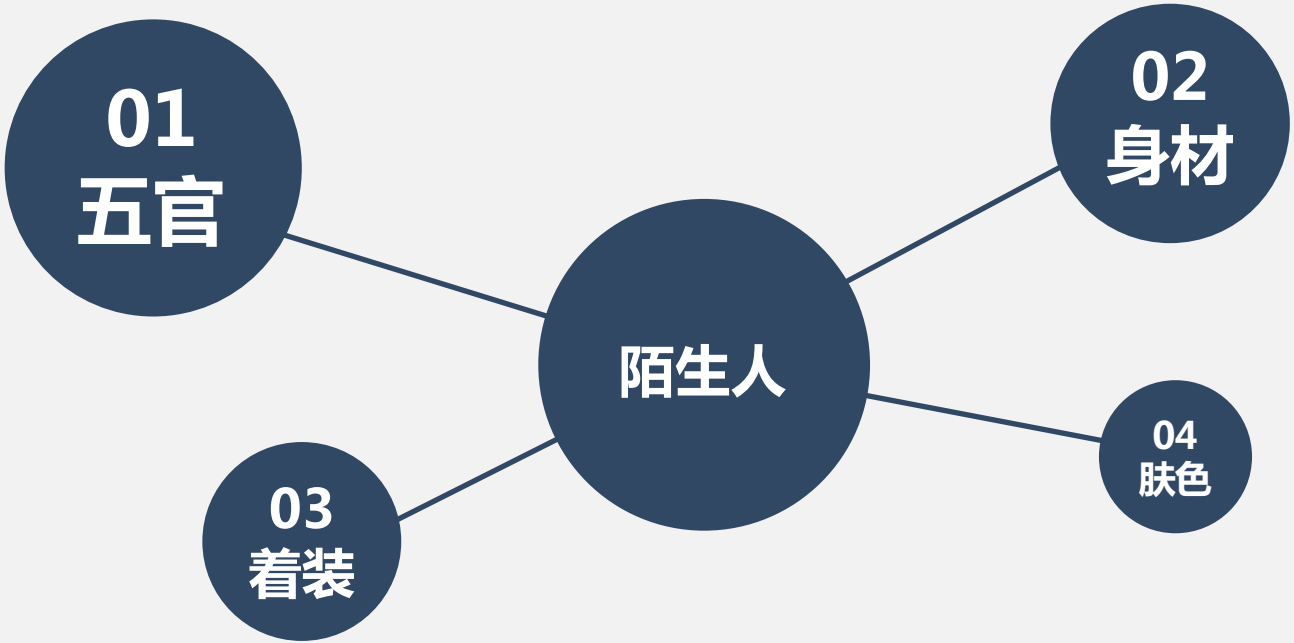
```
Test Loss:   0.27, Test Acc:  92.12%
Precision, Recall and F1-Score...
              precision    recall  f1-score   support

          体育       0.98      0.97      0.97      1000
          财经       0.87      0.99      0.92      1000
          房产       0.99      0.99      0.99      1000
          家居       0.95      0.68      0.79      1000
          教育       0.92      0.85      0.88      1000
          科技       0.88      0.96      0.92      1000
          时尚       0.85      0.97      0.91      1000
          时政       0.89      0.94      0.91      1000
          游戏       0.97      0.92      0.95      1000
          娱乐       0.94      0.94      0.94      1000
```

```
Confusion Matrix...
[[970   0   0   0   9   2  15   0   2   2]
 [  0 987   2   5   0   2   0   4   0   0]
 [  0   0 995   2   1   1   1   0   0   0]
 [  1  74   2 678  29  50  82  41   6  37]
 [  7  18   1   6 850  49   2  56   7   4]
 [  0   4   0  10   5 959  15   2   5   0]
 [  1   0   0   4   3   7 975   0   2   8]
 [  0  44   0   0  14   3   0 935   1   3]
 [  4   8   0   1   3  17  35   0 921  11]
 [ 10   2   0   5   8   5  17   8   3 942]]
Time usage: 0:03:35
```

TextRNN + Attention

Attention

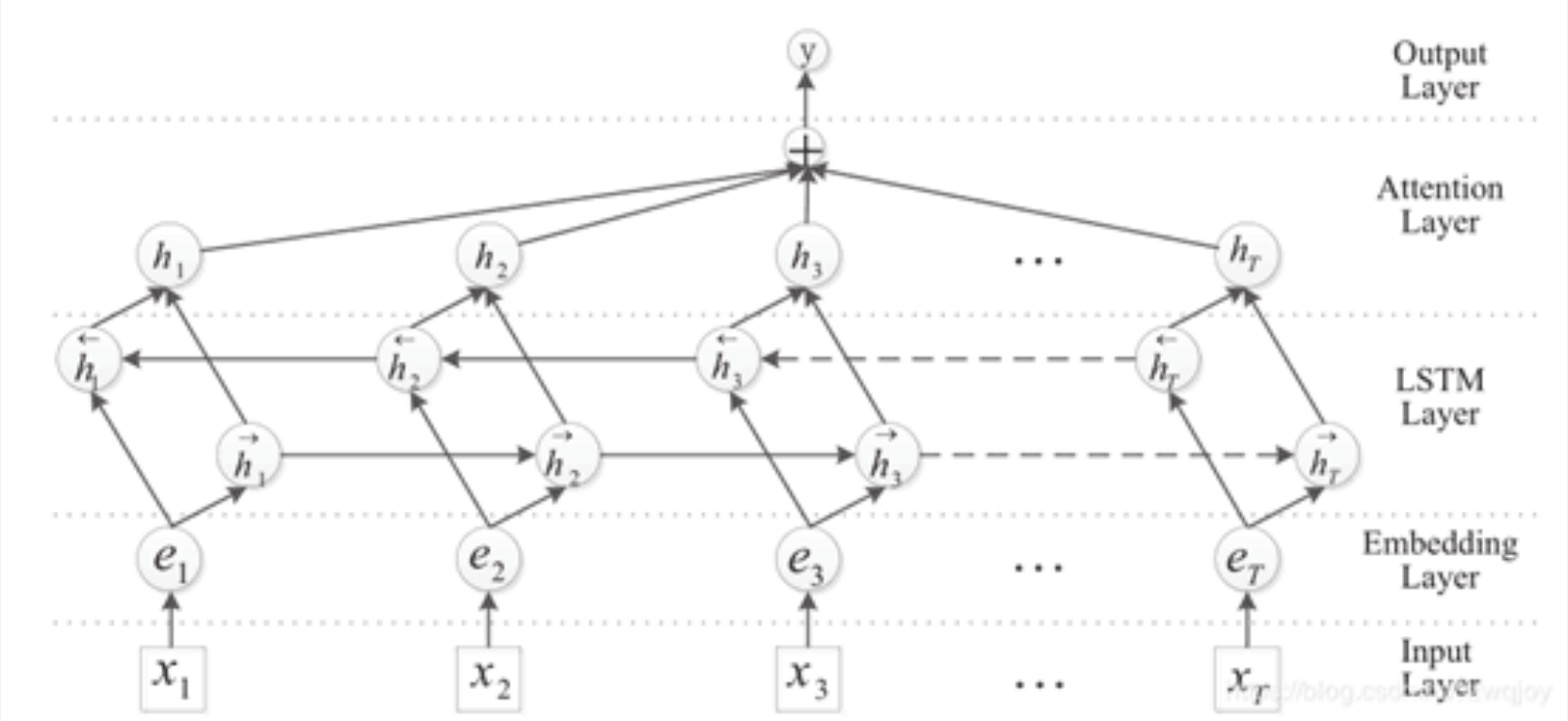


Attention机制与人类对外界事物的观察机制类似，当人类观察外界事物的时候，一般不会把事物当成一个整体去看，往往倾向于根据需要选择性的去获取被观察事物的某些重要部分。

Delicious food and the service was great

“加权求和”

BiLSTM+Attention



数据集

赵本山要追究法律责任：居然诽谤我打小沈阳	9
总有一款适合你 潮人必备多用途电视导购	4
万科退赛 恒大16.6亿抢下深圳建设集团	2
俄企业赴港上市热情不减 资源类企业积极性最高	2
《非诚勿扰》单亲妈妈邂逅沧桑型男 推出微直播	9
朱之文亮的都是绝活《GIGA SLAVE》	8
《非诚勿扰》“冯女郎”车晓带妈妈闯世界(图)	9
美弗吉尼亚大学访华太设计签实习基地协议（组图）	1

从THUCNews中抽取了20万条新闻标题，文本长度在20到30之间。一共10个类别，每类2万条。

类别：财经、房产、股票、教育、科技、社会、时政、体育、游戏、娱乐。

网络结构

```
def __init__(self, config):
    super(Model, self).__init__()
    if config.embedding_pretrained is not None:
        self.embedding = nn.Embedding.from_pretrained(config.embedding_pretrained, freeze=False)
    else:
        self.embedding = nn.Embedding(config.n_vocab, config.embed, padding_idx=config.n_vocab - 1)
    self.lstm = nn.LSTM(config.embed, config.hidden_size, config.num_layers,
                        bidirectional=True, batch_first=True, dropout=config.dropout)
    self.tanh1 = nn.Tanh()
    # self.u = nn.Parameter(torch.Tensor(config.hidden_size * 2, config.hidden_size * 2))
    self.w = nn.Parameter(torch.Tensor(config.hidden_size * 2))
    self.tanh2 = nn.Tanh()
    self.fc1 = nn.Linear(config.hidden_size * 2, config.hidden_size2)
    self.fc = nn.Linear(config.hidden_size2, config.num_classes)
```

网络结构

```
def forward(self, x):
    x, _ = x
    emb = self.embedding(x)  # [batch_size, seq_len, embeding]=[128, 32, 300]
    H, _ = self.lstm(emb)  # [batch_size, seq_len, hidden_size * num_direction]=[128, 32, 256]

    M = self.tanh1(H)  # [128, 32, 256]
    # M = torch.tanh(torch.matmul(H, self.u))
    alpha = F.softmax(torch.matmul(M, self.w), dim=1).unsqueeze(-1)  # [128, 32, 1]
    out = H * alpha  # [128, 32, 256]
    out = torch.sum(out, 1)  # [128, 256]
    out = F.relu(out)
    out = self.fc1(out)
    out = self.fc(out)  # [128, 64]
    return out
```

实验结果

```
Epoch [5/10]
Iter:   5700,  Train Loss:  0.31,  Train Acc: 92.19%,  Val Loss:  0.32,  Val Acc: 90.52%,  Time: 1:47:14
Iter:   5800,  Train Loss:  0.11,  Train Acc: 95.31%,  Val Loss:  0.33,  Val Acc: 90.15%,  Time: 1:49:06
Iter:   5900,  Train Loss:  0.18,  Train Acc: 93.75%,  Val Loss:  0.33,  Val Acc: 89.84%,  Time: 1:50:57
Iter:   6000,  Train Loss:  0.18,  Train Acc: 93.75%,  Val Loss:  0.32,  Val Acc: 90.57%,  Time: 1:52:49
Iter:   6100,  Train Loss:  0.26,  Train Acc: 92.19%,  Val Loss:  0.31,  Val Acc: 90.72%,  Time: 1:54:40
Iter:   6200,  Train Loss: 0.089,  Train Acc: 96.88%,  Val Loss:  0.33,  Val Acc: 90.40%,  Time: 1:56:32
Iter:   6300,  Train Loss:  0.11,  Train Acc: 96.88%,  Val Loss:  0.33,  Val Acc: 90.30%,  Time: 1:58:23
Iter:   6400,  Train Loss:  0.13,  Train Acc: 94.53%,  Val Loss:  0.33,  Val Acc: 90.58%,  Time: 2:00:14
Iter:   6500,  Train Loss:  0.25,  Train Acc: 92.19%,  Val Loss:  0.33,  Val Acc: 90.47%,  Time: 2:02:06
No optimization for a long time, auto-stopping...
Test Loss:   0.3,  Test Acc  90.78%
Precision, Recall and F1-Score...
               precision    recall  f1-score   support

      finance     0.9112    0.8720    0.8912      1000
       realty     0.9326    0.9130    0.9227      1000
       stocks     0.8315    0.8490    0.8402      1000
    education     0.9416    0.9510    0.9463      1000
      science     0.8582    0.8470    0.8525      1000
      society     0.9017    0.9080    0.9048      1000
     politics     0.8600    0.8970    0.8781      1000
       sports     0.9737    0.9640    0.9688      1000
         game     0.9537    0.9270    0.9402      1000
entertainment     0.9188    0.9500    0.9341      1000

  avg / total     0.9083    0.9078    0.9079     10000

Confusion Matrix...
[[872  20  72   4   9   7  14   2   0   0]
 [  9 913  24   2   8  14  13   4   3  10]
 [ 43  17 849   2  41   3  37   1   4   3]
 [  0   1   1 951   6  14  13   2   0  12]
 [  8   6  38  15 847  15  32   2  22  15]
 [  2  14   4  15  10 908  22   1   6  18]
 [ 11   2  23  12  14  27 897   2   2  10]
 [  4   1   2   3   2   9   6 964   0   9]
 [  3   1   7   3  41   4   5   2 927   7]
 [  5   4   1   3   9   6   4  10   8 950]]
Time usage: 0:00:25
```

对比实验

数据集	清华新浪新闻标题数据集
训练集	180000
验证集	10000
测试集	10000

分类：财经、房产、股票、教育、科技、社会、时政、体育、游戏、娱乐

对比实验

Method	Val	Test
	Acc(%)	
FastText	90.23	87.23
TextCNN	91.22	88.17
TextRNN	91.12	88.21
TextRNN + Attention	90.9	87.62
TextRCNN	91.54	88.34
DPCNN	91.25	87.94
Transformer	89.91	86.9
Bert	94.83	91.66
ERNIE	94.61	91.53