实验二：电商评论观点挖掘

摘要

本文采用……

1 引言

命名实体识别（NER）是信息提取的一个子任务，旨在将文本中的命名实体定位并分类为预先定义的类别，如人员、组织、位置等。而一个命名实体就是一个词语或是一个短语，它能够清晰地将一个物体和与他有相似属性的物体区分开来。

NER 是 NLP 中一项基础性关键任务，对于其他的自然语言任务如关系抽取、事件抽取、知识图谱、机器翻译、问答系统都需要 NER 的支撑。

本次实验“步骤一：属性词-情感词识别”需要采用 NER 技术。

步骤二与步骤三可以归为分类任务。传统的机器学习分类算法有：SVM，Logistic Regression，朴素贝叶斯，决策树等。但对于文本分类来说，最重要的是如何将一句话映射到向量空间，同时保持其语义特征。这就涉及到词嵌入（Word Embedding）技术。该技术也广泛应用于 NER 中。

2 实体识别相关研究工作

NER 任务的输入为一个序列：$s =< w_1, w_2, \ldots, w_n >$。

对于其中的每一个实体，输出一个元组列表$< I_s, I_e, I_t >$，其中I_s是实体起始位置，I_e是实体的结束位置，而I_t是这个实体的类型，如图 2.1.

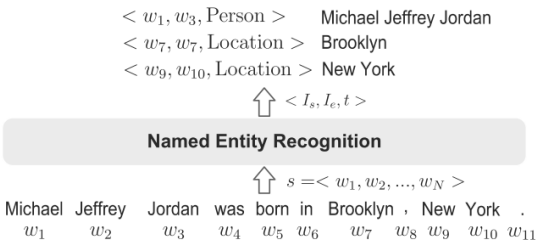


图 2.1 NER 识别出一个 Person 和两个 Location 实体

一般来说，命名实体识别有四种方法：

- 基于规则：传统的 NER 方案是基于手动归纳的规则识别，基于领域词典与语法规则。不需要标注数据，但是制定规则和维护都很麻烦，而且迁移成本高。

- 无监督学习：基于聚类的 NER，通过上下文的相似度抽取，不需要标注数据，但是准确度一般。

- 基于特征的监督学习：需要给定标注数据，并配合以精巧设计的特征。通过 ML 算法就能从数据中识别一些相似的 pattern。常用的特征有：词语级别的特征如大小写，词态学以及词性标注等等，通过特征工程，将文本中的词使用一个或多个 bool 类型或数值类型等表达为一个特征向量。有了特征，许多机器学习算法就能用于监督式的 NER 上，如隐马尔科夫模型（HMM）、决策树、最大熵、支持向量机（SVM）以及条件随机场（CRF）等。

- 基于深度学习：需要标注数据，但是可以自动学习特征。深度学习方法有三点好处：第一，由于深度学习的高度非线性，相比于传统的线性模型（线性 HMM 和线性链 CRF），深度学习模型能够学习到更复杂的特征；第二，深度学习能够自动学习到对模型有益的特征，传统的

机器学习方法需要繁杂的特征工程，需要大量的工程技巧以及领域知识，深度学习不需要；第三，深度学习可以端到端的搭建模型，通过梯度下降的方法去训练学习，这就允许我们设计更复杂的 NER 系统。

深度学习的模型一般为三层结构，如图 2.2. 第一层为输入的分布式表示（Distributed representations for input），基于 char 或 word 嵌入的向量，同时辅以词性标签（POS），gazetter 等人工特征。第二层为上下文编码器（Context encoder），该层通过 CNN，RNN，语言模型 LM，Transformer 等网络获取词义依赖。第三层是标签解码器（Tag decoder），预测输入序列对应的标签，常用的如 Softmax，CRF，RNN，指针网络（Point Network）。

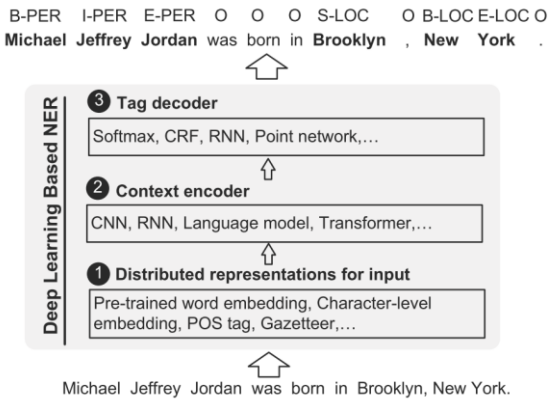


图 2.2 基于 DL 的 NER 三层模型

分布式词表示通过把词映射到低维空间的稠密实值向量，其中每个维度都表示隐含的特征维度。一般 NER 系统的输入有三种表示：word-level，char-level 和混合表示。

词语级别（word-level）的词向量，可以通过 CBOW 或者 skip-gram 的方式训练得到，常用的词嵌入有：Google 的 word2vec，斯坦福的 Glove，Facebook 的 fastText 以及 SENNA。字符级别（char-level）的表示能够显式利用子词级别（如前缀和后缀）的信息，也能够缓解未登录词的问题。可以采用 CNN 或者 RNN 获得字符级别的嵌入表示。

上下文编码器通过接收第一层的嵌入向量来学习编码。这一层有四类方法：CNN，RNN，递归神经网络，Transformer。

RNN 的两种变体：LSTM 和 GRU 常常用来编码 context。其中，双向的 RNN 网络能够有效地综合过去与未来的信息，即前向与后向。理论上说，将 token 经过一个双向的 RNN 编码之后就应该包含整个句子的信息。在文本经过了 RNN 编码后，会再经过一个 CRF 层，这样就可以解码出每个词所对应的标注。本次实验的步骤一，在上下文编码层和标签解码器层，即采用 BiLSTM+CRF 方案。

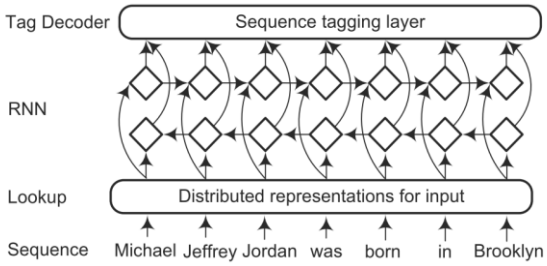


图 2.3 双向 RNN

Google 的 BERT 模型是 Transformer 的一个变体。通过预训练得到模型，再做微调就可以应用到很多 NLP 任务上。

标签解码器是 NER 模型的最后一层，输入一个与上下文有关的表示，然后生成一个和输入的序列对应的标签序列。常见的解码方式为：MLP+Softmax，CRF，RNN 等。

NER 可以被看作一个序列标签问题，如果使用多层感知机（MLP）+Softmax 作为标签解码器，就可以直接把序列标注问题转化为一个多分类问题。

条件随机场（CRF）可以利用全局信息进行标记。CRF 在基于特征的监督式学习算法中就已经被广泛地使用了。很多深度神经网络也是使用 CRF 层作为标签的解码器。

3 实验步骤与结果分析

3.1 属性词-情感词识别

如图 3.1.1，文献表明采用 CNN + BiLSTM + CRF 三层模型，算法正确率已经达到 97%以上。其中，CNN 做字符级别的编码，主要解决未登录词问题，并把词映射为向量；LSTM 用来编码上下文；CRF 能够记住实体序列的规则，纠正 LSTM 的一些低级错误。

由于实验时间、实现难度等因素，本次实验采用的模型结构为 BiLSTM+CRF。

另外，实验选用 Keras 来构建深度学习模型，并主要使用了序贯（Sequential）模型。

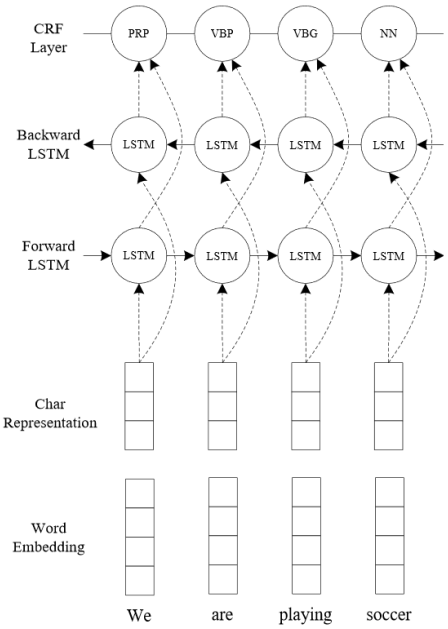


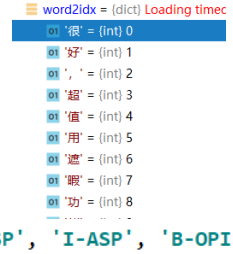
图 3.1.1 CNN+BiLSTM+CRF

首先对数据预处理，使用训练集中的 Train_labels.csv 对 Train_reviews.csv 做“BIO”标注。其中，“B”为实体的开始，“I”为实体的中间，“O”表示非实体。并用“OPI”和“ASP”标识情感词和属性词。预处理结果如图 3.1.2.

很 B-OPI
好 I-OPI
， O
遮 B-ASP
暇 I-ASP
功 I-ASP
能 I-ASP
差 B-OPI
一 I-OPI
些 I-OPI
， O
总 O
体 O

图 3.1.2 “BIO”标注训练数据

为了将文本转化为向量形式，从而可以喂给模型，要根据输入文本构建单字词典，并对一段文本中每个字查找其在词典中的序号。例如输入为：[很好，超值，很好用]



['O', 'B-ASP', 'I-ASP', 'B-OPI', 'I-OPI']

图 3.1.3 词典及标签

由图 3.1.3 可知，句子被向量化为：[0,0…,0,1,2,3,4,2,0,1,5]。如果五个标签依次对应数字“0-4”，句子的 BIO 标注序列被向量化为[-1,-1,…,-1,3,4,0,3,4,0,3,4,4].

每个句子作上述处理，输入模型，并进行训练，如图 3.1.4.

```
def create_model(train=True):
    if train:
        (train_x, train_y), (vocab, chunk_tags) = parse_data.load_data()
    else:
        with open('model/config.pkl', 'rb') as inp:
            (vocab, chunk_tags) = pickle.load(inp)
    model = Sequential()
    model.add(Embedding(len(vocab), EMBED_DIM, mask_zero=True))  # Random en
    model.add(Bidirectional(LSTM(BiRNN_UNITS // 2, return_sequences=True)))
    crf = CRF(len(chunk_tags), sparse_target=True)
    model.add(crf)
    model.summary()
    model.compile('adam', loss=crf.loss_function, metrics=[crf.accuracy])
    if train:
        return model, (train_x, train_y)
    else:
        return model, (vocab, chunk_tags)
```

图 3.1.4 模型构建

模型训练好之后，进行预测，对新文本向量化。输入文本依然为：[很好，超值，很好用]，输出的初始矩阵如图 3.1.5.

	0	1	2	3	4
0	1.00000	0.00000	0.00000	0.00000	0.00000
1	0.00000	0.00000	0.00000	0.00000	1.00000
2	1.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000	1.00000	0.00000
4	0.00000	0.00000	0.00000	0.00000	1.00000
5	1.00000	0.00000	0.00000	0.00000	0.00000
6	1.00000	0.00000	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000	0.00000	1.00000
8	0.00000	0.00000	0.00000	0.00000	1.00000

图 3.1.5

对每一个字都有属于五个标签的概率，选最大的，从而得到输出序列[0,4,0,3,4,0,0,4,4].

这句话有三个 opinion，分别为“好”、“超值”和“好用”，但是很明显这里有问题。“44”的序列仅有中间字，没有开头字，所以这里需要额外处理。

3.2 属性分类

这一节首先采用 Word2Vector 技术训练词向量，然后使用 BiLSTM 技术做特征的表示学习，最后一层为 Softmax 层。模型结构如图 3.2.1.

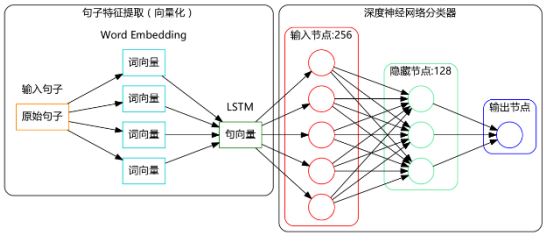


图 3.2.1 使用 LSTM 分类

首先，对训练集的 Train_labels.csv 文件预处理，需要构造三个值：1）asp_opi_combined：每一行的 aspect 和 opinion 合并为一句话，作为 list 数据结构的元素；2）category_labels：是一个 list，元素为每一行对应的标签，如“整体”、“价格”等；3）word_list：list，元素为每一行的 aspect 以及 opinion.

```
asp_opi_combined = {list: 6633} ['很好', '超值', '很好用', '很好', '遮暇功能差一
category_labels = {list: 6633} ['整体', '价格', '整体', '整体', '功效', '整体', '包装',
polarity_labels = {list: 6633} ['正面', '正面', '正面', '正面', '负面', '正面', '负面', '
word_list = {list: 8359} ['很好', '超值', '很好用', '很好', '遮暇功能', '差一些', '还不
```

图 3.2.2 预处理

接着，将标签表示为数字 0-12，将 word_list 作为 Word2Vec 模型的输入，将一个词映射成一个 100 维的向量，并将模型保存到本地。如图 3.2.3.

```
def train_word2vec():
    asp_opi_combined, category_labels, polarity_labels, word_list \
        = get_asp_opi_combined_labels()
    category_term = {'包装': 0, '成分': 1, '尺寸': 2, '服务': 3, '功效': 4,
                     '价格': 5, '气味': 6, '使用体验': 7, '物流': 8, '新鲜度': 9,
                     '真伪': 10, '整体': 11, '其他': 12}
    pol_labels = [category_term.get(word) for word in category_labels]
    if not os.path.exists(file_path_word2vec):
        model = Word2Vec(word_list, size=100, min_count=5, window=5)
        model.save(file_path_word2vec)
    else:
        model = Word2Vec.load(file_path_word2vec)
    return asp_opi_combined, pol_labels, model
```

图 3.2.3 Word2Vec 模型

由上操作我们得到了 Word2Vec 的 model，这样就能从模型中提取模型词典（w2index）和词向量矩阵 embedding_weights。

w2index 为一个 dict，元素格式为：“word：id”，而 embedding_weights 为 list，元素为行向量，每一个向量与 w2index 的 word 相对应。

构建模型需要把输入文本和标签都转化为向量格式，借助 w2index，可以实现。

原始数据为：

sent：遮瑕功能差一些

labels：功效

输出结果为：

$x = [0,0,0 \ldots,0,271,137,0,228,87,1,12]$

$y = [0,0,0,0,1,0,0,0,0,0,0,0]$

把每一句话如上数字化，作为训练语料，和词向量矩阵，一起喂给模型，训练模型：

```
def train_lstm(w2index, embedding_weights, x_train, y_train):
    n_epoch = 10
    Embedding_dim = 100
    model = Sequential()
    model.add(Embedding(output_dim=Embedding_dim,
                        input_dim=len(w2index) + 1,
                        mask_zero=True,
                        weights=[embedding_weights],
                        input_length=100))
    model.add(Bidirectional(LSTM(50), merge_mode='concat'))
    model.add(Dropout(0.5))
    model.add(Dense(13, activation='softmax'))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    if not os.path.exists(file_path_lstm):
        model.fit(x_train, y_train,
                  batch_size=32,
                  epochs=n_epoch,
                  verbose=1,
                  validation_split=0.04)
        model.save(file_path_lstm)
        score = model.evaluate(x_train, y_train, batch_size=32)
        print(score)
    else:
        model.load_weights(file_path_lstm)
    return model
```

图 3.2.4 模型构建和训练

模型训练完成可以保存在本地，在下一次直接加载。

然后是对新输入文本进行预测，同样，将文本转为向量，作为 model.predict()输入。如图 3.2.5.

```
def predict(w2index, model, seg_list):
    seg2id = [w2index.get(word, 0) for sen in seg_list for word in sen]
    sen_input = pad_sequences([seg2id], max_len)
    res = model.predict(sen_input)[0]
    return np.argmax(res)
```

图 3.2.5 对新文本预测

结果实例：

输入文本为“颜色明显”，模型预测每一个标签的概率，并从中取一个最大的作为标签，由图 3.2.6 可知，“4”号概率最大，正好对应“功效”，预测成功。

```
00 = {float32} 0.06842416
01 = {float32} 0.06799826
02 = {float32} 0.067998245
03 = {float32} 0.067998245
04 = {float32} 0.18354106
05 = {float32} 0.06805012
06 = {float32} 0.067998655
07 = {float32} 0.06799844
08 = {float32} 0.06799985
09 = {float32} 0.067998275
10 = {float32} 0.06799827
11 = {float32} 0.06799824
12 = {float32} 0.06799825
```

图 3.2.6 对“颜色明显”的各个标签预测概率

3.3 观点极性分类

与步骤二方案相同，唯一的不同是此处分类为 3 类，步骤二为 13 类。

需要注意的是，训练集各个类别标签数目相差较大：“正面”：5925，“负面”：556，“中性”：152，可能会对模型预测精度产生负面影响。

4 实验心得

深度学习模型很强大，只需输入足够规模标注数据，它就能够自己学习数据的特征，并且有很高的正确率。

深度学习模型实现非常复杂，对于初学者选用 Keras 可以屏蔽 Tensorflow 底层复杂实现，更快的完成模型的构建（但是这样会使得模型变为一个黑箱，我们难以理解其内部实现，进而影响优化时的参数设置）。

开源是个伟大的概念，感谢所有开源代码贡献者。

参考文献

[1] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li , “A Survey on Deep Learning forNamed Entity Recognition”, arXiv preprint arXiv: 1812.09449v1, 2018

[2] Xuezhe Ma and Eduard Hovy, “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”, arXiv preprint arXiv: 1603.01354v5, 2016

[3]