人工智能大作业报告

——Triplc Matching

狄战元 2016K8009929024
陈灿宇 2016K8009929018
李欢洋 2016K8009929042
杨　帆 2016K8009926003

目录

一、 任务描述

1.任务要求

给定一个 json 格式的描述公司过去几年财务状况的句子集合，提取符合句子语义的各个语句中的“times”，“attributes”与“values”组成的三元组并输出。

2.数据描述

课程提供的人工标注数据集包含 3000 条用户数据，每条数据包含 sentenceID、sentence、results、times、values、words、attributes、indexes 等语句信息。sentenceID 为语句的唯一身份号，sentence 为完整的公司财务语句，results 为人工标注的“times”、“attributes”与“values”组成的三元组信息，times 为 results 中的时间词语所在位置，values 为 results 中的金额词语所在位置，words 为语句中的所有词语所在位置，attributes 为 results 中的属性词语所在位置，indexes 为语句中所有词语在字典中的所属类别编号。

二、 实施方案

1.解决思路

通过对数据集的分析，我们认为这是一个序列建模关系抽取问题。首先，由于财务语句本身有一定的格式与规则，所以我们可以利用这一点，总结归纳三元组的出现规律，进行初步的提取。进一步，循环神经网络和卷积神经网络一定程度上可以适用于序列建模关系抽取。在项目初期，我们决定首先进行语句语法规则的观察总结，利用规则对语句进行分类并输出相应三元组。接下来采用神经网络进一步学习我们没有考虑到的语句规则，针对选取的神经网络进行调参，进一步提高模型的预测准确率。

2.整体流程

整体流程简略概括如下：

（1）观察总结数据集中的规律，选取适合序列学习的神经网络。

（2）规则和训练结合提取文本中的三元组—Time, Attributes, Values。

（3）比较可能产生的结果（三元组），选取概率最大的作为最后的输出。

三、 具体实现

1.基于规则进行分类

首先，我们对训练数据集的句子成分进行标注：如果 word 对应 index 是 0，则标注为 T，指代 times;如果 word 对应 index 是 1，则标注为 A，指代 attributes;如果 word 对应 index 是 2，则标注为 V，指代 values。这样便于我们对语句中三元组格式进行分析。

我们针对该问题设计了一个短句的概念。因为整个句子的成分可能会非常复杂，由于规则的特殊性可能无法应对，所以我们在整个句子中划分短句，考虑短句的有效组成。（长句的干扰因素导致判断会非常多）

短句的概念由句子中的语义而来，对于长句，其实句子中有多个表达的语义，因此我们可以进行拆分，寻找句子中紧密相连（语义上）的部分合为同一个短句。这样一个短句中基本带有所要表达的 Attributes, Values 要素，这样可以方便进行下一步分析。

对于短语的划分处理，我们主要根据句子中的标点符号进行判别，对于标点前后语句的关系进行再判别，如果存在紧密联系，我们将之合并成一个短句，比如语句中出现 “value+逗号+value”，“逗号+分别+为”等类型格式的语句时，我们都认为前后语句都属于同一短句。

接下来，我们将语句中的 Time, Attributes, Values 进行组合，得到所有可能的三元组结果。对于每一种三元组组合，我们考虑其中的 Attributes, Values 成分，根据句中的特征对 Attributes, Values 是否在同一短句中进行判断，我们只考虑在同一短句中的组合。

针对所有可能的三元组结果，我们构造规律进行筛选，过滤掉错误结果。步骤如下：

（1）过滤掉含有诸如“大于”、“小于”、“增长”“降低”、“估计”、“了”、“以上”、“以下”等后接 values 的“假 attributes”词语的三元组。

（2）在过滤掉“假 attributes”后，根据三元组中 Time, Attributes, Values 顺序，我们需要过滤掉不符合 TAV 这个顺序的三元组。

（3）过滤掉 Attributes 和 Values 不在同一个短句的三元组。判断是否在同一个短句的一个方式为：经过开头的部分逗号替换为顿号操作后，语句中相邻两个逗号间为一个有效短句，若 Attributes 和 Values 中间出现分号或分词等“断句词语”，在实际语义上会导致 Attributes 和 Values 无法构建联系。这种情况的三元组我们选择过滤掉。

（4）过滤掉从句首到 Attributes 之间以及 Values 到句末之间含“占比”、“平均”等特殊词语的三元组。

（5）由于语句中经常出现多个 Time+一个 Attributes+与 Times 数量相等的 Values 的情况，我们对余下的三元组根据所在语句中 Times 的数量进行 Time 与 Value 的匹配。

这样，我们经过层层过滤，得到了三元组，将这些三元组与数据集的 result 进行对比，统计并打印错误的语句。

2.基于 RNN 模型进行分类

我们尝试采取循环神经网络来进行学习，将问题转化为一个二分类问题。

首先我们通过构造出 3000 个训练数据的所有可能的三元组，即对每个句子 Time, Attributes, Values 组合的所有可能性排列组合，共 126208 条样本。其中 8929 为正样本，即对于这个句子和给定的三元组，该三元组出现在了句子的 results 中，即三元组有效，为正样本。相反的 117279 条为负样本，即该三元组对于这条句子无效。

之后我们处理了正样本和负样本之间的不平衡问题，正样本 8929 个，负样本 117279 个，采用了一种简单的方式，即在负样本中随机抽样作为训练数据，最终使得正负样本均在 8000 个左右。

每条样本中包含了句子的词信息，即句子中所有 words 的词向量，同时还需要包含信息来对给定的三元组进行标记，因为最终我们需要模型从每个样本中拿到句子和给定的三元组信息来判断是否为真。我们的处理方式是在词向量的基础上增加维数，我们使用 word2vec 训练出的 10 维词向量，在此基础上又增加了四维作为标记三元组的内容。

我们生成十维词向量。考虑到最长的语句包含 154 个词汇，我们把输出构建为一个 154*14 的矩阵，其中十四维中包含四维用于标记三元组是否落在 results 中：如果落在 results 中，这四维赋值为 10；反之赋值为 0。模型方面，我们选择训练数据集的 30%作为验证集，采用深度 RNN

网络,使用交叉熵计算残差,对于每一个长度为 n 的序列[x1,x2,x3,...,xn]的每一个 xi,都会在深度方向跑一遍 RNN,跑上 hidden_num 个隐层单元。

3.基于 CNN 模型进行分类

利用 CNN 方法处理三元组提取问题

首先，三元组提取可以转化为一个二分类问题。因为句子中的"times", "attributes", "values"信息已经提取出来了，所以三元组的可能的组合方式是有限的。对于每一种组合方式，可以结合句子信息对其进行二分类，所以我们考虑将三元组二分类的过程转化为对句子进行二分类的过程。

这里就需要对原始的句子进行处理，对于每一种选定的三元组的组合方式，可以有两种处理方法：第一种方法是在原始的句子中将选定的三元组“MASK”住，其他部分保留，如果这个三元组是正确的，则将这个句子的 label 标为 1;如果这个三元组是错误的，则将这个句子的 label 标为 0。第二种处理方法是在原始的句子中将原始的句子中将选定的三元组以外的"times", "attributes", "values"信息“MASK”住，其他部分保留，如果这个三元组是正确的，则将这个句子的 label 标为 1;如果这个三元组是错误的，则将这个句子的 label 标为 0。

在处理完之后，就可以将处理后的句子中的各个词映射为词向量，将整个句子作为输入，经过输入层 ，卷积层 ，全连接层 ，输出层 。得到二分类的结果。

这两种方法我都尝试了一下，提交的版本中是采用的第一种方法，预测的结果可以见 prediction.csv。

虽然对于处理后的句子集的预测准确率可以达到 0.929556,但是由于在处理后的句子集中负样本占绝大多数，所以对正样本的预测准确率其实很低，这也是我们最终没有采用这种方法的原因之一。

那么为什么会想到利用 CNN 呢？这里参考了 Yoon Kim 的 paper：Convolutional Neural Networks for Sentence Classification (EMNLP 2014）

在 RNN 之外，利用 CNN 来处理句子分类的问题其实是一种比较经典的方法。CNN 虽然出身于图像处理，但是它的思路，给我们提供了在 NLP 应用上的参考。“卷积”这个术语本身来自于信号处理，简单地说就是一系列的输入信号进来之后，系统也会有一系列的输出。但是并

不是某一时刻的输出只对应该时刻的输入，而是根据系统自身的特征，每一个时刻的输出，都和之前的输入相关。那么如果文本是一些列输入，我们当然希望考虑词和词的序列特征，比如“Tom 的 手机 ”，使用卷积，系统就会知道“手机是 tom”的，而不是仅仅是一个“手机”。

CNN 的具体网络设计可以参见代码，运行的方法可以参见 README.md

后续进一步的思考方向是增加映射为词向量时的特征维度，完善网络的设计等等。

四、 实验结果及分析

在 3000 条数据的训练数据集上，我们最终生成三元组中错误数量为 58 个，准确率约为 98.1%。最终结果为：recall:0.99 prec:0.99 correct:0.98 F1:0.99。

我们最终采用了规则的方式来进行判断，神经网络模型的结果并不理想。最初的想法是先有规则来进行判断（预想中规则要比神经网络模型好，所以打算以规则为主，神经网络模型为辅），规则判断不了的交由神经网络模型再次进行计算，最终的结果应该会比单独使用规则的方式再好上一些。但是在实际过程中，一是很难断定规则能否判定一个数据，即无法找出超出规则判断能力的数据，我们也就不知道对于一条数据究竟使用哪种方式进行判断。二是神经网络模型的运行结果并不理想，单个三元组预测正确率仅在 60%左右，整体预测起来会更低。所以我们最后只采用了规则的方式来进行预测，在训练集上准确率约为 98.1%。

我们的规则的建立过程不是一蹴而就的，而是在一次次运行程序过程中观察错误值、总结规律、增添规则、完善得到的，这也就使得我们的规则有全面准确的提取三元组能力。由于经验不足，循环神经网络的运用不够熟练，没有完全发挥出神经网络在序列建模中的作用，这是我们有待加强之处。

五、 任务分工

狄战元	李欢洋	杨 帆	陈灿宇（组长）
规则分析 针对数据集进行规则设计，筛选可行规则 **规则全部代码的编写** 使用短句模式的规则分析代码的全部编写工作。为**最终方案** **RNN 模型设计** 设计 RNN 的输入输出，进行数据预处理 **RNN 全部代码的编写** 使用 RNN 训练代码的编写工作。由于正确率不足，未采用 **数据整理** 记录、总结实验中的各项数据及其原因。 **撰写大作业报告** 最终大作业报告的撰写工作	**演示文稿** 演示文稿的制作，同时查阅相关资料 **查阅资料** RNN 相关内容的查阅 **规则分析** 针对数据集进行规则设计，筛选可行规则 **撰写大作业报告** 最终大作业报告的撰写工作	**演示文稿** 演示文稿的制作，同时查阅相关资料 **查阅资料** CNN 相关内容的查阅 **CNN 模型设计** 设计 CNN 的输入输出，辅助进行数据预处理	**课堂汇报** 1 月 12 日课堂展示汇报任务 **CNN 模型设计** 设计 CNN 的输入输出，进行数据预处理 **CNN 模型代码的编写** 使用 CNN 训练代码的编写工作。由于正确率不足，未采用 **撰写大作业报告 CNN 部分**