哈尔滨工业大学

**<<信息检索>>**

**实验报告**

**(2019年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **王丁子睿** |
| **学号：** | **1183710211** |
| **学院：** | **计算机学院** |
| **教师：** | **张宇** |

实验二 问答系统设计与实现

## 实验目的

对问答系统的设计与实现过程有一个全面的了解，实验主要内容包括：

* 对给定的文本集合进行处理、建立索引。
* 找出问题的候选答案句并排序。
* 答案抽取，逐步调优。

## 实验内容

1. 文本集合进行处理、建立索引

对所有文档分词、分句，并建立索引，作为问答系统的检索语料。

1. 问题分类

训练一个问题分类模型，得到问题类别信息，然后将其融入到候选答案句排序和答案抽取的任务中，以取得更好的效果。

1. 候选答案句排序

对所有候选答案句按照其包含正确的可能性进行排序，可能性越大的越靠前。

1. 答案抽取

从候选答案句中抽取正确的答案。

## 三、实验过程及结果

实验代码采用python，机器学习包采用sklearn。

1. 文本集合进行处理、建立索引

数据存储格式位于utils/data.py文件中，其用于从文件中读写数据，保存每条数据的属性，并在数据集上应用各类学习器（分词、分类等）。

其各类方法如下所示：

    @staticmethod

    def load(name: str, path: str):

        """从json文件中加载数据

        Args:

            name: 数据集名称

            path: 数据路径

        """

    def dump(self, path: str):

        """将数据保存到json文件

        Args:

            path: 数据路径

        """

    def get(self, key: str) -> List:

        """提取键对应的属性

        Args:

            key: 属性

        Returns:

            查询属性

        """

    def retrieval(self, key: str, question: List[List[float]], pid: List[int] = None):

        """从数据库中查找和问题最匹配的文档

        Args:

            key: 查询关键字

            question: 向量化的问题

            pid: 真实pid

        Returns:

            查找到的最匹配文档的pid

            若输入提供pid，则额外返回准确率

        """

    def delete(self, key: str):

        """刪除key對應的屬性

        Args:

            key: 刪除屬性

        """

    def update(self, learner, key: List[str], value: str):

        """将key作为参数调用learner，将返回的结果保存到value中

        Args:

            learner: 学习器，必须声明predict

            key: 调用参数键

            value: 保存键

        """

分词器位于utils/tokenizer.py文件中，其用于对给定的句子进行分词，所用为LTP包。

分类器定义如下：

class Tokenizer(object):

    """分词器

    采用哈工大LTP语言技术平台

    Attributes:

        \_\_tokenizer: 分词器

        \_\_vocabulary: 单词表

        \_\_stopwords: 停用词

    """

其各类方法如下所示：

    def predict(self, paragraph: List[str]) -> List[str]:

        """对给定的文段进行分词处理，用空格区分

        Args:

            paragraph: 待分词文段

            limit: 删除频率小于limit的低频词

        Returns:

            分词结果，用空格分隔

        """

向量化器位于utils/vectorizer.py文件中，其用于将分词的句子转换为一个实值向量，采用tf-idf表示法，公式为：

由于向量化的结果规模过于庞大，无法直接保存到本地文件中，所以为此训练了一个向量化器，采用sklearn.feature\_extraction.text.TfidfVectorizer，其可以直接将分词句子转化为tf-idf向量。

向量化器定义如下：

class Vectorizer(object):

     """向量化器

     将一个已分词单词序列对应到一个实值向量

     采用tf-idf表示法

     Attributes:

         \_\_vectorizer: 向量化器

     """

综上所述，该步使用LTP进行分词、去停用词，然后将其映射为实值向量，便于后续机器学习模型进行训练和预测。

1. 问题分类

该任务程序入口为question\_classification.py。

分类器位于utils/classifier.py文件中，其将句子进行分类，由于训练样本较少，所以仅预测每个句子的大类。

分类器分别尝试KNN（N=5）和SVM（每个分类训练一个分类器），其预测准确率分别为35.34%和64.66%。

故后续采用SVM的结果。

分类器定义如下：

class Classifier(object):

    """分类器

    输入代表句子的实值向量，输出句子的分类

    其中分类可见《哈工大IR研究室问题分类体系ver2.doc》

    仅预测大类

    Attributes:

        \_\_kind2id: 分类映射为ID

        \_\_id2kind: ID映射为分类

        \_\_kind\_cnt: 分类的种类数

        \_\_classifier: 分类器

    """

1. 候选答案句排序

该任务的程序入口为answer\_sentence\_selection.py。

首先根据各个句子的向量表示的余弦相似度找到最合适的文档。

由于单个文档中各个句子相似度较高，所以不能直接求各个句子和问题的余弦相似度来获得候选答案句。

考虑到两个句子的编辑距离、最长公共序列可以很好地反映两个句子的相似程度，所以对于每个子句，会生成一个两维的向量，其中第一维为该句和问题的编辑距离，第二维为该句和问题的编辑距离。

然后采用逻辑回归来进行二分类，来求出每个句子为答案句的概率。（不采用其他分类模型是因为不能求出概率）

选择器位于utils/selector.py文件中，其正确率为

1. 答案抽取

该任务的程序入口为answer\_span\_selection.py。

由于在任务2中，我们已经对每个句子做了分类，因此该任务中会对数据依照大类做命名实体识别任务，采用BIO标注法。

具体来说，对于训练集中的一条问题-文档对，设该问题的文档为“赵显宰 1980年 出生 于 韩国 ， 韩国 影视 男演员 。”，问题的分类为NUM，则会给该问题-文档对进行实体标注“O B-NUM O O O O O O O O”，并利用该数据训练识别器。

在进行答案抽取时，我们在获取了问题的类别后，会对文档进行命名实体识别，并抽取出类别对应的单词。

识别器位于utils/recognizer.py文件中，采用CRF算法，特征为：每个单词的前后三个单词、该单词是否为大/小写、该单词是否为数字。

由于CRF模型并不完善，若标注任务并没有标注出与问题类型相同的单词，则答案为空。

## 四、实验心得

* 机器学习相关任务总重要的就是知识的获取，所以如何充分地利用知识（问题的分类），来改善我们最终任务的性能（答案抽取），是需要着重考虑的问题。在本次实验中，尝试了将信息迁移到原来的问题中，取得了不错的效果。
* 数据规模较大，导致机器学习模型的学习和预测效率较低，所以如何改善模型的性能也是需要考虑的问题。在本次实验中，尝试了降维的方法（参见utils/reducer.py），但是由于维数过高（和单词总数同维），导致降维的性能和效果并不好，故未在最终的结果中使用。