第二届“中电慧治”杯政府治理大数据应用

队伍名称：菜鸟飞航

队伍编号：　0　4　8

队长：　　马 秋 平

队员：马秋平、张拓实

1. 问题描述

1、 问题背景

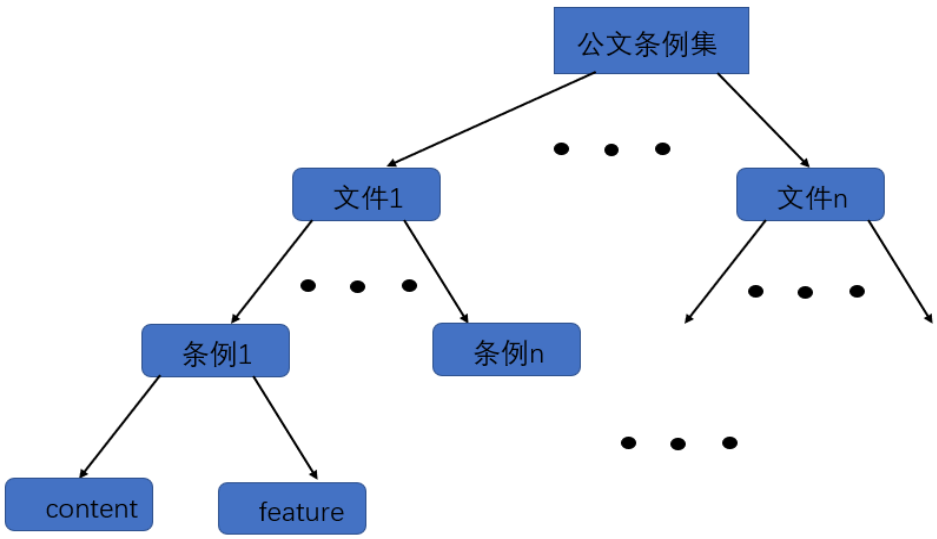
当前，“互联网+”等信息技术与政府治理融合日趋深入，智慧治理作为一种新型的政府治理模式正在推动新一轮的政府转型。在社会由信息化、数字化向智能化、智慧化迈进的环境下，积极推进“大数据+政务服务”，是贯彻落实党中央、国务院决策部署，把简政放权、放管结合、优化服务改革推向纵深的关键环节，对加快转变政府职能，提高政府服务效率和透明度，便利群众办事创业，进一步激发市场活力和社会创造力具有重要意义。 目前，政府各委办局的官方网站上都提供了政策公文检索功能，方便公众查询需要的政策信息。但由于公众普遍无法提出准确的政策关键字，而是直接抛出具体问题，如“请问生育津贴领取的条件是如何？依据的法律规定是什么？”，导致公文检索效果不佳。因此，基于语义搜索算法帮助公众准确检索政策公文与具体条例是十分有价值的。

2、目标问题

需要根据输入的问题，返回对应的政策内容，以及相应的公文名称和具体条例。

2. 模型建立与求解

1. 公文条例的存储结构



由于所给文件中还有很多条公文，且每条公文里面又含有多个条例，例如公文《成都市企业职工基本养老保险社会统筹与个人账户相结合实施办法》中就含有：第六条-2、第六条-13、第六条-1、第六条-4。公文与条例集之间是具有一对多的映射关系，所以采用字典来存储公文及其条例，其中公文名称为作为 Key，条例作为 Value。对于公文下面每一个条例来说，它又包含含有两部分内容：具体内容（content）和特征(feature)。对于条例集，我们同样也用字典来存储。这样整个公文条例集我们就用一个二级字典来存储。

2.分词

我们用常用的汉语分词库 **jieba** 对句子进行分词。

3.问题语句信息提取

中国自古就是礼仪之邦，所以人们在提问时会十分讲究礼仪，其结果就是人们在提问时会加一些没有信息量的词汇进去，比如“您好”、“请问”、“谢谢”等，这些词语的出现是的对句子的信息提取变得困难起来，所以要先去除这些无关词汇，也就是停用词。

当对问题先进行分词，再去除停用词之后，其结果就是包含有具体信息的词向量。

4.特征提取

再测试集中提取出所有答案中包含条例 item_1 的所有问题 $\{q_1, q_2, \cdots, q_n\}$，而对于其中每一个问题，其所对应的词向量为：$q_1 = \{w1_1, w_{12}, \cdots, w_{1m_1}\}, q_2 = \{w2_1, w2_2, \cdots, w2_{m_2}\}, \ldots, \ q_n = \{wn_1, wn_2, \cdots, wn_{m_2}\}$,

统计每一个分词在该问题下的出现频次：

P(Word_1) = count(q1,word Word_1)+ count(q2,word Word_1)+…+ count(qn,word Word_1)

P(Word_2) = count(q1,word Word_2)+ count(q2,word Word_2)+…+ count(qn,word Word_2)

……..

P(Word_k) = count(q1,word Word_k)+ count(q2,word Word_k)+…+ count(qn,word Word_k)

归一化处理：

Pos(word_1) = $\frac{\mathrm{P(Word\_1)}}{\mathrm{P(Word_1)+P(Word_2)+\cdots+P(Word\_n)}}$,

我们把所有的词汇及对应的频次作为该 item 的 feature,

即：

feature(itemi)

$= \{word_1 = Pos(word_1), word_2 = Pos(word_2), \ldots, word_n = Pos(word_n)\}$

根据香浓信息论，word_i 出现对 item_i 贡献的信息量为：

H(word_i) = $-\log \ \ P(word_i)$

那么当多个词语同时出现时，时所贡献的信息量为：

H(word_1, word_2,…, word_k,) = $-\log \ \ P(word_1) - \log \ \ P(word_2) - \cdots - \log \ \ P(word_2)$

5.匹配过程

当读取到用户输入问题 Q 时，对其进行分词和去除停用词之后，得到其词向量

$$\mathrm{Q} = \{qw_1, qw_2, \ldots, qw\_j\}$$

然后对于每一个 item，我们用如下公式来计算其与该问题的相关程度:

$$\mathrm{R(Q, item_i)} = pos(qw_1) + pos(qw_2) + \cdots + pos(qw_k)$$

然后设定一个阈值，输出满足条件的 item 即可。

3、代码

```
import openpyxl  # 导入用来操作 xlsx 的库
import jieba  # 导入用来分词的库
from sys import argv
'''
   获取用户问题及相关的文件
'''
# row_index 用来表示行的索引
row_index = ['E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
             'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
stopwords = ['你好', '请问', '，  ', '的', '-', ' 需要', '您好', '了解', '应该', '准备', '是不是', '不能',
             '没有', '>', '找到', '成都市', ' 可否', '？ ', '我', '没有', '一直', '都在', '吗', '我能', '都', '，  ',
             '尊敬', '本人', '_', '在', '年', '是', ' ', '将', '说', '用', '\n', '今年', '月', '需要', '日',
             '重庆', '2018', '给', '某', '1', '2', '3', '4', '5', '6', '7', '8', '没', '以后', '了',
             '刷', '深圳', '成华区', '哪些', '！  ', '能', '。']
# 导入训练集文件
wb1 = openpyxl.load_workbook(r"data\训练集.xlsx")
# 获取活跃的 sheet
ws1 = wb1.active
# 导入公文条例集
wb = openpyxl.load_workbook(r"data\公文条例集.xlsx")
# 获取活跃的 sheet

ws = wb.active

doc_dict = {}  # 定义一个字典用来保存文件

'''
   先创建一个字典来存储政策条例，键为公文名，值为条例名称
'''
for i in range(2, ws.max_row + 1):
    doc_dict[ws['B' + str(i)].value] = {}

for i in range(2, ws.max_row + 1):
    doc_dict[ws['B' + str(i)].value][ws['C' + str(i)].value] = {}

for i in range(2, ws.max_row + 1):
    doc_dict[ws['B' + str(i)].value][ws['C' + str(i)].value]['content'] = ws['D' + str(i)].value
    doc_dict[ws['B' + str(i)].value][ws['C' + str(i)].value]['feature'] = \
        [list(doc_dict.keys()).index(ws['B' + str(i)].value),
         list(doc_dict[ws['B' + str(i)].value].keys()).index(ws['C' + str(i)].value), {}]


def count_num(jieba_result, count_dict):
    '''
    输入为对问题分词后的列表，和用来记录每个词语对应该条例所出现的次数
    :param jieba_result:
    :param count_dict:
    :return:
    '''
    for item in jieba_result:
        count_dict[item] = count_dict.get(item, 0) + 1
    return count_dict
```

```
def get_distance(array=[], word1='', word2=''):
    '''
    获取输入字符串中的任意两词语之间的距离
    :param array:
    :param word1:
    :param word2:
    :return:
    '''
    if word1 and word2 in array:
        distance_1 = array.index(word1)
        distance_2 = array.index(word2)
        distance = np.abs(distance_1-distance_2)
        return distance
    else:
        return None


def find_doc(a):
    '''
    根据所索引可以查找对应的文件，参数为列表
    :param a:
    :return:
    '''
    dic1 = doc_dict[list(doc_dict.keys())[a[0]]]
    list1 = list(dic1.keys())
    dict2 = dic1[list1[0]]
    result = dict2['content']
    return result


def update_feature(feature_list, update_dict):
    '''
    特征提取之后更新公文条例的 feature
    :param feature_list:
    :param update_dict:
    :return:
    '''
    result1 = doc_dict.get(list(doc_dict.keys())[feature_list[0]])
    result2 = result1.get(list(result1.keys())[feature_list[1]])
    result2['feature'][2].update(update_dict)
    # print(result2['feature'])


def function(a, b):
    '''
    根据文件名和条例名查询内容
    :param a:
    :param b:
    :return:
    '''
    a, b = str(a), str(b)
    a = a.lstrip()
    a = a.rstrip()
    result1 = doc_dict.get(a)
    if result1 is None:
        a = remove_kuohao(a)
        result1 = doc_dict.get(a)
    if type(result1) is dict:
```

```
        if type(b) is int:
            result2 = result1.get(b)
        else:
            result2 = result1.get(b)
            if result2 is None:
                result2 = result1.get(b.strip(' '))
                if result2 is None:
                    for items in result1:
                        content = result1.get(items)['content']
                        if b in content:
                            result2 = result1.get(items)
        return result2['feature']
    else:
        return None


def remove_kuohao(input_str=''):
    '''
    因为有些条例加了书名号，有些没加，所以该函数是为了去除书名号，便于查询
    :param input_str:
    :return:inputz_str
    '''
    if input_str.startswith('《'):
        return input_str[1:len(input_str)-1]


for i in range(2, ws1.max_row):
    question = ws1["B"+str(i)].value
    if question is None:
        break
    words = jieba.lcut(question)
    for j in range(0, len(row_index), 2):
        if ws1[row_index[j]+str(i)].value is not None:
            doc = ws1[row_index[j]+str(i)].value       # 获取文件名
            item = ws1[row_index[j+1]+str(i)].value    # 获取条例名称
            try:
                result = function(doc, item)
                pre_dict = result[2]
                for k in words:
                    pre_dict[k] = pre_dict.get(k, 0)+1
                update_feature(result, pre_dict)
                pre_dict = {}
            except TypeError:
                pass
        else:
            break
# def search(input_str):
#     '''
#     根据问题查询对应的公文条例
#     :param input_str:
#     :return:
#     '''
#     seg_result = jieba.lcut(input_str)
#     print(seg_result)
#     answer_ = '|'
#     answer_file = '您好！'
#     max_len = 0.8
#     max_pos = 0
#     pre_result = []
```

```
#     seg_result = set(seg_result)
#     for sin in list(seg_result):
#         if sin in stopwords:
#             try:
#                 seg_result.remove(sin)
#             except KeyError:
#                 pass
#     # print(seg_result)
#     for feature in feature_matrix:
#         reference = 0
#         common_part = []
#         for word in seg_result:
#             reference = feature[2].get(word, 0) + reference
#             if word in feature[2].keys():
#                 common_part.append(word)
#
#         if reference > 0.1:
#             # if len(common_part)/(len(feature)-2) > max_len:
#             if reference > max_pos:
#                 max_pos = reference
#                 pre_result.append([feature, reference])
#
#     for feature_ in pre_result:
#         if feature_[1] > 0.8*max_pos:
#             # if feature_[2] > 0.7:
#             result_ = search_title(feature_[0])
#             answer_ = answer_ + result_[0]+'|'+result_[1]+'|'
#             regular = "根据"+'《'+result_[0]+'》'+"的相关规定"
#             answer_file = answer_file+regular+doc_dict[result_[0]][result_[1]]['content']
#     return answer_file+answer_
#


def search(input_str):
    '''
    根据问题查询对应的公文条例
    :param input_str:
    :return:
    '''
    seg_result = jieba.lcut(input_str)
    answer_ = '|'
    answer_file = '您好！'
    max_pos = 0
    pre_result = []
    seg_result = set(seg_result)
    for sin in list(seg_result):
        if sin in stopwords:
            try:
                seg_result.remove(sin)
            except KeyError:
                pass
    for feature in feature_matrix:
        reference = 0
        common_part = []
        for word in seg_result:
            reference = feature[2].get(word, 0) + reference
            if word in feature[2].keys():
                common_part.append(word)
```

```
        if reference > 0.1:
            if reference > max_pos:
                max_pos = reference
            pre_result.append([feature, reference])
    for feature_ in pre_result:
        if feature_[1] > 0.9*max_pos:
            result_ = search_title(feature_[0])
            answer_ = answer_ + result_[0]+'|'+result_[1]+'|'
            regular = "根据"+'《'+result_[0]+'》'+"的相关规定"
            answer_file = answer_file+regular+doc_dict[result_[0]][result_[1]]['content']
    return answer_file+answer_


def search_title(a):
    '''
    根据每个条例下的特征查找其所对应的公文名称及条例名称
    :param a:
    :return:
    '''
    result1 = list(doc_dict.keys())[a[0]]
    result2 = list(doc_dict.get(result1).keys())[a[1]]
    return result1, result2


if __name__ == "__main__":
    feature_matrix = []
    matrix_row = 0
    for m in doc_dict:
        one = doc_dict.get(m)
        for n in one:
            two = one.get(n)['feature'][2]
            for o in list(two.keys()):
                if o in stopwords:
                    try:
                        del two[o]
                    except KeyError:
                        pass
            add_all = sum(two.values())
            for item in two.keys():
                two[item] = two.get(item) / add_all
            # print(two)
            doc_dict[m][n]['feature'][2] = two
            feature_matrix.append(doc_dict[m][n]['feature'])
            matrix_row = matrix_row + 1
    wb1.close()
    wb.close()
    test = argv[1]
    with open('result.txt', 'w', encoding='utf-8') as res:
        with open(test, 'r') as que:
            questions = que.readlines()
            for question in questions:
                if question is not None:
                    # print(question)
                    answer = search(question)
                    # print(answer)
                    res.writelines(answer)
                    res.write('\n\n')
                else:
                    break
```

4、参考文献

[1] 爱丽丝·郑 ， 阿曼达·卡萨丽 《精通特征工程》; 人民邮电出版社 2019.
[2] 黄锦辉 等, 《中文自然语言处理导论》: 科学出版社; 2018,
[3] 黄凯 仇 晶 等, 《信息检索与只能信息处理》: 国防工业出版社, 2014
[4] 张仰森, 《统计语言建模与中文文本自动校对技术》: 科学出版社, 2017.
[5] 涂铭 刘祥 刘树春, 《Python 自然语言处理实战-核心技术与算法》 机械工业出版社 2018