冬奥会领域问答机器人实验报告

吴俊亮 贾志杰 孙严顺 陈天宇

一、实验任务

本实验要求实现一个冬奥会领域的问答机器人：当输入一个有关冬奥会的问题时，机器人会给出相应的答案。按照实验要求，输入和输出的文件均为 json 格式；除去初始加载模型的时间，单次响应时间应在 500ms 以内。我们将提供的标注数据 1.xlsx 和标注数据 2.xlsx 转换成了 json 格式的训练集，读者可以参考 train_set.json。将提供的测试集.xlsx 转换成了 json 格式的测试集，读者可以参考 test_set.json。

二、实验设计

（一）实验原理

我们的总体设计思路是使用 python 的 gensim 库，采用 TF-IDF 模型+余弦相似度的方法构建一个检索式问答系统。通过判断测试问题与训练集中问题的相似度，选择相似度最高的训练集中问题的答案作为相应测试问题的答案。

1、 gensim 库

gensim 用于从原始的非结构化的文本中，无监督地学习到文本隐层的向量表达。gensim 有三大宏观概念：语料→向量→模型。

语料：语料是指一组语句的集合，这个集合是 gensim 的输入。在本次实验中语料即为训练集的全部问题。收集语料之后，需要做一些预处理，对于英文就是根据空格将原来的句子划分成单词。我们这里的预处理是使用 jieba 库把中文语句切分成词组。

向量：如果要对语句的隐含结构进行推断，就需要使用适当的数学模型：我们在这里使用的方法是 doc2bow，也就是将语句转化为 bag-of-words（词袋）。在词袋向量中，每个句子被表示成一个向量，代表字典中每个词出现的次数。例如，给定一个包含[‘2020’,‘冬奥会’,‘举办地’, ‘金牌’]的字典，语句[‘2020’, ‘冬奥会’, ‘2020’]字可以表示成向量[(0,2),(1,1)]，表示‘2020’（编号 0）出现了 2 次、 ‘冬奥会’（编号 1）出现了 1 次。词袋模型的一个重要特点是，它完全忽略了单词在句子中出现的顺序，这也就是“词袋”这个名字的由来。

模型：现在我们已经向量化了语料，接下来可以使用各种模型了。我们可以认为模型转换是在两个向量空间进行转换。以本次实验为例，我们通过 TF-IDF 模型把词袋向量转换到另一个向量空间再进行比较，关于 TF-IDF 模型将在下面详细介绍。

2、 TF-IDF 模型

TF-IDF 是一种针对关键词的统计分析方法，用于评估一个词对一个语料库的重要程度。一个词的重要程度跟它在测试语句（一个问题）中出现的次数成正比，跟它在语料库（训练集中的全部问题）出现的次数成反比。这种计算方式能有效避免常用词对关键词的影响，提高了关键词与测试语句之间的相关性。

TF（Term Frequency）指的是某词在测试语句中出现的总次数：

$$TF = \text{某词在测试语句中出现的次数/测试语句的总词量}$$

IDF（Inverse Document Frequency）指的是逆向文档频率。语料库包含某词语的训练语句越少，IDF 值越大，说明该词语具有很强的区分能力：

$$IDF = ln(\text{语料库中语句总数/包含该词的语句数} + 1)$$

知道了“词频”（TF）和“逆文档频率”（IDF）以后，将这两个值相乘，就得到了一个词的 TF-IDF 值。某个词对测试语句的重要性越高，它的 TF-IDF 值就越大。

对于给定的语料库，我们可以计算出每一个词的 IDF；给定一个测试语句，我们可以计算出该测试语句中每个词的 TF。形象地来说，IDF 是“全局变量”，表示一个词的区分能力；TF 是“局部变量”，表示一个词在这个问题中的重要程度。举一个例子：问题“1924 年冬奥会的举办地是什么？”中“冬奥会”和“？”的频率是一样的，即它们的 TF 相同；但是“冬奥会”的重要程度显然高于“？”，即它们的 IDF 不同。

3、 余弦相似度

对于训练集中的每一个问题，我们可以得到一个 TF-IDF 向量，表示为

$$[(\text{词语 1 的 id，词语 1 的 tfidf 值}), (\text{词语 2 的 id，词语 2 的 tfidf 值}), ...]$$

用户输入的一个问题也可以得到相应的 TF-IDF 向量。我们通过比较向量之间的余弦相似度来进行训练语句和测试语句的相似度的比较。余弦相似度是通过计算两个向量的夹角余弦值来评估他们的相似度。余弦值的范围在[-1,1]之间，值越趋近于 1，代表两个向量的方向

越接近，也就是相似度越高。

（二）重要模块分析

1、生成分词结果

我们通过 json.load 方法从 json 格式的训练集中读取全部问答对，存在变量 data 中。

```
with open(train_filepath, encoding='utf-8') as train_file:
    data = json.load(train_file)
```

如果分词结果已经存在，则读取分词结果；否则进行分词并保存分词结果：对于 data 中的每一个问答对，将其中的问题通过 split_word 函数调用 jieba 库进行分词，加入到列表 content 中。

```
content = []
if os.path.exists(splitdata_filepath):
    with open(splitdata_filepath, encoding='utf-8') as f:
        content = json.load(f)
else:
    for value in data:
        question = value['question']
        content.append(split_word(question))
    with open(splitdata_filepath, 'w', encoding='utf-8') as f:
        f.write(json.dumps(content, ensure_ascii=False))
```

2、生成 gensim 字典

如果 gensim 字典已存在，则读取 gensim 字典；否则生成 gensim 字典。上一步中训练集的所有问题已经转换成了分词后的列表 content，现在对这个列表使用 gensim 中的 corpora.Dictionary 方法生成字典并且保存在文件 dictionary 中。字典存储的是词语和 id 之间的映射。此处使用 num_features 变量保存字典长度，并将训练集中的每一个问题转换为一个词袋。所有的问题对应的词袋形成了语料库（corpus）。

```
if os.path.exists(dictionary_filepath):
    dictionary = gensim.corpora.Dictionary.load(dictionary_filepath)
else:
    dictionary = gensim.corpora.Dictionary(content)
    dictionary.save(dictionary_filepath)
num_features = len(dictionary)
corpus = [dictionary.doc2bow(line) for line in content]
```

3、生成 TF-IDF 模型

如果 TF-IDF 模型已存在，则读取 TF-IDF 模型；否则生成 TF-IDF 模型。根据语料库，我们可以计算出 TF-IDF 模型的总词语数、总语句数以及每一个词语对应的 IDF 值。

```
if os.path.exists(model_filepath):
    tfidf = gensim.models.TfidfModel.load(model_filepath)
else:
    tfidf = gensim.models.TfidfModel(corpus)
    tfidf.save(model_filepath)
```

4、 生成 TF-IDF 相似度比较序列

如果 TF-IDF 相似度比较序列已存在，则读取 TF-IDF 相似度比较序列；否则生成 TF-IDF 相似度比较序列。此步通过 gensim 中的 similarities.Similarity 方法生成相似度比较序列，即对训练集中的每一个问题计算出一个 TF-IDF 向量。

```
if os.path.exists(index_filepath):
    index = gensim.similarities.Similarity.load(index_filepath)
else:
    index = gensim.similarities.Similarity(index_filepath, tfidf[corpus
], num_features)
    index.save(index_filepath)
```

5、 检索过程

用户提出的问题存储在变量 sentences 中。首先对用户输入问题进行分词、转词袋表示。相似度比较的过程将词袋 vec 传入 TF-IDF 模型中，得到一个 TF-IDF 向量，该向量与训练集中每个问题的对应向量计算余弦相似度，形成相似度比较序列。之后将相似度比较序列排序，排序后列表的第一位即为训练集中与输入问题相似度最高的问题，系统输出该问题对应答案。

```
sentences = split_word(sentences)     # 分词
vec = dictionary.doc2bow(sentences)   # 转词袋表示
sims = index[tfidf[vec]]              # 相似度比较
sorted_sims = sorted(enumerate(sims), key=lambda x: x[1], reverse=True)
i = sorted_sims[0][0]                 # 最相似的问题的序号
print(data[i]['answer'])
```

三、实验结果

实验结果请参考 output.json。因未提供参考答案，所以无法计算可接受率。

四、组内分工

成员	分工
吴俊亮	完成代码，完善实验报告和实验展示 PPT
贾志杰	查找资料，处理数据，完成部分代码
孙严顺	根据实验报告完成实验展示 PPT
陈天宇	进行环境配置，搜索相关资料，初步完成实验报告

五、参考文献

1. 【gensim 中文教程】开始使用 gensim, https://blog.csdn.net/duinodu/article/details/76618638, 2017. [Online; accessed 2021-01-04].

2. TF-IDF 算法原理及其使用详解，https://zhuanlan.zhihu.com/p/94446764，2019. [Online; accessed 2021-01-04].

3. 检索式问答机器人，https://github.com/vba34520/Retrieval-Bot，2020. [Online; accessed 2021-01-04].