远程项目指导学习报告----人工智能方向

智能聊天机器人

2017/02/21

田矿佳

目录

1. 学习背景

1.1 学习背景

当前的人工智能产业的发展浪潮，主要是源于深度学习算法的提出，在数据量和计算能力的基础上实现大规模计算，属于技术性突破。属于超级人工智能的，关于意识起源、人脑机理等方面的基础理论研究仍有继续突破的余地。

目前，苹果、谷歌、微软、亚马逊和 Facebook 这五大巨头无一例外都投入了越来越多的资源，来抢占人工智能市场，甚至将自己整体转型为人工智能驱动型的公司。国内互联网领军者“BAT”也将人工智能作为重点战略，凭借自身优势，积极布局人工智能领域。

现今中国人工智能行业的创业公司发展领域各色各异，计算机视觉领域拥有最多创业公司，其次就是服务机器人领域，而排名第三的是语音及自然语言处理领域，智能医疗、机器学习、智能驾驶等也是相比比较热门的领域之一。

在当下的科技背景中，许多人工智能是需要通过语言交流为基础，或者说通过语言驱动的产品，来更加方便、快捷的为人们服务，现在许多的智能产品，已经离不开语言了。

在本次的学习中，主要是学习以 NLU 为基础，通过 RASA NLU 为聊天机器数据训练和处理框架，以 Spacy 为后台自然语言处理框架，以及正则表达式、词向量、意图识别、实体抽取、判定角色、多轮单次问询、多轮多次问询、否定问询、状态变迁、询问队列等核心技术。

同时，在本次学习中是以 python 为基础，最后同 wxpy 相结合，

实现机器人在微信上的使用，增添了本次项目成果的实用性。

1.2 Anaconda 和 Jupyter Notebook

在这次学习中，我使用的是 Anaconda 和 Jupyter Notebook 来进行代码的实现,anaconda 是一个 Python 的开源版本,其包含了 conda、Python 等 180 多个科学包及其依赖项。并适用于 Linux、Windows、Mac 等多个操作系统，其中包含的 Jupyter Notebook 是一款十分方便使用于新手使用的 IDE，方便测试代码，但是不能检查出一次错误的地方。同时，conda 指令方便下载各种库和软件（如在本次学习中用到的 en_core_web_md、spacy 等）。

2. 正则表达式与意图识别

2.1 正则表达式

正则表达式作为搜索和获取字符串的一种工具，有十分强大、便捷的功能，它不仅能匹配所需的字符串，获取在字符串中我们想要的特定部分，也十分的灵活。对于初学者来说，正则表达式可能会比较晦涩难懂，但一旦熟悉，是十分方便的。

正则表达式	描述
/\b([a-z]+) \1\b/gi	一个单词连续出现的位置。
/(\w+):\/\/([^/:]+)(:\d*)?([^#]*)/	将一个URL解析为协议、域、端口及相对路径。
/^(?:Chapter|Section) [1-9][0-9]{0,1}$/	定位章节的位置。
/[-a-z]/	a至z共26个字母再加一个-号。
/ter\b/	可匹配chapter，而不能匹配terminal。
/\Bapt/	可匹配chapter，而不能匹配aptitude。
/Windows(?=95 |98 |NT)/	可匹配Windows95或Windows98或WindowsNT，当找到一个匹配后，从Windows后面开始进行下一次的检索匹配。
/^\s*$/	匹配空行。
/\d{2}-\d{5}/	验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。
/<\s*(\S+)(\s[^>]*)?>[\s\S]*<\s*\/\1\s*>/	匹配 HTML 标记。

此外，正则表达式还有不同的模式，运用不同模式的组合可以得到最

终自己所需要的特定字符串。

模式	描述
^	匹配字符串的开头
$	匹配字符串的末尾。
.	匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符。
[...]	用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'
[^...]	不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。
re*	匹配0个或多个的表达式。
re+	匹配1个或多个的表达式。
re?	匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式
re{ n}	精确匹配 n 个前面表达式。例如， `o{2}` 不能匹配 "Bob" 中的 "o"，但是能匹配 "food" 中的两个 o。
re{ n,}	匹配 n 个前面表达式。例如， o{2,} 不能匹配"Bob"中的"o"，但能匹配 "foooood"中的所有 o。"o{1,}" 等价于 "o+"。"o{0,}" 则等价于 "o*"。

2.2 NLP

NLP 是神经语言程序学（Neuro-Linguistic Programming）的英文缩写。

自然语言处理（NLP）是计算机科学、人工智能和语言学的交叉领域。目的是让计算机处理或“理解”自然语言，以执行诸如语言翻译和问题回答等任务。

随着语音接口和聊天机器人的兴起，NLP 成为了信息时代最重要的技术之一，是人工智能的重要组成部分。充分理解和表达语言的含义是一个极其困难的目标。为什么?因为人类的语言很特别。

人类的语言十分的复杂，看似平常能够十分容易的进行理解，但是要让计算机准确的判断人类语言所表达的意思，目前来说是十分困难的。不同的语境，一词多义，否定，语气，断句等等都会造成一句同样的话但是表达了完全不同的意思。因此，NLP 的出现，帮助人们

较为完备的解决了这个问题，虽然不是十分的准确，但总体上来说已经足够应对大部分的语义的处理

NLP 通过文本嵌入（Text Embeddings)和机器翻译两种方式对自然语言进行关键词搜索，提取信息，分类等操作，让自然语言变得让机器能够更加容易进行处理，是目前十分重要的自然语言处理方法。

其中 LSTM(long-short Term Memory)网络试图通过引入门（gate）和明确定义的记忆单元(memory cell)来解决梯度消失/爆炸问题。每个神经元都有一个 memory cell 和三个 gate： input，output 和 forget。这些门的功能是通过停止或允许信息流来保护信息。

LSTMs 已经被证明能够学习复杂的序列.请注意，这些门中的每一个都对前一个神经元中的一个单元具有权重，因此它们通常需要更多资源才能运行。LSTM 目前非常流行，并且在机器翻译中被广泛使用。除此之外，它是大多数序列标签任务的默认模型，其中有大量的数据。

2.3 词向量与 spacy

词向量是嵌入式自然语言处理（NLP）中的一组语言建模和特征学习技术的统称，其中来自词汇表的单词或短语被映射到实数的向量。从概念上讲，它涉及从每个单词一维的空间到具有更低维度的连续向量空间的数学嵌入。

词向量在刚开始接触机器学习，深度学习时，是十分重要的工具。在本次学习当中，将对话中的语句中的每一个单词当作词向量进行处理，在向量上进行数学建模，将自然语言（或语句）处理成为适合计算器处理的形式。

Spacy 是一个 Python 自然语言处理工具包，其中大量使用了 Cython 来提高相关模块的性能，所以它的运行效率十分高。spaCy 提供了一系列简洁的 API 方便用户使用，并基于已经训练好的机器学习与深度学习模型实现底层。

命名实体识别（NER）：

```
In [11]: test_doc = nlp(u"Rami Eid is studying at Stony Brook University in New York")

In [12]: for ent in test_doc.ents:
print(ent, ent.label_, ent.label)
....:
(Rami Eid, u'PERSON', 346)
(Stony Brook University, u'ORG', 349)
(New York, u'GPE', 350)
```

在最终项目的实现中，采用了 spcay 进行自然语言处理，并引入了‘en_core_web_md’包，其采用了 300 维的词向量来进行分析（向量维度越高，可信度的越高），可以通过查看来观测每一个词向量得到的数值。

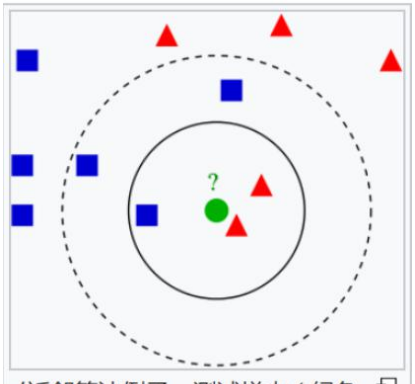
```
# Load the spacy model: nlp, en_core_web_md
nlp = spacy.load("en_core_web_md")

# Calculate the length of sentences
n_sentences = len(sentences)

# Calculate the dimensionality of nlp
embedding_dim = nlp.vocab.vectors_length

# Initialize the array with zeros: X
X = np.zeros((n_sentences, embedding_dim))

# Iterate over the sentences
for idx, sentence in enumerate(sentences):
    # Pass each each sentence to the nlp object to create a document
    doc = nlp(sentences)
    # Save the document's .vector attribute to the corresponding row in X
    X[idx, :] = doc.vector
```

2.4 意图识别与最邻近法则

KNN(k-nearest neighbor 的缩写)最近邻算法，通俗来说，它将每一个句子进行了标签，分类不同的类别，用 k 作为距离度量，寻找与本句最相似的标签示例，将该标签的意图作为自己的意图来进行使用。

最近邻法则作为意图识别中十分重要的部分，在聊天机器人中占了很大的作用，本次学习中便是使用的 scikit-learn 中的最近邻分类，来提高意图识别的准确度，在上述讲到的正则表达式同样也可以用来进行意图识别，但是正则表达式的局限性十分大，只能够满足一些比较简单的要求，对于一些复杂的，还有进行分类这一部分，用最近邻分类法要比正则表达式方便且准确的。

下图是最近邻分类法的原理示意图以及在本次学习中用到的 scikit-learn 中最近邻分类的伪代码示意图。



*k*近邻算法例子。测试样本（绿色圆形）应归入要么是第一类的蓝色方形或是第二类的红色三角形。如果k=3（实线圆圈）它被分配给第二类，因为有2个三角形和只有1个正方形在内侧圆圈之内。如果k=5（虚线圆圈）它被分配到第一类（3个正方形与2个三角形在外侧圆圈之内）。

scikit-learn中的最近邻分类

```
In [1]: from sklearn.metrics.pairwise import cosine_similarity

In [2]: test_message = """
i would like to find a flight from charlotte
to las vegas that makes a stop in st. louis"""

In [3]: test_x = nlp(test_message).vector
In [4]: scores = [
   ...:     cosine_similarity(X[i,:], test_x)
   ...:     for i in range(len(sentences_train)
   ...: ]
In [5]: labels_train[np.argmax(scores)]

Out[5]: 'atis_flight'
```

2.5 支持向量机（SVM）

SVM 即 Support Vector Machine,主要是用于解决数据分类的问题,在本次学习中主要是运用其来实现最近邻法,并不是主要的方法。

```
# Import SVC
from sklearn.svm import SVC

# Create a support vector classifier
clf = SVC()

# Fit the classifier using the training data
clf.fit(x_train,y_train)

# Predict the labels of the test set
y_pred = clf.predict(x_test)
```

3. Rasa_Nlu 和多轮查询

3.1 Rasa_Nlu

Rasa 是开源工具,包含了 Rasa_Core 和 Rasa_Nlu,其中 Rasa_Nlu 主要是用于用户的意图识别和实体识别，如下图所示。

```
{
  "intent": "search_restaurant",
  "entities": {
    "cuisine" : "Mexican",
    "location" : "center"
  }
}
```

而 Rasa_Core 主要是用于对话流程的配置和训练。

使用 Rasa_Nlu 的具体流程大概为：

1. 预料的获取和预处理

2. MITIE 模型训练

3. 构建 rasa_nlu 语料和模型

4. 通过 rasa_nlu 进行训练

3.2 SQL

Structured Query Language(SQL),即结构化查询语言，它允许用户在高层数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式,所以具有完全不同底层结构的不同数据库系统，可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使它具有极大的灵活性和强大的功能。

3.3 否定实体

在人类的语言当中，否定是一种比较特殊但却十分常用的手法，但是，正是因为有否定的存在，使得计算机对于人类语言的理解变得更加的复杂，否定并不是意味着对本句话或者说对上文的否决，或者说是意义相反，否定有时候会表达与现在相反的意义，但也会表达强调的意义，但是也有一些特殊的，双重否定等，所以，对于否定的识别变得十分的重要，识别出否定实体会方便我们对于意图的判断，从而进行相应的操作。

```
# Define negated_ents()
def negated_ents(phrase):
    # Extract the entities using keyword matching
    ents = [e for e in ["south", "north"] if e in phrase]
    # Find the index of the final character of each entity
    ends = sorted([phrase.index(e) + len(e) for e in ents])
    # Initialise a list to store sentence chunks
    chunks = []
    # Take slices of the sentence up to and including each entitiy
    start = 0
    for end in ends:
        chunks.append(phrase[start:end])
        start = end
    result = {}
    # Iterate over the chunks and look for entities
    for chunk in chunks:
        for ent in ents:
            if ent in chunk:
                # If the entity is preceeded by a negation, give it the key False
                if "not" in chunk or "n't" in chunk:
                    result[ent] = False
                else:
                    result[ent] = True
    return result
```

在本次学习当中，主要是通过‘not’，“n’t”来进行判断否定实体的存在，以此来添加否定实体“deny”，在对意图进行处理的时候，会先识别出来是否带有“deny”，如果带有的话，会再对其本身的意图进行一些相应的特殊处理。

3.4 状态机

状态机简单来说就是一个有向图形，由一组节点和一组相应的转移函数组成。状态机通过响应一系列事件而“运行”。每个事件都在属于“当前” 节点的转移函数的控制范围内，其中函数的范围是节点的一个子集。函数返回“下一个”（也许是同一个）节点。这些节点中至少有一个必须是终态。当到达终态， 状态机停止。

```
# Define the states
INIT=0
AUTHED=1
CHOOSE_COFFEE=2
ORDERED=3

# Define the policy rules
policy_rules = {
    (INIT, "order"): (INIT, "you'll have to log in first, what's your phone number?", AUTHED),
    (INIT, "number"): (AUTHED, "perfect, welcome back!", None),
    (AUTHED, "order"): (CHOOSE_COFFEE, "would you like Columbian or Kenyan?", None),
    (CHOOSE_COFFEE, "specify_coffee"): (ORDERED, "perfect, the beans are on their way!", None)
}
```

3.5 多轮问询

在聊天机器人设计的过程中，对话一个十分重要的部分，从而对话当然不仅仅只是一轮对话就结束了，所以多轮的对话，问询就成为了一个十分常见的过程，那么，如何让计算机知道进行到了哪一个部分的对话，或者是到了哪个状态，这个时候就需要使用状态机了，通过状态机来判断问询到了哪一个状态，进而实现多轮问询。

```
# Send the messages
send_messages([
    "I'd like to order some coffee",
    "555-12345",
    "do you remember when I ordered 1000 kilos by accident?",
    "kenyan"
])
```

```
USER : I'd like to order some coffee
BOT : you'll have to log in first, what's your phone number?
USER : 555-12345
BOT : perfect, welcome back!
BOT : would you like Columbian or Kenyan?
USER : do you remember when I ordered 1000 kilos by accident?
BOT : Yes .. and?
USER : kenyan
BOT : perfect, the beans are on their way!
```

4. 学习总结

首先非常幸运能够进入这次课程的学习，我本来是抱着增强自己的背景来加入该项目的，我的第一意愿并不是语言类的AI，而是视

觉或驱动类的AI，但可能也是一种缘分，让我加入到了这个学习小组当中，让我感到十分惊喜的是，聊天机器人的学习是十分有趣的，而且我不仅仅提升了自己的科研背景，而且学习了很多专业且前沿的AI技术，虽然本次学习是远程指导，并且学习周期也较长，鉴于我本身学习有一定的惰性，但是这次学习的趣味性，让我并没有荒废这一个月的学习，一个人的学习当然是有一定难度的，但是，每当我碰到自己不能解决的问题时候，在自己多次尝试却无果后，张老师总是能够及时给到我解答，在此十分感谢老师的指导和帮助，让我顺利的完成了这次的科研项目。

对于张老师每周布置的任务，大概可以用“复习上课的内容+资料的阅读+代码实现+熟悉代码”来概括，我个人认为这是一个比较适合我的学习方式，我会严格的执行，这样不仅仅加强了我对上课内容的熟悉程度，还让我拓展学习，培养自己的独立学习和解决问题的能力，在代码实现和要对代码熟记于心的部分对于我来说是有一定困难的，但是，一周的时间还是足够让我来完成这些任务，除此之外，我还会自己练习一些其他的Python代码，增强了自己编程和逻辑思维的能力。

我有一定的自信说对于自己接受新知识的能力和实践能力是较强的，但是，在本次学习当中，让我对自己有了新的认识，当老师要求的任务其实有些是超出了现在我的能力范围，比如对于一些未学到代码的实现或者一些较为复杂的逻辑实现，且时间又是固定的，我只能通过查阅各种资料，以及想老师或同组的同学交流来解决这部分问

题，但是，有时候我还是不能够完全理解或解决这些问题，所以，这部分是我还需要加强的。

当然在本次学习当中，正则表达式、词向量、意图识别、实体抽取、判定角色、多轮单次问询、多轮多次问询、否定问询、状态变迁、询问队列等核心技术的学习，以及 rasa_nlu 为聊天机器数据训练和处理框架，以 Spacy 为后台自然语言处理框架，对于聊天机器人的功能实现,都让我大开眼界，当然，还有许多前沿的 NLP、NLU，更加丰富的自然语言处理、语义理解、语音识别、语音合成等技术。我虽然并没有熟练掌握每一项技术，但是，我希望在以后的日子里，能够掌握更多的技术。最后,还是感谢在整个过程中老师的悉心指导和帮助,还有同组同学所提供的帮助。希望以后有机会还能在老师或和老师一样专业的教授的实验室继续学习。祝老师在未来的日子工作顺利，也希望自己能够更上一层楼!