好未来 Future Camp: NLP Week1 报告

吕晴 清华大学

Project repo: https://github.com/veronica320/FutureCamp

1 任务一：基于词袋向量计算句子相似度

1.1 基本思路

题目要求基于词袋模型选取与给定问题最相似的句子，思路比较简单：分别选取 TF 和 TF-IDF 作为特征，利用词袋模型将句子向量化；再基于得到的向量，比较 cosine 相似度，从而得到与给定问题最相似的对象。

1.2 实现步骤

- 读取题库数据，去除每一道题目中的非中文字符和停止词（中文停止词库来源：https://github.com/chdd/weibo/blob/master/stopwords/中文停用词库.txt），并进行分词。
- 利用 SKlearn 提供的 CountVectorizer 和 TfidfTransformer 等库函数，提取 TF 和 TF-IDF 特征，对题目进行词袋向量化。
- 随机抽取 5 道题目，计算每道与题库中所有其他题目的 cosine 相似度，选取最相似的三道。

1.3 评测结果展示

以下表格展示了题库中随机抽取的五道题目，利用 TF 和 TF-IDF 作为词袋特征，得到的最相似题目结果（为了美观，这里只展示前三道题目相似程度最高的一条结果，完整结果可以在项目 repo 中的 results 文件中查看）。

原题目	最相似题目（TF）	最相似题目（TF-IDF）
设点 Q 是曲线上的一个动点，求它到直线 l 的距离的最大值	设点 P 是曲线 C 上的一个动点，求它到直线 l 的距离 d 的取值范围	设点 P 是曲线 C 上的一个动点，求它到直线 l 的距离 d 的取值范围
求函数 $f(x)$ 的值域	求函数 $y=(\frac{1}{9})^x+(\frac{1}{3})^x+1$ 的值域	求函数 $y=(\frac{1}{3})^{x^2-4x}, x\in[0,5)$ 的值域
用函数单调性的定义证明函数 $f(x)$ 在区间 $(-1,+\infty)$ 上是增函数	当 $a=-1$ 时，利用函数单调性的定义证明函数 $f(x)$ 在区间 $(0,+\infty)$ 上是增函数	当 $a=1$ 时，利用函数单调性的定义证明函数 $f(x)$ 在 $(0,1]$ 内是单调减函数

表 1: 基于 TF 和 TF-IDF 作为词袋特征找出最相似题目

观察上表发现，TF 和 TF-IDF 作为特征时，找出的相似题目直观上来看都比较准确，也出现了两种方法结果一致的情况。可以认为，这项任务中词袋模型对句子向量化的结果还是比较理想的。

2 任务二：利用 SVM 和 Logistic Regression 进行句子分类

2.1 基本思路

在上一任务的基础上，这里我们需要利用 TF-IDF 词袋向量对句子进行分类。用 SVM 和 Logistic Regression 算法，直接将上题的向量作为特征进行训练即可。

2.2 实现步骤

- 读取数据，根据知识点名称（"name" 字段）在 knowledge_hierarchy 表格中筛选出三角函数与解三角形和函数与导数对应的 id，再同一张表格中找出以此为 root_id 的知识点 id。利用得到的知识点 id，在 question_knowledge_hierarchy_sx 表格中找出对应的 question_id，再在题库中筛选出对应题目。
- 对筛选出的题目进行清理（去除非中文字符，去除停止词，分词），并用任务一的方法提取 TF-IDF 特征。
- 构建特征空间和标签空间，划分训练集和测试集（这里用 7：3 的比例）。
- 采用 SKlearn 的 SVM.linearSVC 和 Logistic Regression 算法训练分类器，在测试集上评估表现。

2.3 评测结果展示

以下是 SVM 和 Logistic Regression 两种算法训练出的分类器，在测试集上的表现:

算法	Precision	Recall	F1 score	Accuracy	ROC_AUC_score
SVM	0.86	0.93	0.89	0.84	0.88
Logistic Regression	0.85	0.94	0.89	0.83	0.88

表 2: SVM 和 Logistic Regression 题目分类模型表现

对比发现，两种算法的表现差异不大（每项指标差异不超过 0.01）。除了以上结果之外，我还尝试了一下非线性的 SVM 分类器（SVM.SVC），并简单调了一下参 (tol, loss，C，max_iter) 等，同样并未发现显著提升，在此不赘述了。

3 任务三：基于预训练 Word2vec 词向量计算词语相似度

3.1 基本思路

这一任务需要借助已有的预训练词向量，找出与给定词语最相近的词。网上这样的资源很丰富，我从 https://github.com/Embedding/Chinese-Word-Vectors 下载了一个在 wikipedia 语料库上训练的词向量。接下来，只需载入并计算即可。

3.2 实现步骤

- 读取知识点体系数据，对知识点进行清理（去非中文字符，去停止词，分词），并统计其中词汇表。
- 下载 sgns.wiki.char 词向量，对其做过滤，只留下词汇表中出现的词，从而加快后续加载速度。
- 用 gensim 加载词向量，随机选取 5 个词，找出词库中与之最相似的词。

3.3 评测结果展示

以下展示了随机选取 5 个词，在词库中找到的与之最相近词语 TOP10，相似度由高至低排序:

目标词语	最相近词语（相似度从高至低）
生物	微生物, 生物学, 生物体, 生物群落, 生物进化, 生物膜, 动物, 水生动物, 植物, 原生动物
环境	生存环境, 生态环境, 环境治理, 环境保护, 保护环境, 环境污染, 环境温度, 环境影响, 适应, 水循环
方法	方式, 分析方法, 步骤, 算法, 法, 手法, 解方程, 技巧, 手段, 具体方法
中国	大陆, 中华民国, 中国共产党, 广东, 台湾, 历史, 美国, 全国, 亚洲, 香港
人类	人类文明, 全人类, 类人猿, 人脑, 动物, 鸟类, 生物, 进化, 人血, 人体

表 3: 基于预训练词向量找出的最相近词语

同样，我们发现结果还是相当好的，基本上与直觉相符。

4 任务四：基于自训练 Word2vec 词向量计算词语相似度

4.1 基本思路

这一任务与任务三思路基本一致，只需要把预训练词向量换成自训练就行了。使用 Gensim，在知识点语料库上训练即可。

4.2 实现步骤

- 与任务三一样，读取知识点体系数据，对知识点进行清理（去非中文字符，去停止词，分词），并统计其中词汇表。
- 用 gensim 在知识点语料库上训练词向量，保存得到的词向量模型。

– 读取上一步训练好的词向量模型，随机选取 5 个词，计算 cosine 相似度找出词库中与之最相似的词。

4.3 评测结果展示

同样，以下展示了与上一任务相同的五个词，使用自训练词向量在词库中找到的与之最相近词语 TOP10，相似度由高至低排序：

目标词语	最相近词语（相似度从高至低）
生物	中, 人类, 理, 问题, 法, 运动, 求, 特征, 基, 社会主义
环境	实验, 人类, 问题, 中, 关系, 发展, 主, 基, 生活, 特征
方法	中, 问题, 动物, 求, 法, 实验, 特征, 影响, 理, 济
中国	探究, 求, 问题, 济, 观察, 发展, 基, 中, 模型, 计算
人类	问题, 环境, 中, 济, 关系, 主, 发展, 探究, 求, 生物

表 4: 基于自训练词向量找出的最相近词语：baseline

与上一任务的结果对比，发现自训练向量的结果直观上不是很好：找到的词实际上和目标词的相差甚远；同时，找到的词中有很多重复的意义宽泛的词（比如“中”，“人类”，“主”，“问题”）等。我猜也许是这些词的先验概率较大，所以在语料库较小的情况下容易产生 bias。以上是默认参数（size=100，min_count=1，window=5 的结果）。由于知识点语料库的句子长度其实很短，一般情况下只有几个词，所以我尝试了一下调参，减小 window size——以下是 size=100，min_count=1，window=1 的结果：

目标词语	最相近词语（相似度从高至低）
生物	素, 影响, 半坡, 锯木, 条件, 部分, 济, 辗转, 达芬奇, 国民党
环境	推测, 赤壁, 行, 确定, 道德, 测量, 根腰, 维新, 追求真理, 绿叶
方法	中, 分布, 法, 遗传疾病, 模块, 相关性, 角速度, 思维, 指代, 练习
中国	问题, 中, 长方形, 发展, 社会主义, 生活, 历史, 图形, 数, 原理
人类	主, 无格线, 济, 问题, 形成, 约法, 中, 萌发, 四形, 名词

表 5: 基于自训练词向量找出的最相近词语：调参结果 1

可以发现，window size 变小后，重复宽泛词的问题有所缓解，但得到的结果还是不太 make sense。于是又调了一下词向量本身的 size，令 size=300，min_count=1，window=1：

目标词语	最相近词语（相似度从高至低）
生物	中, 观察, 求, 发展, 子结构, 量角器, 型, 中国, 实验, 天平
环境	图形, 基, 规律, 特征, 主, 协调, 性质, 运动, 结构, 形成
方法	求, 发展, 人, 基, 设想, 概念, 社会, 规律, 折线, 反身代词
中国	中, 济, 问题, 发展, 特征, 性质, 基, 运算, 作, 文化
人类	中, 主, 纳, 资源, 观察, 探究, 特征, 基, 位置, 末项

表 6: 基于自训练词向量找出的最相近词语：调参结果 2

发现 size 变大之后，重复宽泛词的问题复现了。后来又尝试了一些组合 (size:[100，200，300，500]，min_count:[1，3，5]，window:[1,3,5])，遗憾的是也并没有发现显著的提高。

5 任务五：利用 Word2vec 词向量进行句子分类

5.1 基本思路

这一任务是以上任务的综合，用 word2vec 来表示句子，再基于得到的句子向量进行分类，关键点在于如何从词向量得到句子向量，我尝试了两种方法：一种是简单地取句中所有词的词向量的平均值作为句子向量，另一种是用 gensim 的 doc2vec 直接在语料库上进行训练。

5.2 实现步骤

- 与任务二一致，先读取数据，根据知识点名称在题库中筛选出对应题目，对题目进行清理。
- 构建特征空间和标签空间，划分训练集和测试集。
- 利用预训练词向量平均值或 doc2vec 生成句子向量。
- 采用 SKlearn 的 SVM.linearSVC 和 Logistic Regression 算法训练分类器，在测试集上评估表现。

5.3 评测结果展示

以下是用预训练词向量平均值作为句子向量，SVM 和 Logistic Regression 分类器在测试集上的表现:

算法	Precision	Recall	F1 score	Accuracy	ROC_AUC_score
SVM	0.84	0.92	0.88	0.82	0.85
Logistic Regression	0.84	0.92	0.88	0.82	0.84

表 7: 利用词向量平均值生成的句子向量应用于题目分类的表现

对比用 doc2vec 生成的句子向量，使用同样算法的表现:

算法	Precision	Recall	F1 score	Accuracy	ROC_AUC_score
SVM	0.74	0.97	0.80	0.72	0.74
Logistic Regression	0.73	0.97	0.83	0.70	0.73

表 8: 利用 doc2vec 生成的句子向量应用于题目分类的表现

综合观察，发现 SVM 和 Logistic Regression 两种算法在使用同一句子向量模型时的表现差异不大，与任务二的观察一致。方法一（以词向量平均生成句子向量）的表现与任务二中词袋模型的表现几乎不相上下。相较之下，方法二（doc2vec 生成句子向量）的各项指标普遍来说都低于方法一，可能同样是由于训练语料库过小；但值得注意的是，方法二的 recall 比方法一高出许多，这一点还暂时不确定如何从直觉上解释。

6 问题和思考

这一周任务主要遇到的挑战在于以下两方面，也是希望与老师同学们交流的地方：

- 第一，如何在语料库规模较小的情况下训练出高质量的词向量？任务四的结果与任务三相比，召回的相似词语准确率显然下降了许多。这可能是因为语料库数据较少，或者调参不到位等缘故。由于时间限制，我没能进行大规模调参得到显著有提高的结果，但小范围尝试后发现效果不尽人意。除了调参之外，是否还有其他系统化的方法可以提高词向量质量?
  以及，除了人为观察相似词语结果之外，是否可以用其他定量方法进行更有效率的评估? 比如，可以在已有的评测任务数据集（比如 SemEval Word Similarity Task）上对训练出的词向量做质量评价。若时间允许，这可以作为进一步提升词向量质量的尝试方向。

- 第二，如何从词向量出发得到高质量的句子向量？目前学界对于词向量已经有比较成熟的成果，有大量预训练词向量可以直接拿来用。但如何将句子向量化依然是尚未解决的难题。这次在任务五中尝试了两个比较直接的方法，用词向量取平均和用 doc2vec 训练，发现直觉上更 naive 的方法一反而效果更好。当然，这可能与第一个问题相似，也同样受到了语料库规模的影响。
  在 NLP 研究中，我所了解的大多数任务对于句子表示比较常见的做法是将神经网络的第一层直接作为 Embedding 层。但最近又出现了一些新的尝试，比如 Allen NLP 新开发的 ELMo，在 contextualized word embedding 基础上得到句子向量。感觉相比于词语的向量化，句子向量化无论是在方法还是后续使用上都不是那么直观（比如，完全没有“预训练句子向量”这种东西的存在）。希望与大家探讨一下，如何找到更有效的高质量句子向量化的方法。