机器学习实验报告

——————消费意图检测

姓名：林聪　　学号：16339026　　日期：2019/4/15

摘要：消费意图检测旨在根据用户的文本内容推断用户是否有表现出对产品或服务的购买意图，它是这个互联网快速发展的大环境下，商家充分利用信息资源挖掘潜在商机的重要技术。消费意图检测的首要核心问题在于对文本数据进行向量化，主要的方案有自然语言处理技术中的词袋模型和词嵌入模型。本文提出了一种将词袋模型 TF-IDF 和词嵌入模型 Word2vec 结合的文本向量化方法 TFIDF2vec，从而将两者的优势相结合，并在数据集上获得了可观的效果。这次实验初步说明了本文所提出的 TFIDF2vec 方法具有可行性及进一步研究开发的潜力，经过改进后应当可以获得更好的效果。

1．导言

随着社交网络的发展和普及，微信朋友圈、新浪微博、推特等社交平台成为了广大群众发布、传播和共享信息的普遍场所。在这些平台的用户所公开发布的文本数据中隐含着用户在拥有某种需求的情况下，受一定的购买动机驱使，所表达出的对某种产品或服务的消费倾向或意图，也就是所谓的消费意图。在电子商务迅猛发展的现阶段，社交平台中蕴含的这些用户消费意图信息包含着巨大的商业价值，能够帮助商家更好地理解用户消费行为进而挖掘出潜在的目标客户，为其提供精准、及时的产品或服务推荐，从而将潜在的商机转化为实际的商业利益。

所谓消费意图检测，旨在根据用户在社交平台上发布的短文本数据，判断出短文本数据本身是否蕴含特定的消费意图检测信息，从而断定其用户是否有消费意图，即判断用户发布的一则朋友圈、微博或推特等是否传达出了对某种产品或服务的购买意愿。例如，用户短文本“谁可以陪我看失恋 33 天啊”明确传达出了用户有观看电影《失恋 33 天》的意向，从而该用户购买电影票或者享受观看电影服务的消费意图，故判断用户具有电影消费意图。而用户短文本“上海书展，为伊消得人憔悴”尽管可以认为表现出了购买书籍或参加书展的意向，但并未表现出对电影的消费意图，故应当判断用户不具有电影消费意图。

目前用于消费意图检测的模型主要有：1）基于弱监督的图排序算法。该方法适用于数据总量较大，已标注数据较少的情况；2）基于领域自适应的消费意图检测方法。该方法采用了领域自适应的卷积神经网络模型进行二元分类；3）基于社区问答的消费意图识别和挖掘方法。该方法通过挖掘问题与回答之间的需求对应关系，把消费对象的挖掘问题转化为搭配问题；4）基于模板匹配的方法。该方法提出了人们日常生活中经常出现的 12 种普遍需求，通过模板匹配方法获得了购买了产品的用户，然后基于 unigram 特征、WordNet 中词的语义特征、表达需求的动词特征来训练分类器完成对用户消费意图的识别。

为了提取文本的语义特征，消费意图检测方法的首要任务是将文本数据向量化，其关键在于如何将分词后的文本表示成计算机能进行计算的类型，即如何建立特征工程的问题。对此，在自然语言处理领域有两种常用的表示模型，分别为词袋模型和词嵌入模型。

词袋模型，即不考虑词语在原本文本中的顺序，直接将每个词语或字符统一地放入一个词袋集合中，在按照计数的方式对词语或字符的出现次数进行统计而得的表示模型。在各种

词袋模型中，TF-IDF (Term Frequency-Inverse Document Frequency)，词频-逆文件频率模型，是一种经典的模型。词频指的是某一个特定词语在文本中的出现次数，通常归一化为出现频率，即对某一给定的文本 d 和词语 w，我们有

$$TF_{d,w} = \frac{w在d中出现的次数}{d中词语的总数}.$$

逆向文件频率则用于衡量词语 w 对文本的区分能力，它的定义可写为

$$IDF_w = \log\left(\frac{文本总数}{1 + 包含w的文本数}\right).$$

将以上两者相乘便得到了文本 d 在词语 w 上的 TF-IDF 值：

$$TF - IDF_{d,w} = TF_{d,w} * IDF_w.$$

我们可以根据文本中每一个词语的 TF-IDF 值来对文本向量化，该模型的优点在于简单快速、可解释性强，得到的效果也符合预期，但主要的缺点在于忽略了文本中词语的相对位置，同时生成的特征向量维数和稀疏度过高，这会对后续的训练带来不便。

词嵌入模型则以 Word2vec 为代表，该模型实质为简化的神经网络，其原理类似自编码器，它以词语的 One-hot 向量作为输入，经过两层神经网络之后得到一个与输入向量维数相同的概率分布向量，模型所采用的损失函数通常为交叉熵损失函数，它将使得输出向量逼近输入向量所代表的词语的 One-hot 向量。具体而言，Word2vec 模型又分为 CBOW (Continuous Bag-of-Words)模型和 Skip-Gram 模型，在本次实验中我们将采用适合小数据集的前者，它的特点在于将词语上下文的词语的 One-hot 向量作为输入，输出该词语恢复重建后的概率分布向量。模型训练完成之后，我们真正需要的是模型中间层的输出，即词语的特征向量，之后可以通过拼接或求平均的方式有文本词语的特征向量构建文本的向量。该模型的优点在于充分考虑了文本中词语的上下文信息，缺点则在于不能强调特定词汇在文本中的特殊地位或作用。

本次试验我试图探究一种将 TF-IDF 与 Word2vec 结合的方法，称之为 TFIDF2vec。该方法有着朴素的核心思想，即以词语的 TF-IDF 值为权重对词语的 Word2vec 向量进行加权求和，从而得到文本的特征向量。通过这种方式，TFIDF2vec 可以将文本向量的维度和稀疏度大大降低，并且能够兼顾文本中词语的重要程度，即融合了 TF-IDF 和 Word2vec 的优势并缓解它们的缺陷。

2．实验过程

TFIDF2vec 算法先将超过一定词频的词语转化为 TF-IDF 向量和 Word2vec 向量，相乘得到词语的 TFIDF2vec 向量。之后对于给定文本，找出其中在语料库中超过一定词频的词语，获取词语的 TFIDF2vec 向量后，将 TFIDF2vec 相加，便得到了文本的 TFIDF2vec 向量。

在本次实验中，有三个带有 0/1 标签的、仅做好分词的数据集，分别为 movie 数据集、flight 数据集和 laptop 数据集，它们各自包含训练集 train、验证机 val 和测试集 test 三个子集。实验要求对三个数据集分别训练消费意图检测模型，模型要能够检测出特定数据集所对应的特定消费意图。

为了建立模型，我们首先对数据集进行预处理，将其中的标点符号等无实际含义、与本次试验任务无关的符号去除，并将字符串文本转换为列表数据。之后分别用已有的 TF-IDF 计算模型和 Word2vec 计算模型求得训练集中词频不低于最小词频的词语的 TF-IDF 值和 Word2vec 向量，再将文本词语的 TF-IDF 值与其 Word2vec 向量相乘后相加的到文本的 TFIDF2vec 向量。为了简化问题，我们采用最简单的 Logistic 分类器在训练集上训练消费意

图检测模型，并采用随机梯度对模型进行优化更新，每到一定阶段用验证集进行验证并最终在测试集上进行测试，测试的指标包括准确率、覆盖率和 F1 调和平均。此外，我们还将训练基于单纯的文本 TF-IDF 向量的消费意图检测模型和基于单纯由 Word2vec 向量相加而得的文本向量的消费意图检测模型作为 Baseline。

3．结果分析

本次实验在 Window10 系统的 jupyter notebook 5.6.0 中用 python 3.6.4 进行，其中 TF-IDF 模型和 Logistic 模型由 scikit-learn 0.19.1 提供，Word2vec 模型由 gensim 3.7.2 提供。我们取最小词频为 10，即词频应当大于等于 10。对 Word2vec 模型 CBOW 的主要参数，我们取特征向量的维数为 50，窗口大小为 5，训练迭代次数为 5，学习率为 0.025，并考虑到 Word2vec 的结果数值相比 TF-IDF 往往较小，故将结果放大到 100 倍。对于 TF-IDF 的主要参数，在 TF-IDF Baseline 中取结果的 L2 范数，而在 TFIDF2vec 中不做正则化以维持结果数值的数量级。在每次训练中，Logistic 模型进行 500 个 epoch，优化方法为随即平均梯度下降法，并采用默认的自适应最优步长。

实验在每个数据集上各进行 5 次，所得的结果如下表所示：

		Best	Worst	Average
TFIDF	Time(s)	56.1	59.7	57.24
	Precision	0.9650	0.9614	0.9626
	Recall	0.9642	0.9606	0.9620
	F1-score	0.9641	0.9605	0.9619
TFIDF2vec	Time(s)	8.59	9.06	8.78
	Precision	0.9320	0.9176	0.9235
	Recall	0.9319	0.9168	0.9232
	F1-score	0.9319	0.9168	0.9232
Word2vec	Time(s)	8.68	8.95	8.84
	Precision	0.9145	0.8985	0.9046
	Recall	0.9122	0.8925	0.9018
	F1-score	0.9121	0.8922	0.9016

表 1：movie 数据集结果

		Best	Worst	Average
TFIDF	Time(s)	124	127	126
	Precision	0.9513	0.9457	0.9487
	Recall	0.9511	0.9456	0.9486
	F1-score	0.9511	0.9456	0.9486
TFIDF2vec	Time(s)	15.6	16.0	15.84
	Precision	0.9497	0.9488	0.9492
	Recall	0.9495	0.9487	0.9490
	F1-score	0.9495	0.9487	0.9490
Word2vec	Time(s)	15.6	16.5	15.86
	Precision	0.9355	0.9316	0.9333
	Recall	0.9345	0.9306	0.9323
	F1-score	0.9345	0.9306	0.9322

表 2：flight 数据集结果

		Best	Worst	Average
TFIDF	Time(s)	65	68	66
	Precision	0.9421	0.9350	0.9389
	Recall	0.9416	0.9350	0.9385
	F1-score	0.9416	0.9350	0.9385
TFIDF2vec	Time(s)	6.72	7.07	6.83
	Precision	0.9230	0.9128	0.9174
	Recall	0.9165	0.9125	0.9146
	F1-score	0.9164	0.9125	0.9145
Word2vec	Time(s)	6.64	6.98	6.81
	Precision	0.9207	0.9136	0.9177
	Recall	0.9191	0.9111	0.9156
	F1-score	0.9190	0.9110	0.9156

表 3：laptop 数据集结果

从以上结果可以得出：

- 从准确率、召回率和 F1 指标来看，TFIDF2vec 模型能够在不逊于 Word2vec 模型的基础上追赶 TFIDF 模型，其中在 laptop 数据集上 TFIDF2vec 模型与 Word2vec 模型的效果相差无几，但在 flight 数据集上 TFIDF2vec 模型的效果则能超过 Word2vec 和 TFIDF 模型，而在 moive 数据集上 TFIDF2vec 的效果则接近 Word2vec 和 TFIDF 模型效果的平局值；
- 从耗时角度来看，TFIDF2vec 与 Word2vec 的耗时几乎相等，并且远小于直接用 TFIDF 训练分类检测模型的耗时。这主要的原因应该在于基于 TFIDF 的文本向量的维数和稀疏度过高，不利于分类器的训练；

  综合以上两个方面可以看出，TFIDF2vec 模型在大幅提升速度的优势下仍能够保证较高的训练效果，即能够在开销和效果之间找到一个很好地平衡点。不过也可以看出，该模型的效果在三个数据集上的表现不一，仍有不小的提升空间，这有待进一步的研究和改进。鉴于 TFIDF2vec 的效果总处于 Word2vec 和 TFIDF 之间，可以考虑进一步强调文本中词汇的特殊作用，从而使效果向 TFIDF 靠拢。

4．结论

本次实验之所以选择了在文本向量化表示上试图做一些创新，是因为在必修的数据挖掘课程上正好学到了 TFIDF 和 Word2vec，从而对两种模型的结合产生了某种想法，进而进行了实验。尽管这次实验提出的 TFIDF2vec 方案比较朴素，但从道理上来讲是行得通而且可解释的，并且从实际角度来讲它也符合对 TFIDF 和 Word2vec 两种方案的折中预期。当然，加入仅仅是这样那就没什么意思了，但事实上我们看到 TFIDF2vec 方法在效果和速度上是可以达到甚至超过 TFIDF 模型的效果和速度的，因而本次试验初步说明了这种方法具有提升空间和一定的研究价值。后续的研究和提升可以将重点放在如何更好地在 TFIDF2vec 中强调词汇的特殊地位，从而让该方法在保有 Word2vec 的特点的同时向 TFIDF 做一定的靠拢，从而进一步让 TFIDF2vec 的性能向 TFIDF 靠拢或超过后者。

主要参考文献：

[1] 面向社会媒体的消费意图识别:任务、挑战与机遇[J]. 付博,刘挺. 智能计算机与应用.

2015(04)
[2] 聊天机器人中用户出行消费意图识别方法[J]. 钱岳,丁效,刘挺,陈毅恒. 中国科学:信息科学. 2017(08)
[3] gensim: models.word2vec – Word2vec embeddings.
https://radimrehurek.com/gensim/models/word2vec.html
[4] sklearn: sklearn.feature_extraction.text.TfidfTransformer.
https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html