封面页

目录页

• Executive Summary (summarize your goal and your main finding in a

nontechnical language)

执行摘要:通常在1-2页内概述整个报告的主要发现、结论和建议,适合高层管理者快速了解报告要点。

1. **INTRODUCTION**

说明项目背景、研究目的、范围和方法等,让读者了解研究的上下文。

说明下用nlp预测文本准确度的发展脉络，大概就是最开始都用什么方法研究，后面用什么方法研究，我们用什么方法研究。

1. **METHODS**

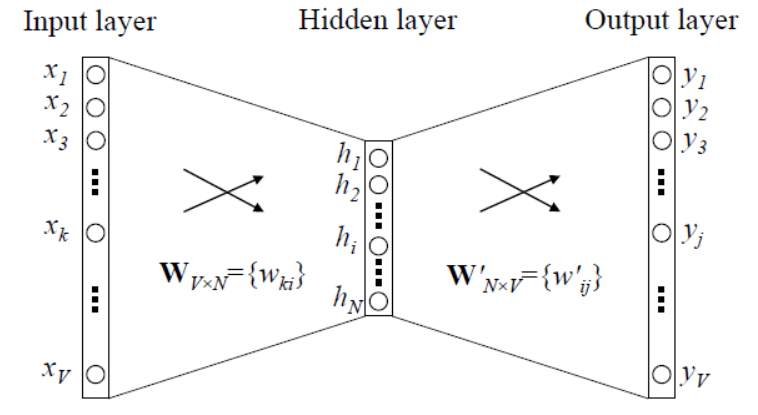
我们研究的主要目标是通过fund的summary来预测它的投资策略。这里面主要分为两部分，第一部分是把文本转化为能够被计算机理解的向量，我们在2.1中详细阐述。第二部分是通过向量拟合模型预测这个fund的投资策略，这个在2.2中阐述。

* 1. ***文本向量化***
     1. Skip-gram Model

Skip-gram是由Mikolov[[[1]](#endnote-0)]等人在2013年提出的一种词向量训练方法,属于Word2Vec模型的一种。

Skip-gram模型的基本思想是:一个词可以用来预测其上下文中的词。举例来说,给定一个句子"The quick brown fox jumps over the lazy dog",以"fox"为中心词,设定一个窗口大小(skip\_window)为2,那么它的上下文就包括"The","quick","brown","jumps","over",模型的目标就是利用单词"fox"来预测这些上下文单词。

训练过程中,模型通过调整词向量的参数,使得中心词的词向量能够较好地预测其上下文词的概率。这个思路的背后是分布式假说,即语义相似的词出现在相似的上下文中。因此训练好的词向量能够捕捉单词之间的语义相似性。



这个图是Word2vec的结构

具体来说我们建立Skip-gram Model的方法是这样的：首先将文本转化为单词列表,建立单词到索引的映射关系,并将单词转化为one-hot向量表示。然后生成训练数据，在每个epoch从文本中随机采样中心词及其上下文词,并将它们转化为one-hot向量。使用Keras定义一个自编码器结构的神经网络模型。编码器部分将one-hot向量映射为低维词向量,解码器部分再将词向量映射回one-hot向量。Skip-gram模型只使用编码器部分。模型参数如下表所示：

| 参数名称 | 值 | 说明 |

|----------|----------|----------|

| batch\_size | 128 | 每个批次的样本数 |

| num\_epochs | 2 | 训练的轮数 |

| embedding\_size | 50 | 词向量的维度,决定了词向量的表达能力 |

| max\_vocabulary\_size | 5000 | 词汇表的最大大小,按照词频从高到低排序,取前max\_vocabulary\_size个词 |

| min\_occurrence | 10 | 一个词被纳入词汇表的最小出现次数,用于过滤低频词,减少噪音 |

| skip\_window | 3 | 上下文窗口的大小,考虑中心词前后各3个词作为其上下文 |

| num\_skips | 4 | 每个中心词采样的上下文词数量,在每个窗口中随机选择4个上下文词来预测中心词 |

| vocab\_size | 2999 | 词汇表的实际大小 |

| Total params | 302949 | 模型总参数数量 |

使用自编码器的训练方式训练模型,目标是最小化解码器的输出与实际上下文词的one-hot表示的交叉熵损失。模型训练完成后,编码器部分就可以将每个单词映射为一个低维词向量。最后遍历所有的单词,得到词到向量的映射字典，这个字典中的每个向量就存储了这个词以及它对应的上下文信息。

* + 1. BERT部分

BERT(Bidirectional Encoder Representations from Transformers)[[[2]](#endnote-1)]是一种预训练的语言表示模型,它使用了Transformer[[[3]](#endnote-2)]的编码器结构,并在大规模无标签文本数据上进行了预训练。BERT的核心思想是通过jointly conditioning on both left and right context,学习词语的上下文表示。这使得BERT能够在各种NLP任务上取得很好的效果。

在我们的项目中，我们先加载预训练的BERT模型和对应的分词器(tokenizer)。这里使用的是'distilbert-base-uncased'[[[4]](#endnote-3)]模型,这是一个经过蒸馏(distillation)的BERT模型,体积更小,推理速度更快,但性能与原始BERT相近。

接下来将文本转换为BERT的输入格式。使用分词器对文本进行分词，如果分词后的tokens数量超过BERT的最大输入长度(这里设为512),则截断tokens。在tokens的开头和结尾分别添加特殊的`[CLS]`和`[SEP]`标记。将tokens转换为对应的ID。创建attention mask,表示tokens的哪些位置是实际的词,哪些位置是padding。如果tokens的长度不足最大长度,则用0进行补齐。

最后使用DistilBERT模型对每个输入进行处理,提取特征。

* 1. ***模型预测部分***
     1. ***CNN***

卷积神经网络（CNN）是一种深度学习算法。CNN在多个领域，特别是在图像和视频分析、自然语言处理等方面展示了出色的性能。CNN的概念最早可以追溯到20世纪80年代，由Yann LeCun[[[5]](#endnote-4)]等人在1990年代提出，并应用于手写数字识别问题。随着硬件性能的提升和大数据的可用性，CNN开始广泛应用于各种图像识别任务中，尤其是在2012年AlexNet[[[6]](#endnote-5)]在ImageNet[[[7]](#endnote-6)]竞赛中的成功之后，CNN的研究和应用得到了极大的推动。

CNN通常由三种类型的层组成：卷积层、池化层和全连接层。

卷积层：是CNN的核心，用于从图像中提取特征。通过对输入图像进行滤波器（或称为卷积核）的滑动窗口操作，生成多个特征图（feature map），每个特征图代表图像数据中的一种特征。

池化层：又称作下采样层，用于减少特征图的空间大小，从而减少计算量和过拟合的风险。常见的池化操作包括最大池化和平均池化。

全连接层：位于CNN的末端，其将前面卷积层和池化层提取并压缩的特征信息用于最终的分类或回归任务。

我们的模型架构（画图）

1. BatchNormalization层:对卷积层的输出进行批标准化,加速收敛并提高泛化能力。
2. MaxPooling1D层:对卷积的输出进行最大池化操作,池化大小为3。这有助于提取最重要的特征。
3. 第二个卷积层:使用128个大小为3的卷积核,激活函数为ReLU。
4. BatchNormalization层
5. MaxPooling1D层
6. 第三个卷积层:使用64个大小为3的卷积核,激活函数为ReLU。
7. BatchNormalization层
8. MaxPooling1D层
9. Flatten层:将三维的卷积输出铺平为一维,以便输入到全连接层。
10. 第一个全连接层(Dense):包含512个神经元,激活函数为ReLU,使用L2正则化来防止过拟合。
11. Dropout层:随机丢弃30%的神经元,进一步防止过拟合。
12. 第二个全连接层:包含256个神经元,激活函数为ReLU,使用L2正则化。
13. Dropout层:随机丢弃30%的神经元。
14. 输出层:包含3个神经元,激活函数为softmax,用于3分类问题。
    * 1. ***RNN***

RNN(循环神经网络)是一种广泛用于处理序列数据的神经网络架构。与传统的前馈神经网络不同,RNN引入了状态的概念,允许信息在序列的时间步之间持久存在。这使得RNN能够捕捉序列数据中的长期依赖关系,在自然语言处理、语音识别、时间序列预测等领域取得了显著的成果。

RNN的概念最早由Hopfield[[[8]](#endnote-7)]在1982年提出,其后Elman和Jordan[[[9]](#endnote-8)]等人在1990年代初进一步发展了RNN。然而,由于梯度消失和梯度爆炸问题,传统RNN难以捕捉长期依赖。为了解决这一问题,研究者提出了长短期记忆网络(LSTM)[[[10]](#endnote-9)]和门控循环单元(GRU)[[[11]](#endnote-10)]等改进的RNN变体。这些模型通过引入门机制和记忆单元,有效地缓解了梯度问题,使RNN能够学习更长的序列依赖。

我们的模型架构（画图）

1. Embedding层:将词汇表中的每个词映射为一个固定维度的稠密向量。这里使用了预训练的词向量(embedding\_matrix),并设置为不可训练。
2. 第一个双向LSTM层:LSTM是RNN的一种变体,能够更好地捕捉长期依赖关系。双向LSTM在序列的两个方向(前向和后向)上运行,能够同时利用过去和未来的上下文信息。这里将return\_sequences设置为True,以返回LSTM在每个时间步的输出。
3. Dropout层:随机丢弃20%的神经元,有助于防止过拟合。
4. 第二个双向LSTM层:与第一个双向LSTM层类似,进一步提取序列的高级特征。
5. Dropout层:随机丢弃20%的神经元。
6. 第三个双向LSTM层:这一层不返回序列,而是返回最后一个时间步的输出,作为整个序列的表示。
7. 第一个全连接层(Dense):包含与词向量相同数量的神经元,激活函数为ReLU。这一层将LSTM的输出转换为更高级的特征表示。
8. Dropout层:随机丢弃20%的神经元。
9. 输出层:包含3个神经元,激活函数为softmax,用于3分类问题。
10. **EMPIRICAL ANALYSIS**

***3.1数据***

我们希望完成的目标是用mutual fund text summaries来预测每个fund用的investment strategy。

介绍下MutualFundLabels.csv的情况，它存储了mutual fund的一些基础信息：id、fund\_name、Performance fee?、Ivestment Strategy、Leverage?、Portfolio composition、Concentration。

Performance fee?这一列缺失的数据达到447个，所以我们直接删去这个自变量。

数据中需要预测的investment strategy总共有五类，

Ivestment Strategy

Equity Long Only (Low Risk) 248

Fixed Income Long Only (Low Risk) 130

Balanced Fund (Low Risk) 84

Long Short Funds (High Risk) 4

Commodities Fund (Low Risk) 1

但是第四类和第五类都很少，drop掉。

这样剩下462个观测。

fund\_name里面存储了mutual fund的name，然后它能够和summary一一匹配上。我们通过fund\_name把txt格式的summaries和表格中的lable匹配上。

原始数据共有545个summary。其中84个summary没有对应的lable，把这些没有lable的summary整理出来作为测试集。留下的summary和整理好的MutualFundLabels.csv中的数据拼起来，整理成我们的原始数据集。

现在划分训练集和验证集。按照8:2的比例划分训练集和验证集。得到的训练集有368

个观测，验证集有93个观测。

下面进行数据清洗，因为我们用于预测的输入是文本，所以要把完整的句子整理为单词，并且去除没有意义的停用词。具体来说，将文本中的换行符\n和制表符\t替换为空格,并将文本转换为小写；使用来自nltk库的word\_tokenize函数对文本进行分词,将文本拆分为单词列表；过滤掉停用词。停用词是指在文本处理中通常被忽略的常见词,如"the"、"a"、"an"等；通过正则表达式过滤掉非字母字符，返回过滤后的单词列表filtered\_sentence。

使用Skip-gram Model来做词汇的向量化。详细的Skip-gram Model的说明在2.1.1有介绍。

使用词形还原(Lemmatization)将一个单词还原为其基本形式。通过词形还原,我们可以将文本中的单词统一为其基本形式,从而减少词汇的多样性,提高文本处理的效率和准确性。这里我们用的是NLTK自然语言处理库提供的词形还原器(WordNetLemmatizer)来进行词形还原。

有关创建knowledge\_base。使用TF-IDF提取关键词。对每个关键词,使用预训练的词向量模型找到与其最相似的一些单词。将关键词和相似单词组合起来,构建一个扩展的知识库。

TF-IDF的计算公式如下:

* TF(Term Frequency):一个词在一个文档中出现的频率。
* IDF(Inverse Document Frequency):一个词在整个语料库中的稀有程度。IDF = log(总文档数 / 包含该词的文档数)。
* TF-IDF = TF \* IDF

TF-IDF的值越高,说明一个词在一个文档中出现的频率越高,且在整个语料库中越稀有,因此它对于该文档的重要性越高。

接下来计算句子与知识库的相关性:

* 将句子中的词向量求平均,得到句子的重心向量。
* 计算句子重心与知识库中每个关键词的余弦距离。
* 取前n\_closer个最小距离的平均值作为句子的相关性得分。

最后,函数返回得分最小的num\_sent个句子,即与知识库最相关的句子。

通过这样的方法把训练集验证集测试集中的关键句子筛选出来。

另一种数据处理的方式是直接用预训练好的bert模型直接把summary的文本进行向量化，转化成能够输入到神经网络中的向量，具体的细节在2.1.2中有说明。

***3.2 训练结果***

参数设置及训练结果

cnn

1. 优化器:使用随机梯度下降(SGD)优化器,学习率为0.01,动量为0.9,使用Nesterov加速梯度。
2. 损失函数:使用分类交叉熵(categorical\_crossentropy)作为损失函数,适用于多分类问题。
3. 评估指标:使用准确率(accuracy)作为评估指标。

Rnn

模型使用分类交叉熵作为损失函数,Adam作为优化器,准确率作为评估指标。

在训练过程中,使用ModelCheckpoint回调函数在每个epoch后保存验证集上性能最好的模型。训练800个epoch,每个batch包含128个样本。

语言部分：训练平台用的谷歌的线上colab，用的T4。。。（那种都一样写在这里，不一样的写进表格）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| model |  |  |  |  |
| epoch | - | 100 | 100 | 200 |
| batch\_size | - | 1024 | 1024 | 2048 |
| learning\_rate | - | 0.01 | 0.001 | 0.001 |
| 损失函数 |  |  |  |  |
| Training time | 2s | 7m17s | 1m16s | 3m46s |
| 优化器 | 0.00082 | 0.00017 | 0.00022 | 0.00039 |
| 评估指标 |  | 0.01320 | 0.01497 | 0.01983 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

通过Table 3我们对比了不同Text Vectorization Method和不同神经网络模型在Validation Set上的预测表现，附上了一部分的参数设置。为了方便比较模型和Text Vectorization Method对训练结果的影响，我们特别控制epoch、batch\_size和learning\_rate相同。首先对于经过了Feature Selection的数据来说可以看到RNN模型表现的最好，在训练时间和F1-score、AUC、Accuracy Score等bench mark上的表现都是最好的。对比CNN\_Small和CNN\_Large，虽然参数量变大了3倍，训练时间也变长了，但是CNN\_Large的表现并没有提高很多，它们获得了非常近似的AUC和Accuracy Score ，甚至数值还低一些。我们猜测原因在与稍微大一些的模型可能对训练集产生了过拟合，反而影响了它在验证集上的表现。

而使用预训练好的BERT进行文本向量化，再投入模型中，发现预测的准确度明显好了很多。主要原因是这个数据没有进行截断，比起经过了Feature Selection的数据保留了更多信息。这个时候参数量较大的模型训练结果就比参数量小一些的模型表现好很多，因为有足够的信息供模型去学习。所以对比所有的模型，通过BERT提取词向量在CNN\_Large上进行训练的模型表现最好，在验证集上的F1-score (Average)为0.85，AUC为0.8568，Accuracy Score 为0.8495。我们用这个模型预测测试集中fund的投资策略，预测结果存储在y\_test\_CNN.csv中。**Figure 2展示的是**ROC Curve of BERT\_CNN\_Large，可以看见拟合效果很好。

另外我也尝试了直接通过预训练好的BERT模型把训练集作为下游任务，进行训练后再在验证集上预测，但是模型的表现不尽如人意。一方面是原始的BERT参数量非常多，即便是在下游上更新部分参数，跌倒一个epoch的时间也和在我们自定义的模型上训练800次的时间差不多。在BERT上更新参数可以看到很快的准确率上升，但是上升到60%左右就会进入平台期，无法有更好的表现。我们推测原因在于原始的BERT是针对通用文本的，而不是金融文本，所以在这个fund的summary上的预测先有一个提升后面就不能持续了。

需要补充的一点是，在多次测训练过程中，模型每次训练出的结果并不完全相同，相应的一些benchmark上的表现也会有细微的差别，但是总体来说会稳定在变化不大的一个范围里面。

就是不同方法和模型上的表现，这里要画图，就是roc曲线，一页多图画到一起

出表，比较区别

1. **CONCLUSIONS AND DISCUSSIONS**

**Contribution**

**REFERENCE**

**Appendix**

1. [] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013). [↑](#endnote-ref-0)
2. [] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018). [↑](#endnote-ref-1)
3. [] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017). [↑](#endnote-ref-2)
4. [] Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv preprint arXiv:1910.01108 (2019). [↑](#endnote-ref-3)
5. [] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86, no. 11 (1998): 2278-2324. [↑](#endnote-ref-4)
6. [] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." Communications of the ACM 60, no. 6 (2017): 84-90. [↑](#endnote-ref-5)
7. [] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009. [↑](#endnote-ref-6)
8. [] Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences 79, no. 8 (1982): 2554-2558. [↑](#endnote-ref-7)
9. [] Elman, Jeffrey L. "Finding structure in time." Cognitive science 14, no. 2 (1990): 179-211. [↑](#endnote-ref-8)
10. [] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780. [↑](#endnote-ref-9)
11. [] Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014). [↑](#endnote-ref-10)