# 房产行业聊天问答匹配


杨逸君*
201800302011
scottyang924@163.com
18 级人工智能班
ShanDong University
Qingdao, ShanDong, China

朱炳文
201800180145
201800180145@mail.sdu.edu.cn
18 级人工智能班
ShanDong University
Qingdao, ShanDong, China

马方洲
201800180088
1665130647@qq.com
18 级大数据班
ShanDong University
Qingdao, ShanDong, China



GOAT
Greatest of All Time


## ABSTRACT

本文介绍我们解决的问题——房产行业聊天问答匹配，主要任务是问答匹配。赛方提供了训练集与测试集，训练集与测试集都包含客户问题文件和经纪人回复两个文件。使用的评价指标是 F1 score。

在工作中，杨逸君主要负责 NLP 模型的选择、训练与优化，朱炳文主要负责数据的处理与增强，马方洲主要负责机器学习方面的尝试与模型融合。以入手赛题为目的，我们一开始结合信息检索与数据挖掘课程内容，尝试机器学习的线性回归与 TF-IDF 等方法，但线上评测效果不佳。之后，我们入门 NLP 领域，了解 NLP 的一些基本概念，并使用 NLP 领域的方法解题，着重采用的就是谷歌提出的 BERT 模型，并在其基础上作了后接层的优化，尝试了各种相关的预训练模型如 RoBERTa 等，并结合评测结果对单模进行模型融合融合方法包括 Voting、Stacking、Blending 等，得到不错的成绩；此外我们尝试用不同的方法增强数据集包括 EDA 加噪、文本回译，还通过研究训练集特点，创新了文本拼接的数据增强方法。在比赛尾声，我们借助华为云的 ModelArts 跑了一次 Large 模型，并进一步融合模型，获得当前最佳方案。

最终，A 榜分数 0.78027499 分，第 112 名，超越 461 支队伍；B 榜分数 0.79319546 分，第 116 名，超越 462 支队伍。



## KEYWORDS

QA Matching, Linear Regression, Natural Language Process, BERT, RoBERTa, Data Augmentation, Model Ensemble


## 1 INTRODUCTION

我们的工作围绕 2020CCF 大数据与计算智能大赛（CCF BDCI）中的房产行业聊天问答匹配问题展开。在帮助客户实现更美好的居住过程中，客户会和服务者（房产经纪人）反复深入交流对居住的要求，这个交流发生在贝壳 APP 上的 IM 中。

IM 交流是双方建立信任的必要环节，客户需要在这个场景下经常向服务者咨询许多问题，而服务者是否为客户提供了感受良好、解答专业的服务就很重要因此，需要准确找出服务者是否回答了客户的问题，并进一步判断回答得是否准确得体。我们的任务是：给定 IM 交流片段，片段包含一个客户问题以及随后的经纪人若干 IM 消息，从这些随后的经纪人消息中找出一个是对客户问题的回答。

随着产业平台规模的不断扩大，人工资源逐渐不足，需要 AI 参与这个过程。通过解决这类问题，我们将深度学习知识应用于十分贴近实际生活的房产行业聊天问答匹配问题，这不仅对于房产行业，对于其他行业的相关工作也有启发作用。另外，这是一个 NLP 问题，我们在课上接触不多，算是比较全新的领域，因此我们想通过这道赛题深入了解深度学习对于解决这类问题的效果。

评价指标使用 F1 score：2 * (精度 * 召回) /（精度 + 召回）。其中，精度指识别为 1 并且真实标签为 1 的经纪人回复条数 / 识别为 1 的经纪人回复条数；召回指识别为 1 并且真实标签为 1 的经纪人回复条数 / 真实标签为 1 的经纪人回复条数。

## 2 RELATED WORK

### 2.1 TF-IDF

TF-IDF（term frequency–inverse document frequency）是一种用于信息检索与数据挖掘的常用加权技术。TF 是词频 (Term Frequency)，IDF 是逆文本频率指数 (Inverse Document Frequency)。TF-IDF 普遍用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 加权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。除了 TF-IDF 以外，因特网上的搜索引擎还会使用基于链接分析的评级方法，以确定文件在搜寻结果中出现的顺序。

### 2.2 MAE

MAE 是指平均绝对误差（Mean Absolute Error），含义为观测值与真实值的误差绝对值的平均值。若真实值为 $y = (y_1, y_2, ..., y_n)$，模型的预测值为 $y = (\hat{y_1}, \hat{y_2}, ..., \hat{y_n})$，那么该模型 MAE 的计算公式为

$$MAE = \sum_{i=1}^{n} \frac{|y_i - \hat{y_i}|}{n} \tag{1}$$

### 2.3 Basic concepts of NLP

- 分词：是将一串书面语言分成其组成词的问题。中文分词指的是使用计算机自动对中文文本进行词语的切分，即像英文那样使得中文句子中的词之间有空格以标识。中文分词被认为是中文自然语言处理中的一个最基本的环节。
- 词向量：是 NLP 中语言模型与表征学习技术的统称。概念上而言，它是指把一个维数为所有词的数量的高维空间嵌入到一个维数低得多的连续向量空间中，每个单词或词组被映射为实数域上的向量。词嵌入的方法包括人工神经网络、对词语同现矩阵降维、概率模型以及单词所在上下文的显式表示等。在底层输入中，使用词嵌入来表示词组的方法极大提升了 NLP 中语法分析器和文本情感分析等的效果。
- Seq2Seq：是用于 NLP 的一系列机器学习方法。应用领域包括机器翻译，图像描述，对话模型和文本摘要。Seq2seq 将输入序列转换为输出序列。它通过利用循环神经网络（递归神经网络）或更常用的 LSTM/GRU 网络来避免梯度消失问题。当前项的内容总来源于前一步的输出。seq2seq 主要由一个编码器和一个解码器。编码器将输入转换为一个隐藏状态向量，其中包含输入项的内容。解码器进行相反的过程，将向量转换成输出序列，并使用前一步的输出作为下一步的输入。
- NER：是指识别文本中具有特定意义的实体，主要包括人名、地名、机构名、专有名词等，以及时间、数量、货币、比例数值等文字。指的是可以用专有名词（名称）标识的事物，一个命名实体一般代表唯一一个具体事物个体，包括人名、地名等。NER 属于从非结构化文本中分类和定位命名实体感情的子任务，其过程是从是非结构化文本表达式中产生专有名词标注信息的命名实体表达式。
- 文本分类：和其他的分类没有本质的区别，核心方法为首先提取分类数据的特征，然后选择最优的匹配，从而分类。但是文本也有自己的特点，根据文本的特点，文本分类的一般流程为：1. 预处理；2. 文本表示及特征选择；3. 构造分类器；4. 分类。通常来讲，文本分类任务是指在给定的分类体系中，将文本指定分到某个或某几个类别中。被分类的对象有短文本，例如句子、标题、商品评论等等，长文本，如文章等。

### 2.4 Attention

Attention 机制由 Bengio 团队与 2014 年提出并在近年广泛的应用在深度学习中的各个领域，例如在计算机视觉方向用于捕捉图像上的感受野，或者 NLP 中用于定位关键 token 或者特征。

Attention 机制模仿了生物观察行为的内部过程，即一种将内部经验和外部感觉对齐从而增加部分区域的观察精细度的机制。例如人的视觉在处理一张图片时，会通过快速扫描全局图像，获得需要重点关注的目标区域，也就是注意力焦点。然后对这一区域投入更多的注意力资源，以获得更多所需要关注的目标的细节信息，并抑制其它无用信息。

### 2.5 Transformer

谷歌团队近期提出的用于生成词向量的 BERT 算法在 NLP 的 11 项任务中取得了效果的大幅提升，堪称 2018 年深度学习领域最振奋人心的消息，而 BERT 算法中的最重要的部分便是 Transformer 概念。

Transformer 中抛弃了传统的 CNN 和 RNN，整个网络结构完全是由 Attention 机制组成。更准确地讲，Transformer 由且仅由 self-Attenion 和 Feed Forward Neural Network 组成。一个基于 Transformer 的可训练的神经网络可以通过堆叠 Transformer 的形式进行搭建，实验通过搭建编码器和解码器各 6 层，总共 12 层 Encoder-Decoder，并在机器翻译中取得了 BLEU 值的新高。

### 2.6 Transformers API

State-of-the-art Natural Language Processing for Pytorch and TensorFlow 2.0.

Transformers (formerly known as pytorch-transformers and pytorch-pretrained-bert) provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet…) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

# 3 DATA

## 3.1 Original Datasets

我们的数据集来源是题目中贝壳找房官方给定的数据集，数据内容来自 IM 聊天交流片段，片段包含一个客户问题以及随后的经纪人若干 IM 消息。

训练集：客户问题文件和经纪人回复两个文件，涉及 6000 段对话（有标签答案），每段对话带有一条客户问题，和若干条经纪人回复，每个问题对应的回复绝大多数为 1-2 条，6000 段对话总计 21585 条回复。客户问题文件数据格式如下：

| Field | Type | Note |
|---|---|---|
| 问题 id | Int | |
| 客户问题 | String | 同一对话 Id 只有一个问题 |

- **数据示例**

| 对话 id | 客户问题 |
|---|---|
| 1 | 您好，请问这个户型有什么优缺点呢? |

经纪人回复文件数据格式如下：

| Field | Type | Note |
|---|---|---|
| 对话 id | Int | 对应客户问题文件中的对话 id |
| 经纪人回复 id | Int | id 对应真实回复顺序 |
| 经纪人回复内容 | String | 已脱敏 |
| 经纪人回复标签 | Int | 1 表示此回复是问题回答，0 相反 |

数据例子:

| 对话 id | 回复 id | 回复内容 | 回复标签 |
|---|---|---|---|
| 1 | 1 | 你是想看看这套房子是吗 | 0 |
| 1 | 2 | 在的 | 0 |
| 1 | 3 | 此房房型方正得房率高 | 1 |

测试集，客户问题文件和经纪人回复两个文件，涉及 14000 段对话（无标签答案），其中客户问题文件数据格式同训练集，而经纪人回复文件数据格式与训练集相同，只是去除了“经纪人回复标签”列；每个问题对应的回复绝大多数为 1-2 条，14000 段对话总计 53757 条回复。

## 3.2 Data Analysis

获得数据集后我们首先做的是针对数据集的数据分析。由于官方所给的 IM 交流数据集中的客户问题和经纪人回复分别在两个不同的文件中，为了能更清楚地了解对话交流的具体信息，我们先将分散在两个不同文件中的问题和回答依据其相同的对话 id 聚合在一块。**Tabel 1** 展现了问题和回复聚合之后的结果，在后面的实验过程中也将看到，将问题和回答聚合到一个文件中也是将数据传入神经网络训练前必须步骤。

训练集数据聚合后，我们开始对数据集整体开始分析。首先分析了问题和回答的数目的，去重之后得到的结果为，**问题数：4230, 答案数： 16838**，原始的 6000 段对话中的 6000 条客户问题经过去重之后变为 4230 条，而原始的 21584 行回答，实际上也只有 16838 条；这个结果说明训练集中的有近 1/4 的数据为重复数据。

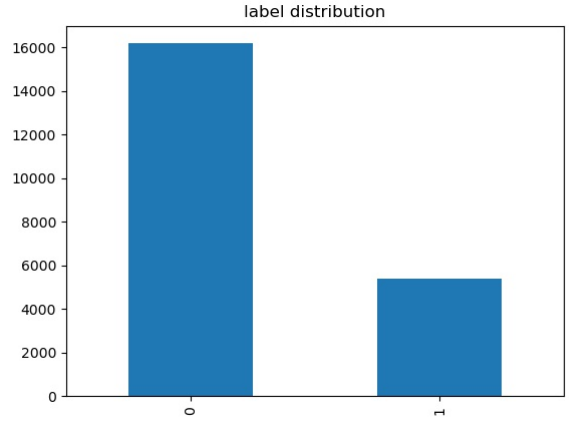


**Figure 1: Label distribution**

后续我们对训练集中的经纪人回复标签进行了统计，**Figure 1** 发现，大多数回答的标签都为 0，也即非针对客户问题的回答占据了训练集的绝大部分。

**Table 2: Train query length distribution**

| type | length |
|---|---|
| mean | 9.150806 |
| std | 5.045822 |
| min | 2.000000 |
| 25% | 6.000000 |
| 50% | 8.000000 |
| 75% | 12.000000 |
| max | 65.000000 |

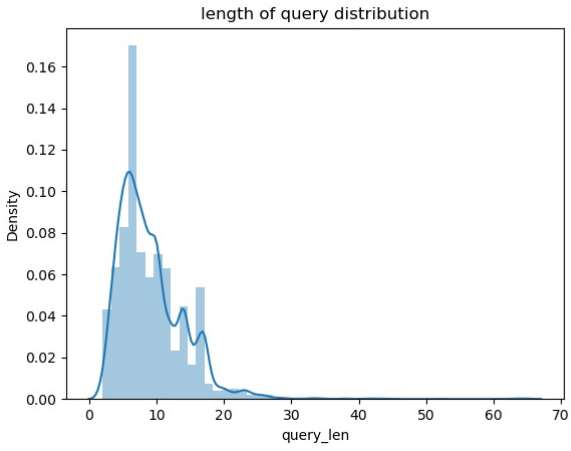


**Figure 2: Train query length distribution**

**Table 1: 聚合示例（训练集）**

| id | query | reply_id | reply | label |
|---|---|---|---|---|
| 0 | 采荷一小是分校吧 | 0 | 杭州市采荷第一小学钱江苑校区，杭州市钱江新城实验学校。 | 1 |
| 0 | 采荷一小是分校吧 | 1 | 是的 | 0 |
| 0 | 采荷一小是分校吧 | 2 | 这是 5 楼 | 0 |
| 1 | 毛坯吗? | 0 | 因为公积金贷款贷的少 | 0 |
| 1 | 毛坯吗? | 1 | 是呢 | 0 |

**Table 3: Train reply length distribution**

| type | length |
|---|---|
| mean | 10.410350 |
| std | 10.537412 |
| min | 1.000000 |
| 25% | 4.000000 |
| 50% | 8.000000 |
| 75% | 13.000000 |
| max | 200.000000 |

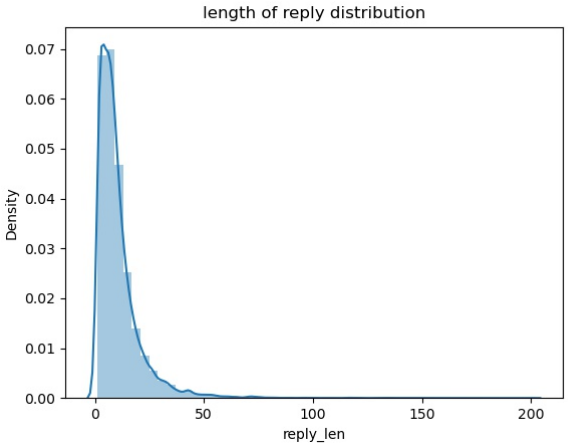


**Figure 3: Train reply length distribution**

随后我们对训练集中的 query 和 reply 分别统计了文本长度。从 **Table 2** 结果可以看到，训练集客户问题的文本长度平均值大约为 9，最短问题的长度为 2，最长的问题长度为 65，50% 的问题长度小于 8，绝大部分问题文本长度小于 12，集中在 7 左右。**Figure 2** 为其长度统计的可视化；从 **Table 3**，训练集经纪人的回复文本长度平均值大约为 10，最短回复的长度为 1，最长的回复长度为 200，50% 的回复长度小于 8，绝大部分回复的文本长度小于 13, 集中在 10 左右，**Figure 3** 为其长度统计的可视化。

**Table 4: Customer query segmentation in training data**

| query | query segmentation |
|---|---|
| 采荷一小是分校吧 | 采荷 一小 是 分校 吧 |
| 毛坯吗? | 毛坯 吗 ? |
| 靠近川沙路嘛? | 靠近 川沙 路 嘛 ? |
| 这个税多少钱 | 这个 税 多少 钱 |
| 有房子可以带我看看吗? | 有 房子 可以 带 我 看看 吗 ? |

**Table 5: Train query segmentation length distribution**

| type | length |
|---|---|
| mean | 5.721460 |
| std | 3.106128 |
| min | 1.000000 |
| 25% | 4.000000 |
| 50% | 5.000000 |
| 75% | 7.000000 |
| max | 40.000000 |

由于在自然语言处理问题中对文本数据并不是将整段话一块处理的，需要将一段话进行分词，再将词条转化成词向量进行训练，这里利用中文分词包 jieba 进行对问答语句文本进行分词处理，**Table 4**、**6** 分别为训练集文本分词后结果。

**Table 6: Agent answer segmentation in training data**

| reply | reply segmentation |
|---|---|
| 因为公积金贷款贷的少 | 因为 公积金 贷款 贷 的 少 |
| 这是 5 楼 | 这是 5 楼 |
| 房本都是五年外的? | 房本 都 是 五年 外 的 |
| 您是首套还是二套呢? | 您 是 首套 还是 二套 呢 ? |
| 所有费用下来 654 万 | 所有 费用 下来 654 万 |

**Table 7: Train reply segmentation length distribution**

| type | length |
|---|---|
| mean | 6.600954 |
| std | 6.524305 |
| min | 1.000000 |
| 25% | 3.000000 |
| 50% | 5.000000 |
| 75% | 8.000000 |
| max | 122.000000 |

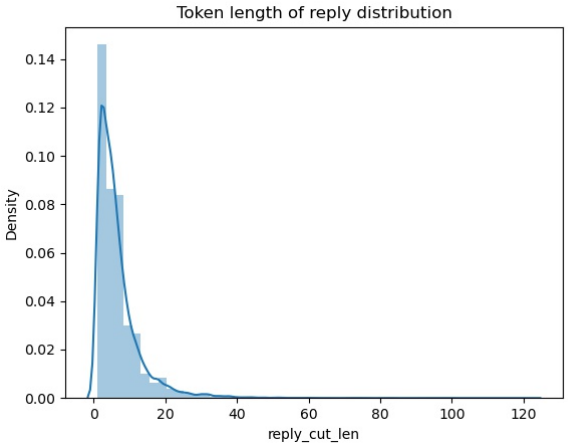


**Figure 4: Train reply segmentation length distribution**

随后我们对训练集每条问答语句的词数进行了统计。从 **Table 5** 可以看到，训练集客户问题文本的词数的平均值大约为 5，词数最少为 1，最大词数为 40，50% 的分词数小于 5，绝大部分问题文本的词数小于 7, 从输出结果来看，问题文本词数集中在 5 左右，统计结果可视化如 **Figure 4**; 如 **Tabel 7**，经纪人回复文本的词数的平均值大约为 6，词数最少为 1，最大词数为 122，是问题词数的三倍多，50% 的分词数小于 5，绝大部分问题文本的词数小于 8, 从输出结果来看，问题文本词数集中在 4 左右，统计结果可视化如 **Figure 5**。

**Table 8: Test query length distribution**

| type | length |
|---|---|
| mean | 9.127928 |
| std | 6.069857 |
| min | 2.000000 |
| 25% | 5.000000 |
| 50% | 8.000000 |
| 75% | 12.000000 |
| max | 231.000000 |

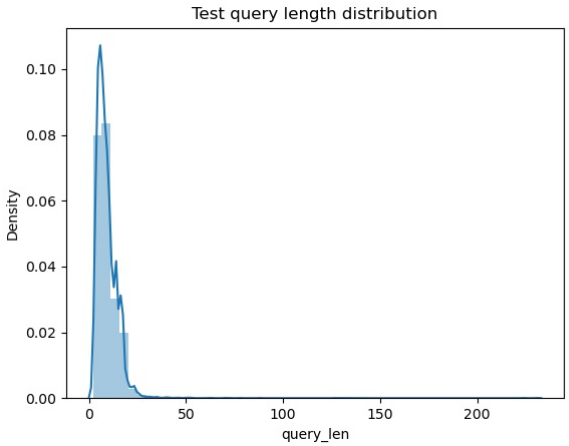


**Figure 5: Test query length distribution**

**Table 9: Test reply length distribution**

| type | length |
|---|---|
| mean | 10.694179 |
| std | 10.698465 |
| min | 1.000000 |
| 25% | 4.000000 |
| 50% | 8.000000 |
| 75% | 13.000000 |
| max | 229.000000 |

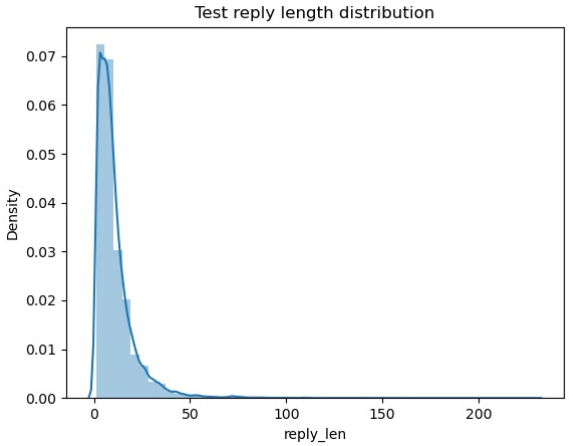


**Figure 6: Test reply length distribution**

我们对测试集也进行了相同的处理与数据分析。如**Table 8**，测试集客户问题的文本长度平均值也约为 9，最短问题的长度为 2，最长的问题长度达到 231，50% 的问题长度小于 8，绝大

部分问题文本长度小于 12，从统计结果的可视化 **Figure 6** 可以看到，文本长度集中在 7 左右。如 **Tabel 9** 测试集经纪人回复的文本长度平均值大约为 10，最短回复的长度为 1，最长的回复长度为 229，50% 的回复长度小于 8，绝大部分回复的文本长度小于 13, 也集中在 7 左右，文本长度统计结果如 **Figure 7** 。**从分析结果来看，训练集和测试集的数据无论是客户问题文本还是回复文本都很相似，文本的平均长度、数据分布、长度标准差都极其近似，这也很好的说明了人们日常生活中问答的语言习惯都是类似的**。

# 4 APPROACHES

## 4.1 Data Preprocess

通过观察训练集我们发现，文本数据的query中有部分数据包含网址 URL 链接和文本语言中不常见的表情和符号，这些内容属于处理分析问答数据中的无关噪声，我们删除了query数据中的这些内容。观察reply中的数据，除了包含上述query中已经存在的问题，还存在回复内容 String 本身为空的情况，同时当下网络聊天中运用火热的" 小表情" 成为了许多 reply 回复的主要内容，这些表情无法为我们任务的标签预测产生积极的作用，为了消除其成为噪声的可能，我们也针对回复中包含的这些表情做了清洗。然后我们将清洗后内容为空的回复和本身就为空的回复一并删除，对数据集进行了有效的精简。除此之外，我们还做了其他一些清洗工作，包括去除 html 标签、去除冗余字符、去除括号补充内容等。针对测试集的清洗工作大体类似训练集。

- **清洗实例**

训练集第 3737 条对话：
**query：**
您好，我想了解一下这套房子 https://shangye.ke.com/tj/sp/buy-/detail/1100245626
**经过清洗之后：**
您好，我想了解一下这套房子

训练集第 70 条对话 id 为 0 的的回复：
**reply：**
您再确定一下是不是看错了 (^_^)
**经过清洗之后：**
您再确定一下是不是看错了

## 4.2 Data Augmentation

由于本身的训练集样本很少（只有 6000 条对话），而测试集有多达 14000 条对话，数据量远多于训练集。训练样本少，会导致网络模型的容易出现过拟合现象，模型的泛化能力差。数据增强是扩充数据样本规模的一种有效地方法，它可以利用有限的标注数据，获取到更多的训练数据，从而减少网络中的过拟合现象，训练出泛化能力更强的模型。
基于以上理论，我们利用数据增强从有限的标注数据，获取到了更多的带标注训练数据。对本次赛题，我们尝试了传统文本数据增强技术——加噪和回译。加噪即为在原数据的基础上通过替换词、删除词等方式创造和原数据相类似的新数据。回译则是将原有数据翻译为其他语言再翻译回原语言，由于语言逻辑顺序等的不同，回译的方法也往往能够得到和原数据差别较大的新数据。增加这类噪声数据，理论上可以提升网络模型的鲁棒性。

**EDA 加噪** 这种方法出自 ICLR 2019 workshop 论文《EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks》，我们知道中文文本的语序不是那么重要，如果存在部分词语顺序错误、错别字，人类也能理解这句话是什么意思，在分类、阅读理解、对话系统、检索等大多数自然语言处理领域，这些噪声是可容忍以的。论文中介绍了如下几种增强方式：

- 同义词替换 (Synonyms Replace)：不考虑 stopwords，在句子中随机抽取 n 个词，然后从同义词词典中随机抽取同义词，并进行替换。
- 随机插入 (Randomly Insert)：不考虑 stopwords，随机抽取一个词，然后在该词的同义词集合中随机选择一个，插入原句子中的随机位置。该过程可以重复 n 次。
- 随机交换 (Randomly Swap)：句子中，随机选择两个词，位置交换。该过程可以重复 n 次。
- 随机删除 (Randomly Delete)：句子中的每个词，以概率 p 随机删除。
- 基于以上四种增强方法，我们对训练数据的query和reply分别进行增强，再重新将同一段对话对应的query和reply拼接起来，最终从原本 6000 条对话，21585 条样本，增强到了 18000 条对话，131718 条样本。

**文本回译** 在这个方法中，我们用机器翻译把一段中文翻译成英文，然后再翻译回中文。回译的方法不仅有类似同义词替换的能力，它还具有在保持原意的前提下增加或移除单词并重新组织句子的能力。回译往往能够增加文本数据的多样性，相比替换词来说，有时可以改变句法结构等，并保留语义信息。但是，回译的方法产生的数据依赖于翻译的质量，大多数出现的翻译结果可能并不那么准确。因此针对文本回译方法，我们没有使用翻译效果不那么好的 python translate 包和 textblob 包（仅支持少量翻译），而选择使用 Python 调用百度 API 实现自然语言的翻译。

## 4.3 Text Montage

在细致观察了训练集数据之后，我们发现，针对一个客户问题的若干个回答是有对话连贯性的。基于数据集语句的这种特性，同时考虑其完整性，我们将所有的回答中 label 为 1 的顺序拼接后再与问题拼接，组成 query-answer1-answer2... 的训练样本，该样本的 label 不变仍为 1。

## 4.4 Linear Regression

线性回归是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法，运用十分广泛。其表达形式为

$$y = wx + b \tag{2}$$

回归分析中，只包括一个自变量和一个因变量，且二者的关系可用一条直线近似表示，这种回归分析称为一元线性回归分析。如果回归分析中包括两个或两个以上的自变量，且因变量和自

变量之间是线性关系，则称为多元线性回归分析。
线性回归的应用包括：
如果目标是预测或者映射，线性回归可以用来对观测数据集的和 X 的值拟合出一个预测模型，当完成这样一个模型以后，对于一个新增的 $x$ 值，在没有给定与它相配对的 y 的情况下，可以用这个拟合过的模型预测出一个 $y$ 值；
给定一个变量 y 和一些变量 $x_1, ..., x_p$，这些变量有可能与 $y$ 相关，线性回归分析可以用来量化 $y$ 与 $x_j$ 之间相关性的强度，评估出与 $y$ 不相关的 $x_j$，并识别出哪些 $x_j$ 的子集包含了关于 $y$ 的冗余信息。

## 4.5 BERT

BERT 的全称是 Bidirectional Encoder Representation from Transformers，BERT 是基于 Vaswani et al(2017) 的论文"Attention is all you need" 中提出的 Transformer 模型构建的多层双向 transformer encoder。该模型的主要创新点都在 pre-train 方法上，即用了 Masked Language Model 和 Next Sentence Prediction 两种方法分别捕捉词语和句子级别的 representation，进一步增加词向量模型泛化能力，充分描述字符级、词级、句子级甚至句间关系特征。

**Embedding** BERT 中有三种 Embedding，即 Token Embedding, Segment Embedding,Position Embedding：

- `Token Embedding`: WordPiece tokenization subword 词向量，第一个单词是 CLS 标志，可以用于之后的分类任务.
- `Segment Embedding`: 用来表明这个词属于哪个句子，因为预训练不光做 LM 还要做以两个句子为输入的分类任务.
- `Position Embedding`: 这里的位置词向量不是 Transformer 中的三角函数，而是 BERT 经过训练学习出来的.

因为 Transformer 既没有 RNN 的 Recurrence 也没有 CNN 的 Convolution，而序列顺序信息又是很重要。Transformer 使用正弦波计算 token 的位置信息，类似模拟信号传播周期性变化。这样的循环函数可以一定程度上增加模型的泛化能力。但 BERT 直接训练一个 Position Embedding 来保留位置信息，每个位置随机初始化一个向量，加入模型训练，最后就得到一个包含位置信息的 embedding，最后这个 Position Embedding 和 Word Embedding 的结合方式上，BERT 选择直接拼接。

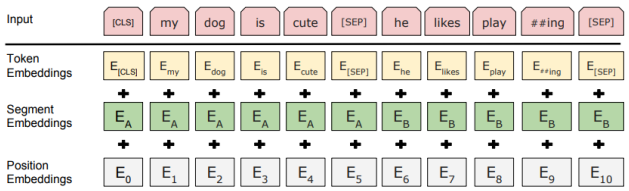


**Figure 7: Embedding of BERT**

**Masked Language Model** MLM 可以理解为完形填空，作者会随机 mask 每一个句子中 15% 的 token，然后将 masked token 位置输出的最终隐层向量送入 softmax，来预测 masked token。而对于盖住词的特殊标记，在下游 NLP 任务中不存在。

因此，为了和后续任务保持一致，作者按一定的比例在需要预测的词位置上输入原词或者输入某个随机的词。如：my dog is hairy：

- 有 80% 的概率用 "[MASK]" 标记来替换——my dog is [MASK]
- 有 10% 的概率用随机采样的一个单词来替换——my dog is apple
- 有 10% 的概率不做替换——my dog is hairy

Mask 如何做也是有技巧的，如果一直用标记 [MASK] 代替（在实际预测时是碰不到这个标记的）会影响模型，所以随机 mask 的时候 10% 的单词会被替代成其他单词，10% 的单词不替换，剩下 80% 才被替换为 [MASK]。要注意的是 Masked Language Model 预训练阶段模型是不知道真正被 mask 的是哪个词，所以模型每个词都要关注。

**Next Sentence Prediction** 选择一些句子对 A 与 B/C，其中 50% 的数据 B 是 A 的下一条句子，剩余 50% 的数据 C 是语料库中随机选择的，学习其中的相关性，添加这样的预训练目的是目前很多 NLP 的任务比如 QA 和 NLI 都需要理解两个句子之间的关系，从而能让预训练的模型更好的适应这样的任务。

- BERT 先是用 MASK 来提高视野范围的信息获取量，增加 duplicate 再随机 MASK
- 全局视野极大地降低了学习的难度，然后再用 A+B/C 来作为样本，这样每条样本都有 50% 的概率看到一半左右的噪声；
- 直接学习 Mask A+B/C 是没法进行的，因为不知道哪些是噪声，所以又加上 Next Sentence Prediction 任务，与 MLM 同时进行训练，这样用 NSP 来辅助模型对噪声/非噪声的辨识，用 MLM 来完成语义的大部分的学习。

**Why Select BERT** BERT 是截至 2018 年 10 月的最新 state of the art 模型，通过预训练和微调横扫了 11 项 NLP 任务，这首先就是最大的优点；而且它还用的是 Transformer，也就是相对 RNN 更加高效、能捕捉更长距离的依赖。对比起之前的预训练模型，它捕捉的是真正意义上的 bidirectional context 信息。展开来说：

- Transformer Encoder 因为有 Self-Attention 机制，因此 BERT 自带双向功能
- 因为双向功能以及多层 Self-Attention 机制的影响，使得 BERT 必须使用 Cloze 版的语言模型 Masked-LM 来完成 token 级别的预训练
- 为了获取比词更高级别的句子级别的语义表征，BERT 加入了 Next Sentence Prediction 来和 Masked-LM 一起做联合训练
- 为了适配多任务下的迁移学习，BERT 设计了更通用的输入层和输出层
- 微调成本小

## 4.6 Other Related Models

BERT-base 是经典的预训练模型，自它提出后，陆续有许多基于 BERT 的改进优化模型，接下来我们将简单介绍我们在解决房产行业聊天问答匹配问题过程中所尝试过的预训练模型。此外显然我们的问题是处理中文语句的，所以这里提及的所有

预训练模型都是基于中文语料库的。

**ERNIE1.0** 由百度提出，模型本身保持基于字特征输入建模，使得模型在应用时不需要依赖其他信息，具备更强的通用性和可扩展性。相对词特征输入模型，字特征可以建模字的组合语义，通过相同字的语义组合学到词之间的语义关系。此外 ERNIE 的训练语料引入了多源数据知识，提升了模型语义表示能力，这使得 ERNIE 相较于 BERT-base，在语言推断效果上更胜一筹，比如语义相似度任务、情感分析任务、命名实体识别任务以及检索式问答匹配任务。我们的问题正好属于问答匹配任务，因此选择尝试一下该模型，事实上目前已经有 ERNIE2.0，但由于我们是基于 transformers 编程的，huggingface 上没有 ERNIE2.0 的预训练模型，无法直接下载，所以后来并没有尝试 2.0 版本。

**BERT-WWM** 即 Pre-Training with WholeWord Masking for Chinese BERT，其对 BERT-base 的改进主要体现在 Mask 的方式上，使用全词 Mask，与百度的 ERNIE 相比，BERT-WWM 不仅仅是连续 Mask 实体词和短语，而是连续 Mask 所有能组成中文词语的字。具体做法是，针对中文，如果一个完整的词的部分字被 Mask，则同属于该词的其他部分也会被 Mask，即对组成同一个词的汉字全部进行 Mask，即全词 Mask；这样做的目的是在预训练过程中，模型能够学习到词的语义信息，训练完成后字的 Embedding 就具有了词的语义信息，对我们中文 NLP 任务是很有用的。

**BERT-WWM-EXT** 是 BERT-WWM 的一个升级版，主要增大了预训练数据集，次数达到 5.4B，同时增大训练步数，训练第一阶段 1M 步，训练第二阶段 400K 步。对于解决我们的实际问题，除去 RoBERTa-LARGE 预训练大模型，BERT-WWM-EXT 是在规模相似的预训练模型中单模效果最好的，具体将在 Experiments 讲述。

**RoBERTa-WWM-EXT** 相较于 BERT-WWM-EXT 最大的改进：

- 动态 Masking，相比于静态，动态 Masking 是每次输入到序列的 Masking 都不一样；
- 取消 Next Sentence Predict 任务，移除 Next Predict Loss，相比于 BERT，采用了连续的 full-sentences 和 doc-sentences 作为输入（长度最多为 512）；
- 采用更大的 batch size，batch size 更大，training step 减少，实验效果更好；
- text encoding，基于 bytes 的编码可以有效防止 unknown 问题。另外，预训练数据集从 16G 增加到了 160G，训练轮数比 BERT 有所增加

**RoBERTa-WWM-EXT-LARGE** 以上提及的模型都是 base 规模的，包含 12-layer，768-hidden，12-heads，110M parameters，这些都可以在我们本地的笔记本（显存 4G）上跑，而 RoBERTa-WWM-EXT-LARGE，“人如其名”，是 Large 规模的，包含 24-layer，1024-hidden，16-heads，330M parameters，本地笔记本显存不足无法跑动 Large 模型，在比赛尾声时，我们使用了华为云的 ModelArts 在显存 32G 的服务器上跑了一次 Large 模型，经过其他数据、模型融合处理后，得到了不俗的效果。

## 4.7 Downstream Tasks

BERT 等预训练模型的提出，简化了我们对 NLP 任务精心设计特定体系结构的需求，我们只需要在 BERT 等预训练模型之后下接一些网络结构，即可出色地完成特定任务。这是因为 BERT 通过大量预料的无监督学习，已经将语料中的知识迁移进了预训练模型的 Embedding 中，为此我们只需要在针对特定任务增加结构来进行微调，即可适应当前任务，这也是迁移学习的魔力所在。对于 BERT，可以分为四大下游任务：句子对分类任务、单句子分类任务、问答任务、单句子标注任务，根据不同的任务分别微调 BERT 预训练模型。

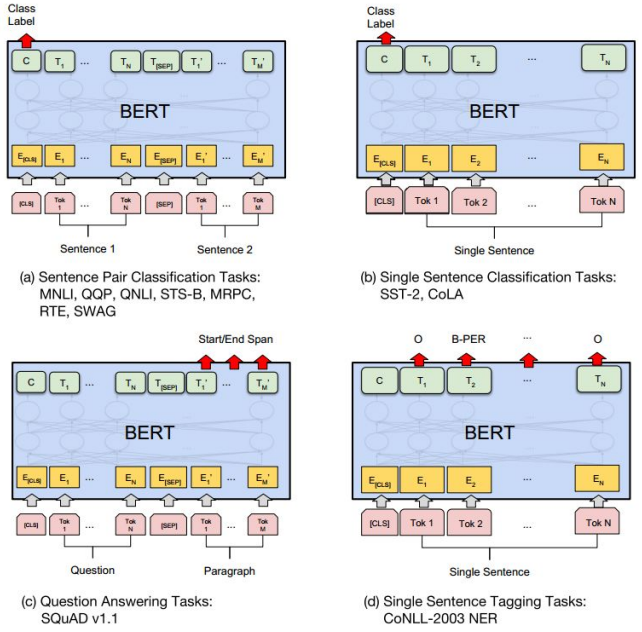


**Figure 8: NLP 四大下游任务微调结构图**

分析房产行业聊天问答匹配这个问题，可以确定它的下游任务是句子对分类任务或问答任务。

**NSP** 我们首先考虑的思路是把它当成 Next Sentence Prediction，即下一句预测，这是一个二分类问题，针对一个提问的几个回答，它们只有“是下一句”或“不是下一句”两种标签。根据第一种思路，在预训练模型后增加结构以进行下游任务的微调；在我们的新增模型中，我们调用了 transformers 的 BertForNextSentencePrediction 加载具有 NSP 任务的预训练模型，经过 bert_nsp 的前向传播就可以获取每个句子的 logits，根据 logits 得知 label。

**Fully Connected Layers Optimization** 一开始我们是直接用 BERT 做的句子分类任务，训练出来的模型评测结果并不理想，因此我们对下游任务模型进一步优化。优化的基本思路是 BERT 预训练模型最后一层所有的输出是一个 3 维张量，将输出的张量用池化操作可以降维到 2 维，从而提取关键特征，然后与句子向量拼接共同组成特征向量输入到全连接中输出最终结果；

更具体地说，首先取 BERT 最后一层的 [CLS] token 对应的向

量作为句子向量，然后取最后一层的所有 hidden_state，它的 shape 是 (batch_size,sequence_size,hidden_size)，接着对这些 hidden_state 先做最大池化操作再做均值池化操作，将句子向量与得到的最大池化向量、均值池化向量拼接，经过第一个全连接层用 tanh 作激活函数对 outputs 进行非线性变换，再经过第二个全连接层得到最终的 outputs：(batch_size, num_class)

## 4.8 Train

经过上述方法的介绍，我们可以基本确定使用的预训练模型与微调模型了。接下来就分三个部分具体讲解训练的过程：训练数据、预训练、微调训练

**Data** 我们得到经过数据预处理的训练集和测试集后，首先要将原有训练集按照 5 折交叉验证划分成新的训练集和验证集，然后利用 BertTokenizer 从预训练模型中加载 tokenizer，对输入文本进行处理，即在文本开头加上 [CLS]，在每个句子后面加上 [SEP]，编码输入到 BertModel 中才能被正确识别。但在实际解决问题中，我们还定义了一个 zyDataset 类，其作用是便于获取某个词项的编码值和标签（如果是训练集或验证集的话有标签）以及编码长度，将经过 tokenizer 编码后的文本初始化成自定义的 zyDataset 类再加载。

**Pre-Training** 在上文已经提及了许多与 BERT 相关的预训练模型，在实际解决问题中，直接使用 transformers 这个 API 提供的 BertModel.from_pretrained() 从 huggingface 加载对应的预训练模型即可。

**Fine-tuning** 在加载好的预训练模型之后接上一些新的层来处理下游任务用给定的数据集进行微调，数据集就是 Input 中处理好的数据，新的层就是我们之前定义好的优化模型。微调训练的损失函数是交叉熵损失函数，优化器是 Adam，针对不同的预训练模型规模选择合适的学习率和 batch size（batch size 不宜过小，会造成过拟合），使用 100 次迭代作热身训练有助于减缓模型在初始阶段对 mini-batch 的提前过拟合现象，保持分布的平稳；训练使用 Kfold，分 5 折交叉验证，每折 10 代训练出一个模型并预测概率，将 5 个模型预测的概率求和取平均后得到最终的 label；经过 5 折交叉验证的结果要比单折模型预测的结果的 F1 score 高得多，这其实也是一种模型融合的思想。

## 4.9 Model Ensemble

**Voting** 首先我们采用加权融合的方式对训练得到的各种模型进行融合。开始我们先尝试对所有模型赋予相同的权重，之后又将表现较好的模型的权重增大，经过多次调节权重后得到该融合方式下的最好结果。

**Stacking** Stacking 融合模型共分为两层。第一层是进行各个单模型的训练和预测，第二层则是将第一层中各个单模型的输出值作为第二层模型的输入值，输入到线性回归、GBDT 等模型进行训练得到最终的输出值。因为 Stacking 融合方式中第二层模型一般不应过于复杂，所以此处选择线性回归模型进行融合。对于第一层中各个单模型的选取，采取类似加权融合中的处理方式，先将所有模型赋予相同的权重进行输入，之后再考虑将表现比较好的模型进行复用。

**Blending** Blending 与 Stacking 大致相同，只是 Blending 的主要区别在于训练集不是通过 K-Fold 的 CV 策略来获得预测值从而生成第二阶段模型的特征，而是建立一个 Holdout 集。简单来说，Blending 直接用不相交的数据集用于不同层的训练。以本题数据为例，我们将训练集划分为两部分（d1，d2），测试集为 test。第一层：用 d1 训练多个模型，将其对 d2 和 test 的预测结果作为第二层的 New Features。第二层：用 d2 的 NewFeatures 和标签训练新的分类器，然后把 test 的 New Features 输入作为最终的测试集，对 test 预测出的结果就是最终的模型融合的值。

## 4.10 Tricks Summary

**Kfold** 对样本进行 Kfold 然后训练，得到 k 个模型再进行 ensemble，这种方式的好处是可以让更多的数据参与到训练，同时多个模型进行投票，也会带来或多或少的提升。在实际运用中，我使用的是 k=5，即 5 个单折模型预测出的概率相加取平均后预测 label，这样做得到的效果远比单折模型来得好。

**Stacking** 虽然 BERT 的特征提取能力强大，但是在 BERT 后面接一些新的层，总能带来一定的提升。这种方式可以看作是两种模型的 Stacking，即利用 BERT 做特征提取，后面的模型在其上做下游任务。

**Data Cleaning and Augmentation** 数据清洗与增强往往是提高模型泛化能力的技巧之一，仅依靠官方给的训练集去训练模型难以达到理想的成绩，因此可以对原有的数据集清洗，清洗掉脏数据后进行增强，增强的方法在之前也提到过，同义词变化、随机插入删除、回译等。

**Model Ensemble** 前面提到的 Kfold 其实就算一种模型融合，是将单模的多折模型融合起来即 5 折交叉验证对应 5 个单模融合；此外我们还尝试了将多个不同的单模基于 logits 融合，方法类似 Kfold，但效果一般；其中主要研究的是 Voting、Stacking、Blending 这三种经典的模型融合方法，其中 Stacking 效果最佳，效果对比将在 Experiments 中展开。

**Large Model** 使用大规模预训练模型，简单粗暴地提升效果，需要借助算力完成。

## 4.11 Other trials

除了上述的预训练、下游任务微调等，我们还认为其他一些预训练模型或工具是可以尝试的，但由于时间原因，还未来得及完成，或者只是作了了解。

NEZHA，Neural Contextualized Representation for Chinese Language Understanding，由华为提出的预训练模型，面向自然语言理解任务，相较于 BERT-base，在模型上 NEZHA 使用函数式相对位置编码，而在 BERT 中使用的是绝对位置编码，此外还使用了全词掩码（WWM），这是上面提及过的一个优化；在训练上，NEZHA 使用了混合精度训练并在训练过程中使用了 LAMB 优化器。但 transformers 无法下载 NEZHA 的预训练模型，如果要使用 NEZHA 需要重构我们的代码，所以暂时搁置了。

UER-py，是一个在通用语料预训练以及对下游任务进行微调的工具包。我们知道预训练模型有几个关键的部分：语料、编码器、目标任务以及微调策略。UER-py 能让用户轻易地对不同的部分进行组合，复现已有的预训练模型（比如 BERT），并为用户提供进一步扩展的接口，规范地使用可以帮助我们在一系列下游任务上取得比 Google BERT 更好的效果。针对房产行业聊天问答匹配问题，在模型训练和推理中可以选择使用 UER，因为它自带交叉验证 cross validation，支持 stacking 集成，同时给出了使用 LightGBM 和贝叶斯超参数搜索的脚本，可以非常方便地应用一些提高 F1 score 的 tricks。我们在解决问题中途发现了 UER-py，如果要使用 UER-py，我们已有的训练好的模型的格式是与它不匹配的，需要用 UER-py 重新训练模型，鉴于时间有限，只是简单地了解。

### 4.12 Evaluation

评测方式分为线上评测和线下评测，评价指标主要使用 F1 score：2*(精度 * 召回)/(精度 + 召回)，具体在 Introduction 有提及。其中，线上评测不多说了，就是 DataFountain 比赛官网的官方评测，官方评测分为 A 榜和 B 榜，A 榜只是拿了测试集的部分数据来评测，B 榜用测试集的全部数据来评测。线下评测是训练过程中进行的，即用 evaluation() 评测验证集，得知当前折当前代的模型性能，包括 val_loss、accuracy、precision、recall 和 F1 score；根据 val_loss，保存 loss 最低的模型作为 best，同时保存当前最后一个模型即 last, 观察 best 和 last 的关系 (修改时间)，可以大致掌握目前的训练是否已经过拟合。

## 5 EXPERIMENTS

### 5.1 Data Augmentation

**EDA 加噪** 从前面的数据集文本长度统计中，我们发现不同的句子长度区别很大，尤其对于 reply 文本长度标准差很大。而 EDA 的方法是一种加噪的过程，它改变了句子中词的顺序和数目，尤其对于长度较短的句子，这种操作有很大可能会让原句面目全非，改变句子原意。长句子相对于短句子，存在一个特性：长句比短句有更多的词，在进行 EDA 增强的过程中能吸收更多的噪声，即使改变部分词序和数目人们也能基本理解句意。因此在增强的过程中，需要基于文本长度来决定不同文本中需要改变的单词数。对于不同的句长，因 EDA 增强而改变的单词数可能不同，句子越长的改变的词就越多。具体实现：对于 SR、RI、RS，遵循公式：$n = \alpha * l$，$l$ 表示句长，$\alpha$ 表示一个句子中需要改变的单词数的比例。在 RD 中，让 $p$ 和 $\alpha$ 相等。

EDA 方法中还有两个参数需要根据具体实例自行确定，一个是 num_aug 参数：每一条语料将增强的个数, 另一个是 alpha 参数：每一条语料中改动的词所占的比例。针对我们的原始 6000 条对话，21584 条回复，按照论文中的建议，alpha 参数应设为 0.1，num_aug 参数相应的设为 4。而由于这道题目的文本数据为一条 query 对应若干条 reply，因此我需要先将 query 的增强数目固定。起初我们走进了数据增强的越多，模型泛化能力就越强的误区。一开始设置的为 5 条，由分析可知对于整个训练集，query 每增加一条，相应的训练条目就会增加该 query 对应的若干 reply 总共的增强条数。后面由于增强后的训练集太过庞大达到了 75w 行数据。当我们训练了一个晚上之后，初步估计该数据集训练完至少需要一周的时间。基于训练时间的权衡，我们不得不将 query 增强的条数减小到 3。

同时我们又发现，对于 reply 句子的增强，当 num_aug 设为 4 时，每一种增强方法会执行 $int(num\_aug/4) + 1 = 2$，在最后的增强序列中会产生 8 条新的句子。对于短句，虽然改变的词数 $\alpha * l$ 一般为 0，也即不改变句子，但仍然会增强 4 条相同的句子到增强结果中，例如回答中常见的“好的”，“是的”，“嗯”这类表肯定的短句语气词，重复的这种回复是无意义的，并没有对数据集进行有效的扩充，还会给数据带来冗余。因此在原始方法的基础上，我只在每一条语句增强的经过去重的结果，挑选其中的若干条，加入训练集。实验中挑选的条数为 $n = max(1, int(num\_words/3))$，仍然采用前面的思想，增强语句数目由原始句长决定，对于长度越长的 reply，选取的增强语句数目越多。

当挑选的条数计算结果为 1 时，我们仅将原始的句子重新写入训练集，并不会再增加新的句子。当条数计算结果大于 1 时，我们先将原句子加入训练集，随后在增强结果中随机选取剩下的 n-1 条。这种增强策略下，最终产生了 18000 条对话，131718 条回复样本。

- **增强实例**

1. 训练集第 31 条对话

| query | augmented |
|---|---|
| 什么时候能够看房呢? | 时候什么能够看房呢? |
| | 什么能够呢? |
| | 什么什么时候能够看房呢? |

| reply | augmented |
|---|---|
| 您好这套房子随时可以看 | 您好可以房子随时这套看 |
| | 您好看房子随时可以这套 |
| | 您好房子随时可以看 |
| 您您什么时候有时间看房? | 您您什么时候有看房 |
| | 什么有时间时候看房? |
| | 时候您看房您有时间什么? |

2. 训练集第 65 条对话

| query | augmented |
|---|---|
| 那前面光线怎么样 | 那右边光线怎么样 |
| | 那光线怎么样 |
| | 怎么样前面光线那 |

| reply | augmented |
|---|---|
| 不影响啊 | 不影响啊 |
| 就是看顶楼你介意不 | 介意看顶楼你就是不 |
| | 看你不 |
| | 看就是看顶楼你介意不 |
| 漏雨的话会跟您说的 | 漏雨跟您会的话说的 |
| | 漏雨会跟您 |
| | 漏雨的话会跟您能够说的 |

**文本回译** 对于文本回译，在网络查询对比各种翻译 API 之后，

我选择了评价较高的百度 API 进行文本回译。在获取了 APP ID 和个人密匙后进行回译，我们先将整个训练集文本翻译成英文返回，再重新将回译后的中文文本写回训练集，最后我们在训练集中重新加上了原始数据。

- **回译实例**

例如：训练集第 61 条对话
**query:**
original: 我想了解一下这里的房子
translated: 我想知道这里的房子

**reply:**
回复 id 0: 您是考虑买还是卖?
translated: 你在考虑买还是卖?

回复 id 1: 之前您有没有过来看过这个小区呢
translated: 你以前去过这个社区吗
回复 id 2: 有 153-166-178 平的
translated: 是 153-166-178 平的
回复 id 3: 我给您发几套，您看一下
translated: 我给你寄几套。看一看

## 5.2 Text Montage

在进行文本拼接的过程中，会遇到 reply 标签有多个为 1 的情况，这种情况下，就需要考虑拼接的顺序和一次拼接的 reply 条数。假设针对一个问题的 reply，标签为 1 的有 n 条，我们遵循回答的先后顺序，先将其两两连接，得到 n-1 条拼接的 reply，随后再顺序拼接三条 reply 得到 n-2 条新的 reply...，直达所有 label 为 1 的 reply 拼接成一条完整 reply。最后将拼接得到的 reply 与原始若干条 reply 一同组成新的训练集回复文本。这种方法得到的对话数仍为 6000 条，回复样本增加到了 23325 条。

- **拼接实例**

1、训练集第 2 条对话的 reply，如 Table 10、11

**Table 10: Original reply list of Dialogue 2 in training data**

| id | reply_id | reply | label |
|---|---|---|---|
| 2 | 0 | 您是首套还是二套呢? | 0 |
| 2 | 1 | 所有费用下来 654 万 | 1 |
| 2 | 2 | 包含着税费和我们的服务费和房款 | 1 |
| 2 | 3 | 好的 | 0 |
| 2 | 4 | 链家天鸿美域店 NAME，电话是 PHONE（同微信号），随时联系? | 0 |

2、训练集第 25 条对话的 reply，如 Table 12、13

**Table 11: Spliced reply list of Dialogue 2 in training data**

| id | reply_id | reply | label |
|---|---|---|---|
| 2 | 0 | 您是首套还是二套呢? | 0 |
| 2 | 1 | 所有费用下来 654 万 | 1 |
| 2 | 2 | 包含着税费和我们的服务费和房款 | 1 |
| 2 | 3 | 好的 | 0 |
| 2 | 4 | 链家天鸿美域店 NAME，电话是 PHONE（同微信号），随时联系? | 0 |
| 2 | 5 | 所有费用下来 654 万, 包含着税费和我们的服务费和房款 | 1 |

**Table 12: Original reply list of Dialogue 25 in training data**

| id | reply_id | reply | label |
|---|---|---|---|
| 25 | 0 | 链家 NAME 很高兴为您服务 | 0 |
| 25 | 1 | 永丰楼 6 楼到顶的 5 楼 | 1 |
| 25 | 2 | 南北通透的两室 | 1 |
| 25 | 3 | 私产 满五年 | 1 |
| 25 | 4 | 性价比挺合适的 | 1 |

**Table 13: Spliced reply list of Dialogue 25 in training data**

| id | reply_id | reply | label |
|---|---|---|---|
| 25 | 0 | 链家 NAME 很高兴为您服务 | 0 |
| 25 | 1 | 永丰楼 6 楼到顶的 5 楼 | 1 |
| 25 | 2 | 南北通透的两室 | 1 |
| 25 | 3 | 私产 满五年 | 1 |
| 25 | 4 | 性价比挺合适的 | 1 |
| 25 | 5 | 南北通透的两室，私产 满五年 | 1 |
| 25 | 6 | 私产 满五年，性价比挺合适的 | 1 |
| 25 | 7 | 永丰楼 6 楼到顶的 5 楼，南北通透的两室 | 1 |
| 25 | 8 | 南北通透的两室，私产 满五年，性价比挺合适的 | 1 |
| 25 | 9 | 永丰楼 6 楼到顶的 5 楼，南北通透的两室，私产 满五年 | 1 |
| 25 | 10 | 永丰楼 6 楼到顶的 5 楼，南北通透的两室，私产 满五年, 性价比挺合适的 | 1 |

从实例中我们可以看到，对于第 2 段对话的 reply，label 为 1 的 reply 只有两条，因此仅产生一条拼接结果；而对于第 25 段对话的 reply，有 4 条 label 为 1 的 reply，会产生表中 reply_id 为 5,6,7 的由两条原始 reply 顺序拼接而成，reply_id 为 8,9 的由三条原始回复顺序拼接而成，reply_id 为 10 的由四条原始回复顺序拼接而成的新 reply。

## 5.3 Comparisons

我们将经上述三种方法处理后的数据集单独进行训练，以比较不同数据处理方法对本题效果的影响。训练所采用的模型为

在初始数据集上表现最好的 bert-wwm-ext 模型。线上评测结果如 Table 14.

**Table 14: 不同数据增强方法下的线上 F1-score**

| 增强类型 | F1-score |
|---|---|
| 无增强 | 0.77319309601 |
| EDA 加噪 | 0.76951921789 |
| 文本回译 | 0.77611940299 |
| 文本拼接 | 0.77538294700 |

针对 EDA 加噪，该方法评测得分相较未处理的原数据有明显降低，我们初步分析，该方法下产生大量新的训练数据，而在增强的过程中，EDA 方法没有文本关键词的侧重，在随机删除和插入过程中很可能使原始数据丧失语义结构和语义顺序，改变了句子的意思，但其仍保留原始的类别标签，从而有可能产生了标签错误的句子，为训练数据带来的较多意外噪音，而这种噪音并不能使模型提高对测试集的泛化能力。针对文本回译，即使存在改变句法结构的现象，但基本保留了原句的语义信息，增强结果的类别标签并不存在错误，增加文本数据的多样性的同时并没有产生更多噪音，因而显著提高了模型在测试集上的泛化能力。针对文本连接，其并没有增加文本数据的多样性，训练数据自始至终都来自原始数据，但由于考虑了对话的连贯性，增加了训练文本长度，因而模型对测试集中长度较长的回复文本有更好的拟合效果，使得模型在测试集上的泛化能力也有不小提升。

## 5.4 Linear Regression

在进行深度学习模型的尝试之前，我们先进行了机器学习中线性回归模型的尝试。

**数据预处理** 对源数据的问答语句进行分词处理，并去除停词表中的无用词。收集文档中所有单词，并计算每条语句中的 TF-IDF 值，对于每条问答语句形成一个文档中包含所有词的 TF-IDF 的向量，由所有语句对应的向量组成 TF-IDF 矩阵，提供给之后模型训练使用。这里的 TF-IDF 用以根据字词的在文本中出现的次数和在整个语料中出现的文档频率来计算一个字词在整个语料中的重要程度。

**模型选择** 我们首先基于 TensorFlow 构建了线性回归模型

$$y = wx + b \tag{3}$$

以问答语句的 TF-IDF 矩阵作为线性回归式中的 x，并将标签进行 one-hot 编码后作为 y，这样根据问答语句中 TF-IDF 的相似性判断是否匹配，训练模型得到参数 w 和 b。然后使用训练得到的参数进行预测，但线上分数为 0。思考后发现这种设计方式当应用于本题时存在一定的问题。这种设计方式更适合于标签为多分类的情况，比如判断某条新闻语句属于哪种类型的新闻，比如将“周五,A 股三大指数全天弱势盘整, 沪指再度失守 3400 点关口”归类为“金融”类别，“国足世界杯前大利好！李铁助攻 64 岁陈戌源破门”归类为“体育”类型等。

于是我们对模型的变量设计进行调整。新模型将客户提问语句的 TF-IDF 矩阵作为 x，将经纪人回复语句的 TF-IDF 矩阵作为 y，并将标签为 0 和 1 的数据集分开训练，得到两个不同的模型，分别代表倾向于将输入语句分类为 0 或 1 的模型。然后通过训练集上的数据训练得到两套参数 w 和 b，并根据训练得到的参数进行预测，然后在训练得到的 y 与实际值 y* 之间计算 MAE，比较两套模型的 MAE，取较高者作为标签的分类。线性回归模型的最终得分如下

**Table 15: 线性回归最终得分**

| Method | F1-score |
|---|---|
| Linear Regression | 0.58327686678 |

## 5.5 Fully Connected Layers Optimization

这一部分的实验，我们比较在 APPROACHES 中提及的两种下游任务模型：用 BERT 直接做句子分类任务（后面简单称之为 NSP）、用 BERT 做特征提取再后接新的层处理下游任务（后面简单称之为 Bert_Fc）。这里使用的是原数据集即未做增强，使用的学习率都是 2e-5。

**Table 16: NSP 和 Bert_Fc 的线上评测效果**

| Model | F1 score |
|---|---|
| NSP,bert-base, 单折 | 0.00000000000 |
| NSP,bert-base,KFold | 0.74471760797 |
| NSP,bert-wwm,KFold | 0.75228070175 |
| Bert_Fc,bert-base, 单折 | 0.00000000000 |
| Bert_Fc,bert-base,KFold | 0.76679410158 |
| Bert_Fc,bert-wwm,KFold | 0.76592816843 |

Table 16 分别测试了 NSP 和 Bert_Fc 在 bert-base 和 bert-wwm 上的线上评测效果，不难看出整体上 Bert_Fc 比 NSP 效果要好，在房产行业聊天问答匹配这个问题中，Bert_Fc 这个后接的新层要比直接使用 NSP 做句子分类任务更好地解决了这个下游任务，这说明我们的优化是有效的！

## 5.6 Single Models

这一部分我们只讨论 KFold 下使用 Bert_Fc 后接层的单模效果，因为单折单模在线上评测的效果非常差,（从上面的表格中也可以看出，线上评测结果都是 0 意味着把所有 label 都判断成了 0 或都判断成了 1，因此基本可以忽略不计；而 Bert_Fc 比 NSP 效果好，所以 NSP 就不过多尝试了），比较基于不同的预训练模型后接下游任务的优化模型经过微调训练后的模型评测结果，预训练模型包括 bert-base-chinese,chinese-bert-wwm, chinese-bert-wwm-ext,chinese-roberta-wwm-ext,ernie-1.0,chinese-roberta-wwm-ext-large。

Table 17 比较了基于不同的预训练模型的单模 KFold 的线上评测效果。由于时间有限算力有限，Large 模型直接用回译的数据集跑了，没有用原数据集试过，所以比较起来可能没有那么客观，但是理论上 Large 模型肯定是要比 base 模型好的，毕竟规模放在那里。除此之外，这里的结果都是其他模型用原数据集跑的。Large 模型微调训练的学习率设置的是 1e-6，而其他规模的预训练模型微调训练的学习率经过调整在 2e-5 效果较好；但其中 ernie-1.0 比较特殊，一开始沿用 2e-5 的学习率

**Table 17: 单模 KFold 的线上评测效果**

| Model | F1 score |
|---|---|
| bert-base-chinese | 0.76679410158 |
| chinese-bert-wwm | 0.76592816843 |
| chinese-bert-wwm-ext | 0.77319309601 |
| chinese-roberta-wwm-ext | 0.75197680548 |
| ernie-1.0 | 0.75109293471 |
| chinese-roberta-wwm-ext-large,backTrans | 0.77959879376 |

去训练它耗时很长，收敛速度很慢，一晚上一折都没跑完，经过几次调整，在学习率为 1e-6 时线上评测勉强上 0.75，是几个模型中效果最差的，理论上感觉不会那么差，可能还是超参没有调好的缘故，由于它可能并未完全收敛，在后面模型融合中没有考虑加入 ernie。

在几个 base 规模的预训练模型中，chinese-bert-wwm-ext 线上评测分数最高，在后面的模型融合中会给予它更高的权重，而 chinese-roberta-wwm-ext 作为 chinese-bert-wwm-ext 的改进版，出乎意料，线上评测分数倒数。不过，我们之后比较了除 ernie、large 外其他 4 个模型对测试集预测的结果 csv 文件，对比很多对问答，发现在其他 3 个 bert 相关的模型都判断某个问答对为 0 时，roberta 经常会出现不同于其他三个的判断，而从我们人工的角度来判断，结果常常是 roberta 答对了，所以 chinese-roberta-wwm-ext 虽然线上评测效果不佳，但往往会在一些特定问答对中表现得特立独行，而这种特立独行又很大概率是对的，Table 18 列出了一些比较有代表性的例子，例子中其他三个模型的预测 label 是 0，而 RoBERTa 的预测 label 是 1。因此在后面的模型融合中我们并没有将其摒弃，也给予它了相当一部分权重。

**Table 18: 一些体现 RoBERTa 特立独行的问答对例子**

| Query | Reply |
|---|---|
| 对这套房子很感兴趣 | 明天有时间来看看吗 |
| 多少钱 | 33 平总价 54 万左右 |
| 你现在方便带我看一下这个房子吗 | 嗯嗯好的 |
| 问下对方付款方式怎么样的 | 贷款 115 万！其余给现金 |
| 那这套的价格在 1.6 万？ | 嗯。 |

## 5.7 Model Ensemble

**KFold** K 折交叉验证，是提升模型性能的一个 trick，在实际使用中其实也属于模型融合，因为经过 K 折会得到 K 个模型，我们将 K 个模型预测的 logits 求和取平均再判断 label，也就是将这 K 个模型融合了，只不过这 K 个模型都是基于相同的预训练模型。这里的对比实验可以直接看 Table 1，基于 bert-base 的单模单折线上评测结果都是 0，虽然结果很奇怪，但我测了两次都是 0，基本可以确认单模单折的效果极差，把 label 都判断成了 0 或 1，而 KFold 具有很显然的提升，所以我们之后的训练都是基于 KFold 的，KFold 对于这个问题必不可少。

**基于 logits 的多模融合** 如果说 KFold 是单个模型的多折融合，那么这里就可以说是多个模型的单折融合，之所以说是单折，是因为我们只是将每个单模 5 折交叉验证训练中的 best 模型拿出来预测，将他们的 logits 求和取平均判断 label，这种方法其实我们本身就没有抱很大希望，因为之前已经得知单模单折的线上评测相当差，如果将几个单模单折的结果融合起来，效果未必会比 KFold 好。但这种思路是有启发意义的，因为当然可以将多模多折融合，只是我们一开始并没有想到这个思路，没有保存单模多折的所有模型，所以如果要应用这个方法，只好从头开始训练每个模型多折累加它们的 logits，显然时间、算力不足，所以这里并没有实践。

**Table 19: 基于 logits 的多模融合的效果对比**

| Model | F1 score |
|---|---|
| bert-base-chinese | 0.76679410158 |
| chinese-bert-wwm | 0.76592816843 |
| chinese-bert-wwm-ext | 0.77319309601 |
| chinese-roberta-wwm-ext | 0.75197680548 |
| 4model,logits | 0.75228070175 |
| 4model,vote | 0.779 |

Table 19 比较了基于 logits 的多模 best 融合与基于大多数投票的多模融合以及单模的效果，可以看出基于大多数投票的多模融合是线上评测最好的，而基于 logits 的多模 best 融合效果还没有单模 KFold 来得好，这在我们意料之中，因为基于大多数投票的多模融合融合的是每个单模多折的结果文件，每个结果文件都有它对应模型多折的贡献，算是一种多模多折融合的转换形式，两者的差距也就不言而喻了；而效果不如 KFold，也说明了多模单折的融合不如单模多折融合。

**Voting,Stacking,Blending** 我们使用 Voting、Stacking，Blending 三种方法对之前得到单模进行模型融合，如 APPROACHES 中所说，我们选择的单模都是线上评测效果不错或有特殊之处的模型，经过多次组合比较，我们选择了四个较为合适的单模：Bert-base、Bert-wwm、Bert-wwm-ext+ 文本拼接数据集、RoBERTa-wwm-ext。融合的最优结果如下图所示

**Table 20: 三种模型融合方法最优结果比较**

| 融合方法 | 分数 |
|---|---|
| Voting | 0.77997886 |
| Stacking | 0.78013245 |
| Blending | 0.77932775 |

Stacking 效果最好，线上分数可达 0.78，Voting 次之，Blending 效果最差。

## 5.8 Greatest Of All Time

Greatest Of All Time，也就是我们的队名 GOAT，通常用于体育届比喻极具影响力的球员，比如 Michael Jordan, Lebron James, Lionel Messi, Cristiano Ronaldo 等等。我们想要在这里指代我们当前的最佳方案。

综合上面的实验，我们尝试了数据增强、多种预训练模型与新的后接层以及模型融合，得出我们的 GOAT：使用 Stacking 方法融合这 5 个模型

- Bert-base+ 原始数据集
- Bert-wwm-ext+ 原始数据集
- Bert-wwm-ext+ 文本拼接数据集
- RoBERTa-wwm-ext+ 原始数据集
- RoBERTa-wwm-ext-large+ 回译数据集

获得我们当前的最高分：


**Figure 9: 房产行业聊天问答匹配 A 榜成绩**


**Figure 10: 房产行业聊天问答匹配 B 榜成绩**

## 6 CONCLUSION

至此，我们本次 CCF BDCI 之旅也告一段落了。我们选择房产行业聊天问答匹配这个赛题的初衷是希望解决赛题的同时了解买房卖房的热点问题，可以为我们以后买房提供借鉴经验。实际也确实如此，在我们后续分析数据集时我们了解到许多从来没有听说的名词和买房要考虑的问题。在整个比赛过程中，我们由浅入深，首先结合信息检索与数据挖掘课程内容，将机器学习与信息检索方法结合解决问题，虽然分数不理想，但具有重要意义。之后我们涉足相对陌生的 NLP 邻域，率先去了解了自然语言处理的一些基本概念，然后选择了适合于处理问答匹配任务的由谷歌提出的 BERT 模型，尝试了各种相关的预训练模型，并自己优化了后接层，更好地处理下游任务。为了进一步提高线上评测分数，我们采用了诸多 trick，比如数据增强、模型融合、Kfold 等等，也得到了不错的结果。除此之外，我们使用 Google Colab 和华为云的 ModelArts 为我们提供算力，也得以跑了 Large 模型。

这次比赛也给我们了很多启示：

一是数据集很重要，在做进一步的模型探究前，应该先分析数据，了解数据的分布，从而得知它是一个怎样的问题，训练它会出现什么问题，才能选择更适合它的模型；

二是理清下游任务，NLP 主要有四大下游任务，光有预训练模型显然不够，理清下游任务，设计更适合解决问题的全连接层才能对症下药；

三是理性地使用 tricks，模型融合是打比赛一个很有效的 trick，但要理性地使用融合，不能一味地堆叠各种模型，质量比数量更重要，应该先分析手头几个模型的特性和效果，选择效果不错或有特别之处的模型进行适当融合才能得到想要的提升。

## 7 APPENDICES

### 7.1 References

[1] Jason Wei and Kai Zou. 2019.EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In EMNLP-IJCNLP.
[2] Google《Recurrent Models of Visual Attention》2014.
[3] Google《Sequence to Sequence Learning with Neural Networks》NIPS 2014.
[4] Google《Attention is all you need》NIPS 2017.
[5] Google《Bert：Bidirectional Encoder Representations from Transformers》NAACL 2019.

### 7.2 Codes

相关代码链接：https://github.com/Scottyoung99/201800302011_Yyj_IR/tree/master/Project-GOAT-Beike