bestfitting 团队代码思路

语料处理

- 语料使用了训练集的最后一亿 + 测试集A + 测试集B，提取出query和title并用set去除重复项，保存到corpus文件下，作为总的语料库

word2vec和fasttext

- 根据上述语料，使用`gensim`官方推荐的用于处理大型语料库的`LineSentence Format`训练出`word2vec`和`fasttext`两种词向量，`class EpochSaver`用于查看每个epoch下的loss并保存。
- `word2vec`从第8个epoch开始loss异常地变成了0，最终我们选取的是第7个epoch的结果。
- 由于时间有限，`fasttext`只训练的20个epoch。
- 关于词向量训练的超参数，我们选择向量的长度是常规的300维，考虑到query与title可能有同样的词的缘故，`min count`选取2，最终得到了八十多万词的300维表示。

tfidf和countvectorizer

- 用`sklearn`模块下的`TfidfVectorizer, CountVectorizer`两个函数训练出了`tfidfvectorizer`和`countVectorizer`，用于后续的特征处理。

特征工程

总共提取了6类特征。

1. nuniuqe全局特征

计算query和title重复值出现的频数。

```
dataframe.groupby('query').title.transform('nunique')
dt.groupby('title').query.transform('nunique')
```

2. `tfidf`和`cv`相似度特征。

使用上面得到的`TfidfVectorizer`和`CountVectorizer`计算每个title和query的稀疏向量，进而利用`paired_cosine_distances, paired_euclidean_distances, paired_manhattan_distances`分别计算稀疏向量之间的余弦距离、欧几里得距离和曼哈顿距离，这些距离是后面模型输入的重要特征。

3. 长度特征

query、title的长度与其差、比值，共同出现的词数与各自出现unique的词数之间的比值等

4. query的局部或者整体是否存在于title中。

5. query、title之间的文本特征。

编辑距离，jaccrad相似度，最长公共子序列，最长匹配的长度，前缀后缀特征等文本特征。

6. 词向量相似度特征。

用`word2vec`和`fasttext`可以得到每句query、title的句向量表示，利用tfidf加权求得一个300维的向量，计算`scipy.spatial.distance`模块下的各种相似度距离。

注：在这些大量特征的计算过程中，我们为`DataFrame`的apply定义一个`func`函数，用`multiprocessing`模块的进程池加速特征计算的速度。query、title分别用长度15和30进行padding和truncating。最后，我们用`hdf5`模块存储特征数据集。

lightgbm 模型

在lightgbm模型中使用了三千万训练数据，用了特征工程中的几乎所有特征进行5折训练得到5个lightgbm模型，对这5个lgb模型加权后的测试集A得分为0.594。

根据这5个模型对测试集和训练集进行预测，得到的5维向量作为新的特征输入到神经网络中，利用了stacking的思想（最终的提交没有用到lightgbm的预测结果）

esim 模型

esim模型的输入为`pad_sequences`后的query、title和13个自定义特征（2个nuniuqe全局特征，6个tfidf、cv相似度特征和5个lgb的预测值），自定义特征使用mlp方式融入神经网络中，即在最后的`Dense`层中使用在`merged = Concatenate()([q1_rep, q2_rep, magic_dense])`将特征融入。

总共训练了两个esim模型，两个模型主要区别是embedding的词向量不同，一个是`word2vec、fasttext`的平均值，一个只使用`word2vec`，以此增加模型的多样性。

两个esim在测试集A的分数分别是0.6185和0.6184。因为embedding matrix不同，两个esim模型还是具有一定的差异性，最终我们把两个esim的预测结果加权平均（0.53+0.47），取得了比单一模型更好的效果（测试集A分数0.6227），这也是我们最终提交测试集B的方案。

2019-8-14

bestfitting团队