自然语言处理第一周项目文档—内容相似度分析

1.项目内容：

本次项目提供一系列的英文句子对，每个句子对的两个句子，在语义上具有一定的相似性；每个句子对，获得一个在 0-5 之间的分值来衡量两个句子的语义相似性，打分越高说明两者的语义越相近。

项目提供数据为 txt 文件，字段之间以 tab 分割。

训练数据文件，共有 1500 个数据样本，共有 4 个字段；第一个字段为样本编号，第二个字段为一个句子，第三个字段为另一个句子，第四个字段为两个句子的语义相似度打分，如下：

10001 two big brown dogs running through the snow. A brown dog running through the grass. 2.00000

10002 A woman is peeling a potato. A woman is slicing a tomato. 1.33300

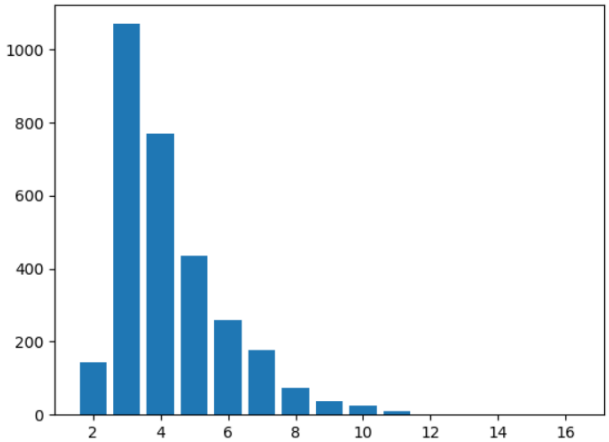
测试数据文件，共有 750 个数据样本，共有 3 个字段；第一个字段为样本编号，第二个字段为一个句子，第三个字段为另一个句子， 举训练样例说明如下：

10001 two big brown dogs running through the snow. A brown dog running through the grass.

10002 A woman is peeling a potato. A woman is slicing a tomato.

2.预处理：

预处理包括两个部分，一个是句子长度的统计，一个是词频统计。



通过句子长度发现大多数语句长度小于 13 个单词，且大多集中在 3 个到 4 个之间，只有一个在 16 个单词，基本就无视了。由此在初期阶段推断这个相似度应该和词的关系特别大。

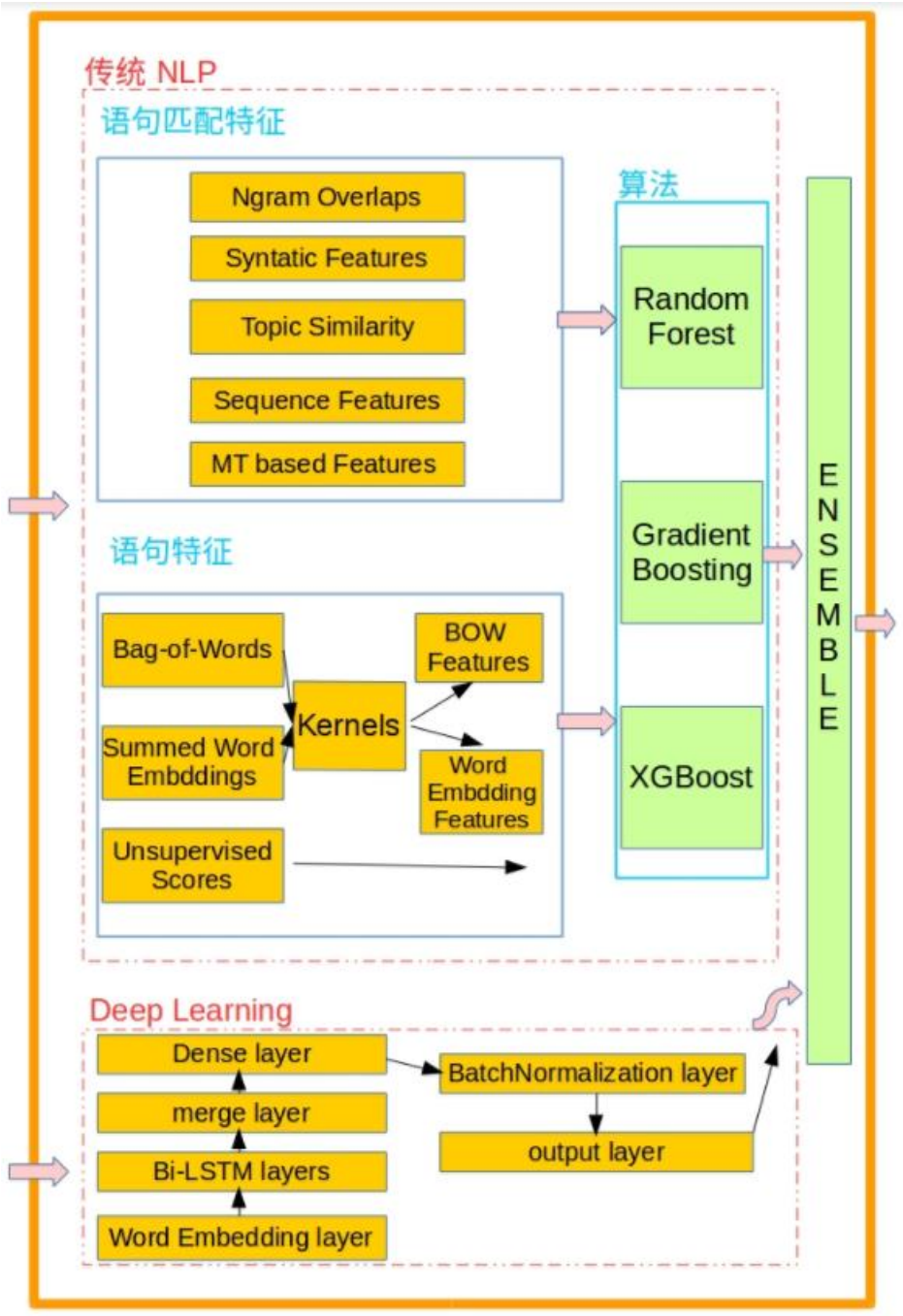
训练集词云

测试集词云

通过词频统计发现训练集和测试集的高频词其实差不多。也没有什么太多顾虑了。通过查询字典的方法查询了 glove.840B.300d，发现总共有 40 个词不在这个范围里，仔细研究后发现这几个词是错词，但因为比例很小，也没有在意了。

3.基本思路：

我们的模型按照下图展示主要由三部分组成：



传统 NLP 模型：

通过对每个 sentence pair 分别提取匹配特征（即词之间的相关性）和语句特征（即单个句子表征形式）来构建一个相似度计算模型。这些特征都将输入到最后的回归模型中，加以拟合得到我们的最终预测。

深度学习模型：

通过将 sentence pair 表征为离散向量的形式，然后输入到 end-to-end 的网络中，我们计算出最后语句的相似度。

Ensemble 模型：

用来求前面模块的加权平均值来得到最后的 score。

下面我们将具体展示各个特征的意义。

3.1 传统 NLP 模型

3.1.1 匹配特征

我们采用了五种 sentence pair 匹配特征来直接计算两个句子之间的相似度。

N-gram Overlap：

我们用 Si 来表示 n-grams 提取语句后后的集合，它的值通过下式定义：

$$ngo(S_1, S_2) = 2 * (\frac{|S_1|}{|S_1 \bigcap S_2|})^{-1}$$

在这里我们先对文本做了预处理，去除了 stopwords，然后针对 n={1,2,3,4}n={1,2,3,4}进行了 gram 提取,最后我们获得了 4 个 features。

Sequence Features(序列特征)

序列特征是另一种捕捉语句内部信息的方法，它通过计算词性标注后语句的最大公共子字符串的长度来计算语句间的相似度。我们先求取两个语句的 POS tags，然后再用公共子字符串的长度除以两个句子的总 tags 长度。最后我们得到了 2 个 features。

Topic Features

为了衡量语句间的相似性，我们采用了 Latent Dirichlet Allocation （LDA）模型。这里我们简单的设置了 topic 的数量为 6，对训练数据和测试数据进行了转化，得到 6 个特征。

Syntatic Features

除了直接从结构化的句子词性 list 中直接求解公共子字符，我们还采用了树形结构的词性分析，通过计算在树形结构下的最大 subpath，我们获取了一 1 个特征，此处词性树形结构的获取采用的是 nltk 库中的 tagger 函数。

MT based Features

BLEU 方法为机器翻译的评测方法之一，通常利用 n-gram 的方法计算两个文本之间的相似程度。

$$\mathrm{BLEUn} = \mathrm{BP} * \exp(\sum_{n=1}^{N} w_n * logp_n)$$

其中 BP 为惩罚因子，针对两个文本相差长度较大有作用，这里 n 取值为 4。

最后我们收集了 8 个 sentence pair 的特征。每个提取方向都有很大的可扩展性，如 n-grams 中，对原始文本和处理后的文本卡一进行 word level 的特征提取，对于序列模型考虑 longest common prefix/suffix 等，对 Syntatic Features 采用不同的 tree kernels，对 MT based Feature 采用 GTM-3，NIST Ol 等计算方式。由于时间和经验的双双不足，我们只采用了很少的特征～

3.1.2 语句特征

这里我们通过直接向量化每个句子，然后再通过向量之间的计算来获得句子之间的相似度分析。

BOW Features

我们将每句话都用 Bag-of-Word 方式转成向量，并用 tf-idf 进行加权处理。

Word Embedding Features

Word embedding 是一种将词转换成向量的方法。尝试过直接使用测试集，训练集语料训练 wordvec 模型，但因为数据量小，且是短文本，因而效果极差。后面使用了 glove 的与训练模型，与训练的模型语料来源于维基百科 11G 的英文文本，足以使得词向量内容准确，我们的词向量词典均来源于 glove.840B.300d

虽然这些特征能够很好地表征出语句的信息，尤其是对于语句间的相对关系方面，word2vec 一直是佼佼者。但是我们也可以发现两者的维数都达到了 K 级别，这是我们所不能接受的，毕竟前面的提取一共才 8 个！这不公平！所以我们通过采用核函数（kernel function）来对句中所有的词向量做一个整合，将它们从高维向低维转化。 最后我们 cosine distance、word mover distance、IDF 加权和调和平均等方法，得到了 3 个特征。

Unsupervied Score Features

无监督方法参考文献 http://nlp.arizona.edu/SemEval-2017/pdf/SemEval025.pdf

无监督方法主要的计算公式为：

$$\text{Score} = \frac{\sum TfIdf(w_i) * match(w_i, w_j)}{\sum TfIdf(S_1, S_2)}$$

其中，S1，S2 指两个句子，wi 表示存在于两个句子中的所有词，wj 为当前 wi 所处的句子的另一句中存在的词。match 函数表达如下：

$$\text{match} = \begin{cases} \max\left(cos\left(vec(w_i), vec(w_j)\right)\right), & w_j \text{的近义词中存在} w_i \text{或} w_j = w_i \\ 0, & w_j \text{的近义词中不存在} w_i \text{且} w_j \neq w_i \end{cases}$$

词向量使用 glove.840B.300d 向量，对于未出现的词，赋值为 300 维均为 1 的向量。

无监督方法最终在平台上效果为 0.79

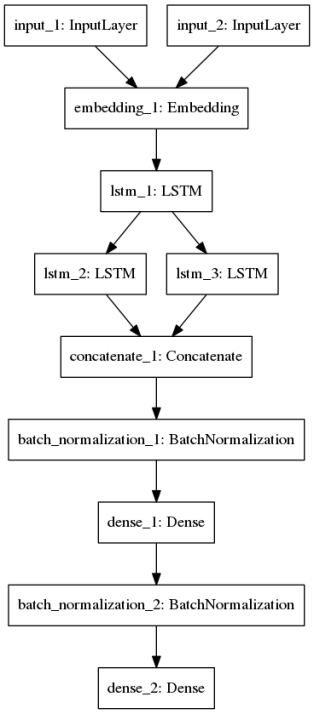
3.1.3 回归算法

我们一共使用了四种回归模型来预测最后的 score：SVR，Random Forests（RF），Gradient Boosting(GB) 和 XGBoost（XGB)。前面三种都是采用 scikit-learn 中实现，而 XGB 是采用 xgboost 中实现。通过一段时间的实验，由于 SVR 表现的效果不佳，最终被黯然离场。

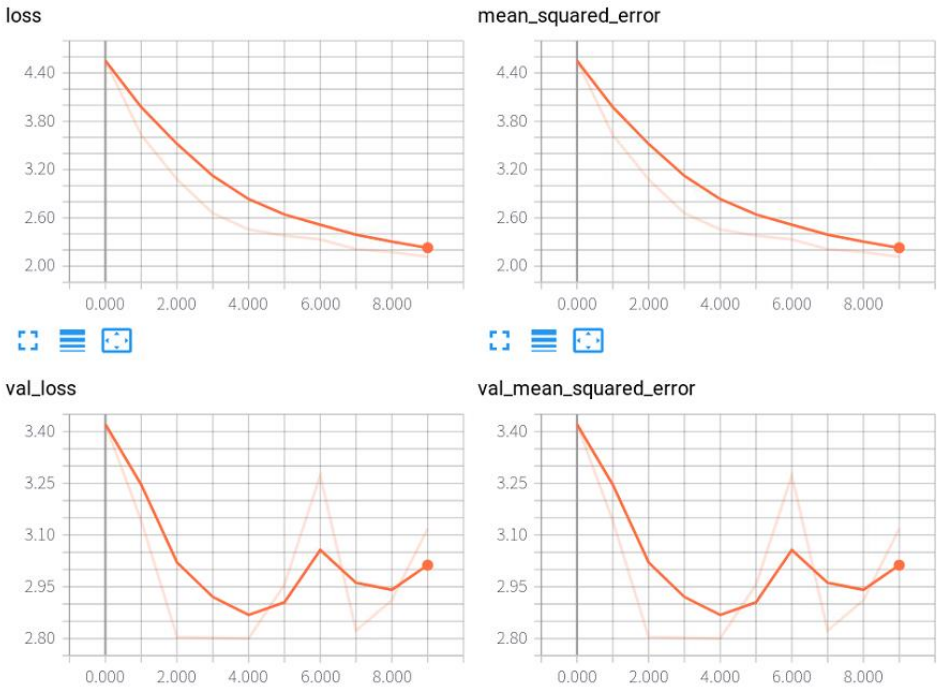
3.2 深度学习模型

神经网络方法主要由 lstm 融合，最后效果不是特别好，最好记录只有 0.6 的成绩，应该是参数没调整正确，时间不太够，就没有怎么尝试了。用的损失函数为均方误差，优化器为 adam

神经网络结构图如下，词向量，embedding 层采用 glove.840B.300d，第二轮 LSTM 均为反向 LSTM：



神经网络方法得到如下的结果图



神经网络实验图如图有 4 组，分别为训练集 loss，训练集方差值，验证集 loss，验证集方差值。很明显验证集的 loss 都开始往上走了训练集还没有到达一个较好的点。后期对神经网络主要是调这里的参数，数据量小，很容易达到拟合，也很容易过拟合。

3.3 Ensemble 模型

由于在最后的评估中，深度学习模型下的 score 表现较差，所有最后的 ensemble 中我们没有给它的 socre 赋权重，当前的结果都是基于传统 NLP 的 score。

4 实验过程及相关结果

4.1 数据集

数据全部来自于 training 中 1500 个样本，我们通过 nltk 包进行了分词、词性还原、去除 stopwords 后，对剩余样本进行简单的 sting 处理。如 10001 样本：

two big brown dogs running through the snow. A brown dog running through the grass

处理后为：

two big brown dog run snow brown dog run grass

所有的代码都将开源到 github 中： 送入模型前，我们使用了 sklearn 中的 train_test_split 函数将数据集划分成 0.7：0.3 的训练集和交叉验证集。然后采用 sklearn 中的

mean_squared_error 函数和 scipy 中的 pearsonr 函数作为我们验证的标准。（本次比赛的测试的得分，个人猜测就是 pearsonr 得分）

4.2 特征表现

下面我们首先对特征集中每一个维度的特征进行单独打分，验证出单个特征对于模型分类的贡献程度，评分标准为 均方差和 pearsonr 系数：

4.3 算法表现

同样，我们比较了各大算法的表现，结果如下

此处的结果中，各大算法用的是默认参数，并未进行特意的调参。

NLP_features	Feature_id	MSE	Pearsonr
Bow	Feature1	1.001731651056412	0.78045596293431219
Topic_sim	Feature4	1.8473856253236474	0.53179930348386584
N-grams	Feature16	1.192015725496709	0.74412026252105734
Syntatic_fea	Feature20	1.1511071694075183	0.76568358169188389
Word_emb	Feature23	1.0391790550017963	0.77148408902644672
Unsuper	Feature26	0.94397729796824703	0.79460488760853254
All_feature		0.83266723473	0.821753213223

Algorithm	MSE	pea
RF	0.791437677422	0.831331526812
GB	0.841775448927	0.820219618878
XGB	0.83266723473	0.821753213223
RF+GB+XGB	0.805710132129	0.827956148017

4.4 成绩

虽然在此处 Random Forest 表现出了最为优异的性能，但是在最后实际的 test 数据中，Random Forest 最佳的表现只有 0.83 左右，Ensemble 还是以 0.01 多的优势达到了总体最优 0.85，并且顺利地帮助我们达到了第五名的成绩。助教老哥们厉害的不行！

5 总结

这一周的学习中可以说是几经波折，最后看来结果还行。但是时间的分配不尽合理，花了近 3 天多的时间都在看论文，试图理解论文后去还原结果。导致后来 coding 和调参时间不足。对于特征的把握也不够全面，前前后后增加了三次特征。还是那句老话，具体问题具体分析，本来以为神经网络厉害完了，但对于小数据确实不太容易训练，波动很大，而另一方面，这个数据长度又短，词的含量也不多，所以传统方法能达到较好方法也和情理。