机器学习概论课程实验报告

Fake News Detection

孟垂正*

2017 年 6 月 11 日

目录

*学号: 2013010952, 邮箱: mengcz95thu@gmail.com, 电话: 17600739085.

1 问题介绍

该问题来源于比赛 Fake News Challenge[1]. 比赛的目标是利用人工智能技术提高检测一个消息是否为谣言的工作的自动化程度, 减轻人类审查员的负担. 本次挑战并没有打算一劳永逸地自动化全部的检测步骤, 而是试图完成其中比较重要的一个步骤–Stance Detection, 即检测不同的媒体对于同一个话题的态度. 很显然如果他们的态度相差很大或者负面态度居多, 该消息的可信性将大打折扣. 这一步工作将成为未来的自动谣言检测系统的重要组成部分之一.

2 相关工作

本次需要解决的问题属于 stance detection 问题, 即给出 2 段文本, 判断它们表达的观点是否一致. 类似的问题之前也被数次提出过并作为一些比赛的题目. 例如 2016 年的 SemEval-Task 6 要求参赛者判断给出的 twitter 文本是否支持其所属的主题, 并需要完成有监督/无监督 2 种情形下的任务 [1]. 在该比赛中深度学习的模型得到了广泛的应用, 分获第一名和第二名的 2 支队伍分别使用了 RNN[2] 和 CNN[3] 解决该问题.

Stance detection 问题是文本分类问题的一种, 目前已有较多使用 RNN 等模型完成文本分类任务的工作发表, 例如使用双向 RNN 进行分类 [4], 使用 RNN 与 CNN 的混合模型进行文本分类 [5], 以及使用引入 attention 机制的 RNN 模型进行分类 [6][7].

与传统的提取特征的方法相比, 以上的深度学习模型要求将文本转化为数值向量后输入模型进行计算, 即需要进行合适的 word embedding. 这一领域的工作有 Mikolov 等提出的 word2vec[8], Pennington 等提出的 GloVe[9] 和 Bojanowski 等提出的 Fasttext[10].

3 实验任务

实验需要完成的任务是: 在比赛方提供的有标注数据基础上构建模型, 判断消息正文与消息标题之间的关系, 包括同意, 反对, 不表态, 无关四种. 具体如下:

- 输入: 一个标题和一段正文 (二者可能来自同一个消息也可能不是).
- 输出: 标题与正文的关系, 分为以下 4 类:
  - Agrees: 正文内容同意标题内容;
  - Disagrees: 正文内容反对标题内容;
  - Discusses: 正文与标题讨论了同一个话题, 但没有表明态度;
  - Unrelated: 正文与标题讨论的话题不同.

数据来源是比赛方提供的有标注数据[2], 规模在 50k 篇文章左右, 包含了标题, 正文以及标题和正文之间的关系.

4 模型设计

4.1 Baseline

Baseline 模型来源于数据提供方[3]. 该模型从标题和正文中手工提取特征, 然后使用 Gradient Tree Boosting Classifier 进行分类.

提取的特征包括:

- word overlap features: 标题和正文词汇的 Jaccard 相似度 [11].
- refuting features: 由 refuting words 在标题中的出现情况组成的 0-1 向量. Refuting words 包括'fake', 'fraud', 'hoax', 'false', 'deny', 'denies', 'not', 'despite', 'nope', 'doubt', 'doubts', 'bogus', 'debunk', 'pranks', 'retract'. 若某个词在标题中出现则对应位置为 1, 否则为 0.

[1] http://www.fakenewschallenge.org/
[2] https://github.com/FakeNewsChallenge/fnc-1
[3] https://github.com/FakeNewsChallenge/fnc-1-baseline

- polarity features: 标题和正文表现出的态度的" 极性". 一段文本的" 极性" 的计算方法是统计前述 refuting words 在其中出现次数的奇偶性. 偶数次的否定体现出肯定的态度, 奇数次的否定则仍旧体现出否定态度.
- binary co-occurrence: 标题中的词汇在正文中出现的次数总和.
- binary co-occurrence stops: 除去 stop words(一些无实际意义的词汇, 例如 a, the, is 等) 的标题中的词汇在正文中出现的次数总和.
- count grams: 标题中 n 个连续字母/n 个连续单词在正文中出现的次数, 对于字母, 模型选取了 n=2,4,8,16; 对于单词, 模型选取了 n=2,3,4,5,6.

分类器使用决策树作为基分类器, 同时使用 Gradient Boosting 方法 [12] 作为增强学习方法. Gradient Boosting 方法与 Boosting 方法执行流程类似, 均为串行执行, 但前者每次更新时使用损失函数对分类结果的梯度而非真实结果训练新的基分类器, 并确定一个使得更新后损失结果最小的步长, 本轮更新后的分类结果为上一轮的结果 + 步长 * 本轮基分类器的结果. Gradient Boosting 的优势是预测能力较强, 缺点是不便于并行化.

4.2 Word Embedding

大多数机器学习方法更擅长处理数值类型的输入数据, 因此有必要使用 word embedding 将输入文本中的单词映射为 n 维向量. 一个好的 word embedding 应当将语义相近的词汇映射到 n 维空间的相近位置, 同时词向量之间的位置关系应当能较好地反应原词汇之间的关系. 考虑到本实验中可用数据较少, 直接使用预训练的模型效果更好.

综合考虑各个预训练模型的大小, 特征维数和使用的数据集大小, 本实验选择 Stanford NLP Group 在 twitter 数据集上训练的的 GLoVe 模型[4]作为 word embedding. 该模型包含 1.2 百万个单词, 结果为 25/50/100/200 维的词向量.

4.3 损失函数

对于神经网络模型, 需要选择合适的损失函数以进行参数求导和优化. 这里选择的损失函数为分类问题中常用的 Cross Entropy 函数, 该函数的形式如下:

$$L(X, Y) = -\sum_{n}\sum_{i} y_i^{(n)} \log\left(f(x^{(n)})_i\right)$$

其中 $y_i^{(n)}$ 为二值函数, 表示第 n 个样本的真实类别是否为 i, $f(x^{(n)})_i$ 为模型输出的将第 n 个样本预测为 i 类的概率. 可以观察到 $e^{-L(X,Y)}$ 实际上就是模型将输入 X 分类到 Y 的概率, 因此对该损失函数求极小相当于进行了极大似然估计.

[4]http://nlp.stanford.edu/data/glove.twitter.27B.zip

4.4 基于 CNN 的模型

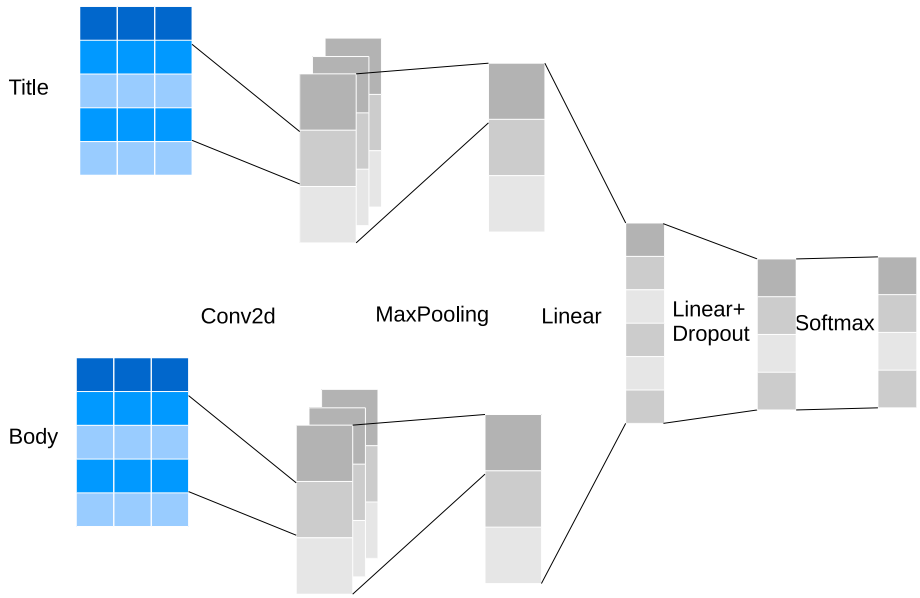

Figure 1: 用于分类的 CNN 模型示意图, 包括卷积, 池化, 线性层和 softmax 层

CNN 全称为卷积神经网络 (convolutional neural network), 该网络通过引入卷积核实现了对局部特征的提取, 同时减少了模型的参数数量. 在文本分类任务中, 卷积核可以在一定程度上捕捉上下文之间的词汇关系. 本实验用于文本分类的 CNN 模型组成部分主要有:

- word2vec: 将标题/正文文本通过预训练的 word2vec 模型转化为输入矩阵, 大小为文本长度 x 词向量维数.
- 卷积层: 卷积核大小为窗口大小 x 词向量维数, stride 为 1, 不进行 padding. 标题文本卷积核数量与词向量维数相同. 对于标题文本窗口大小为 3, 对于正文窗口大小为 5.
- 池化层: 使用 MaxPooling, 对每个卷积核的结果取最大值.
- 线性层: 将 title 和 body 文本经过卷积池化的结果拼接后输入线性层, 得到隐层结果.
- 线性层 +Dropout 层: 将隐层结果输入线性层得到长度为 4(类别数) 的输出层. 对隐层的结果进行一定概率 (0.1) 的 dropout 操作, 即使得该层神经元的输出在训练时有 0.1 的概率为 0, 在预测时恢复正常, 可以抑制过拟合现象的发生 [13].
- Softmax 层: 将输出层的结果输入 softmax 函数得到最终分类结果, 输出样本属于 4 种类别的概率.

4.5 基于 RNN 的模型

RNN 全称为循环神经网络 (recurrent neural network), 该网络要求输入数据按照一定序列进入网络, 时刻 t 输入的数据包括序列中的第 t 个数据和 t-1 时刻的网络输出结果. 这一特点使得 RNN 具有一定的记忆功能, 适合处理文本等序列数据, 因此 RNN 模型在自然语言处理领域得到了广泛的应用. 输入序列较长时 RNN" 记忆" 的向量会出现衰减现象, 但 2 种特殊的 RNN 单元 LSTM[14] 和 GRU[15] 的提出解决了这一问题. 以下的模型均使用 LSTM/GRU 作为网络基本单元.

下面几种 RNN 模型的区别主要在于 RNN 部分, 分类部分均使用有 1 个隐层的线性网络 +softmax 进行分类, 同时对隐层做 dropout 处理抑制过拟合.

文本序列均经过 GLoVe 模型转换为词向量序列, 为简便未在图中画出.

4.5.1 双向 RNN 模型

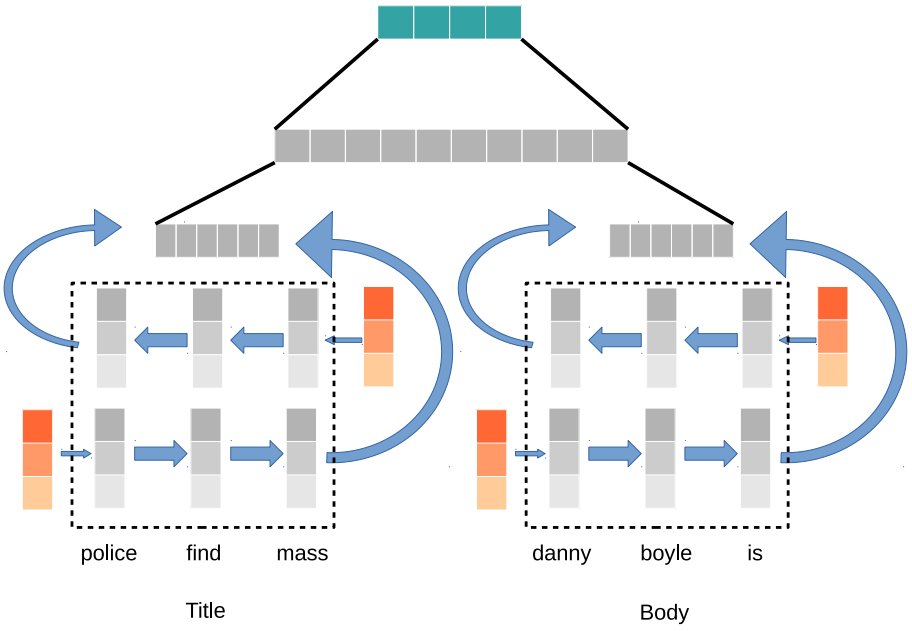

Figure 2: Unconditional Bidirectional RNN 模型示意图

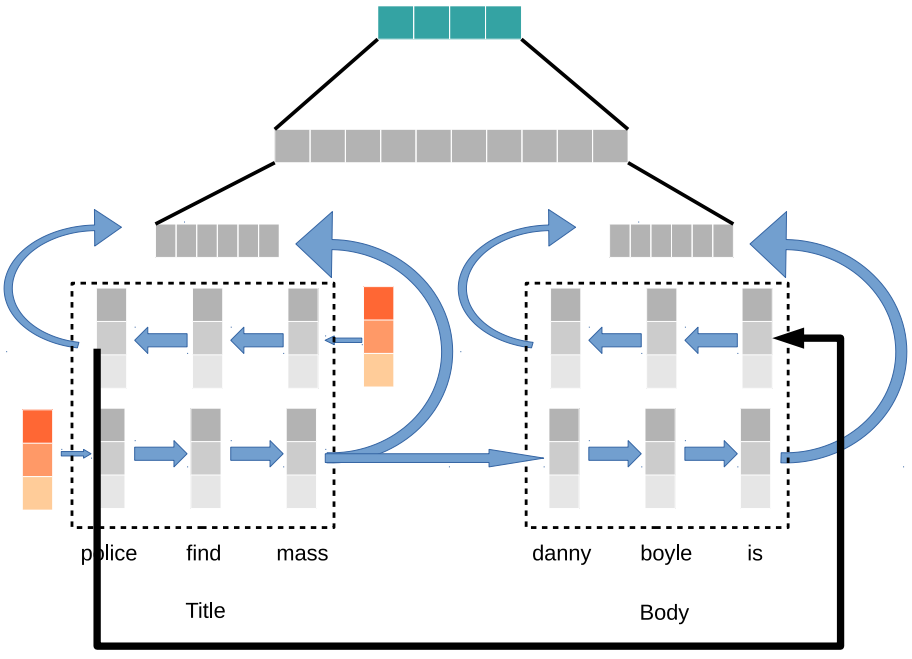

Figure 3: Conditional Bidirectional RNN 模型示意图

双向 RNN 模型将输入序列分别按照顺序和反序输入, 2 种顺序得到的最终向量拼接后作为 RNN 的输出向量. 将标题文本和正文文本的 RNN 输出向量拼接后送入分类网络进行分类.

根据处理正文文本的 RNN 使用的初始向量的种类, 可以构造 2 种不同的双向 RNN 模型:

- Unconditional Bidirectional RNN: 处理正文的 RNN 使用零向量作为初始向量.
- Conditional Bidirectional RNN: 处理正文的 RNN 使用处理标题的 RNN 的输出向量作为初始向量. 这种做法在标题和正文之间建立了联系, 可能会取得更好的效果.

4.5.2 引入了 attention 机制的 RNN 模型

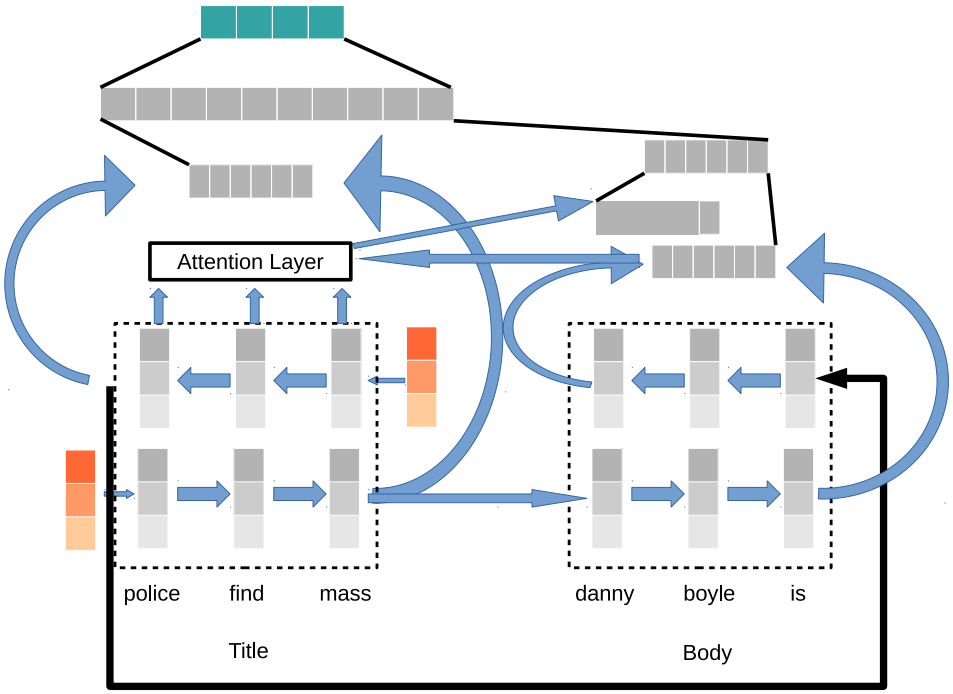

Figure 4: 引入 attention 的 Conditional Bidirectional RNN 模型 1 示意图

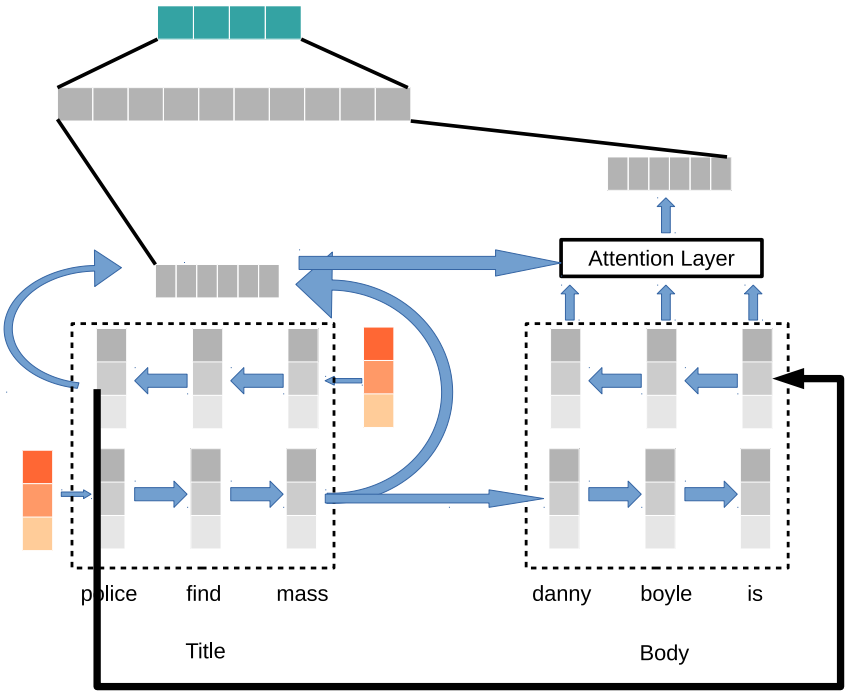

Figure 5: 引入 attention 的 Conditional Bidirectional RNN 模型 2 示意图

在 RNN 模型的基础上存在许多变种, 其中一种引入了所谓的 attention 机制 [6]. Attention 机制的原理是根据 RNN 最后一个单元的输出向量为所有单元的输出向量赋予不同的权重, 各个单元输出向量加权后与最后一个单元的输出向量拼接, 经过一个非线性层后输出. 根据前人的研究和实验验证, 效果较好的权重计算方式为: 最后一个单元的输出 h_T 与各个单元的输出 h_t 做双线性变换 $h_T^T W h_t + b$ 的结果经过 softmax 函数处理后作为 h_t 的权重, 非线性层的函数可选为 tanh.

具体到本实验的问题, 由于标题和正文在 2 个 RNN 中处理, 可以使用正文 RNN 的输出向量与标题 RNN 的所有输出向量做 attention, 结果与标题 RNN 最后的输出向量拼接后进入分类网络.

文献 [7] 对 attention 模型进行了一定简化, 主要区别是取消了非线性层, 直接使用加权后的结果作为输出向量. 该简化模型同样经修改应用到了本实验中.

4.6 混合模型

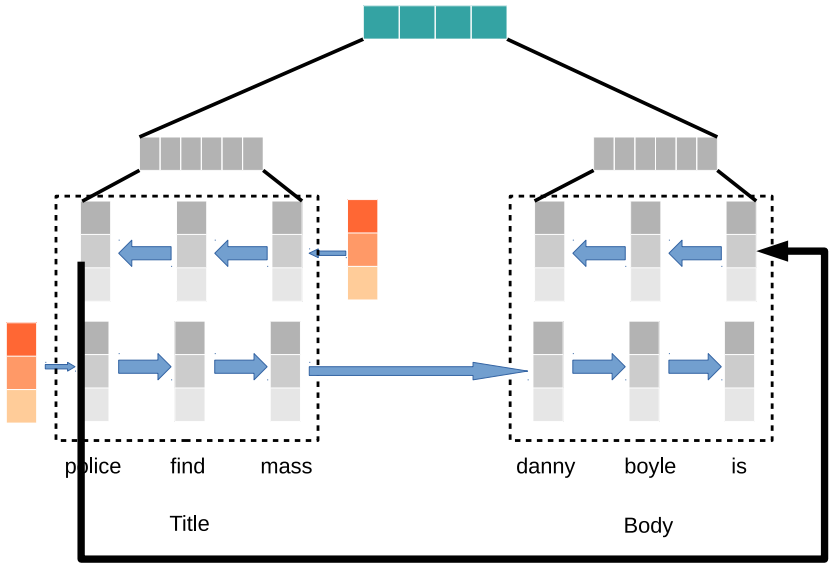

Figure 6: CNN-RNN 混合模型示意图

混合模型结合了 RNN 与 CNN 的 pooling 操作 [5], 使用 RNN 替代 CNN 中的卷积部分, 在文本分类等任务中有较好的效果. 具体组成如下:

- 卷积层: 使用 bidirectional rnn 各阶段的输出作为" 卷积" 结果.
- 非线性层: 各阶段的输出经过同一个线性层后被激活函数 tanh 处理.
- 池化层: 对各阶段的输出沿阶段数所在轴做 maxpooling.
- 分类层: 标题和正文分别经前述 3 层处理后的输出拼接后经过线性层和 softmax 得到分类结果.

5 实验

5.1 实验数据

实验数据由 Fake News Challenge 的主办方提供, 规模在 50k 篇文章左右, 包含以下 2 个文件:

- train_bodies.csv: 正文和对应的 ID;
- train_stances.csv: 态度, 标题, 标题对应的正文 ID.

实验中将 80% 的数据作为训练集, 20% 的数据作为测试集. 对于 Baseline 方法, 选择训练集的 10% 作为验证集进行 10-fold 交叉验证; 对于神经网络模型, 由于运行一次耗时较长且数据量有限, 故直接使用全部训练集数据进行网络训练.

经过简单分析可以发现提供的数据非常不平衡, 高达 73.13% 的例子属于 unrelated, 而 discuss, agree, disagree 的样例仅占 17.83%, 7.36% 和 1.68%. 若直接使用会使得分类结果出现严重偏差, 分类器很容易将所有样例判断为 unrelated.

常用的处理不平衡数据的方法包括 over sampling, under sampling 和人为增大损失函数中占比较低的类别的权重. Over sampling 对所有占比非最大的种类的数据进行有放回抽样, 直到所有类别的样例数目相等; under sampling 则对所有占比非最小的种类的数据进行不放回抽样, 使得所有类别的样例数目相等; 对于某些分类问题常用的损失函数 (例如 cross entropy 函数), 可以人为调整不同类别的权重. 显然实验中使用 under sampling 得到的数据集规模过小, 不利于深度学习方法的处理, 而调整权重的方法不能解决 SGD 类优化方法划分 batch 时很可能出现的某一占比极少的类别在某一个 batch 中完全不出现的情况从而造成优化速度下降, 因此使用 over sampling 预处理数据. 需要注意的是 over sampling 必须在划分训练集和测试集之后在训练集上进行, 否则测试集类别分布的信息会" 泄露" 到训练集中, 使得模型在测试集上的表现偏好.

数据的另一个问题是标题长度和正文长度相差极大, 标题的最大长度 <50, 而正文的最大长度 >2000, 人类在判断标题与正文的关系时往往阅读正文的开头部分即可, 因此过长的正文对于神经网络模型正确分类很可能意义不大. 出于加速训练和节省内存的考虑, 使用神经网络模型时仅使用正文的开头部分, 为了验证假设, 实验比较了截取不同长度正文时的分类效果.

5.2 评价标准

实验问题属于多分类问题, 采用以下的评价标准:

- 精确度 (accuracy): $\frac{\text{被正确分类的样例数}}{\text{样例总数}}$.
- 比赛方的评价方法: 对于每个样例, 若被成功区分 related(包括除了 unrelated 之外的 3 类)/unrelated, 可得到 0.25 分, 若被分类正确且属于除了 unrelated 之外的 3 类, 则可继续得到 0.75 分.

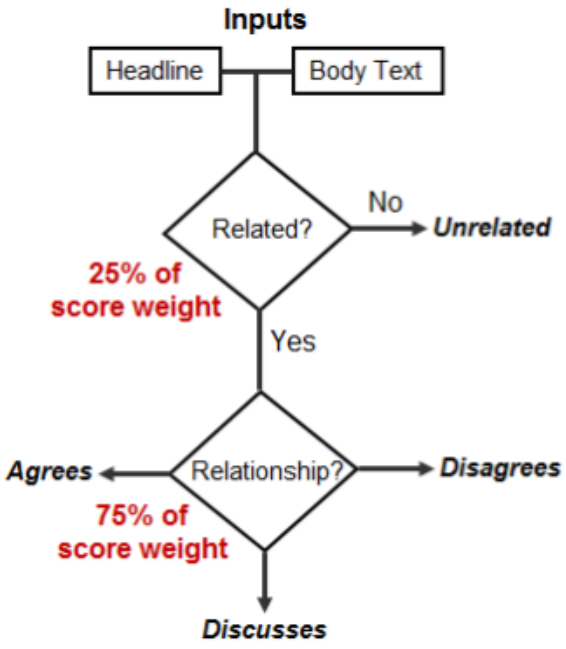

Figure 7: Fake News Challenge 评分标准

- 将多分类问题看做多个二分类问题的组合, 考察模型在每个二分类问题上的表现, 包括 precision,

recall 和 F1-score.

5.3 模型实现, 参数设置与运行环境

Baseline 模型由比赛主办方提供, 基于 scikit-learn 中的 GBDT 分类器实现. 其余的神经网络模型基于 PyTorch 框架[5]实现. PyTorch 框架提供了常见的 CNN, RNN 单元, 同时提供了自动求导功能, GPU 加速功能和多种优化器, 便于使用者搭建灵活的网络结构和使用 GPU 加速训练.

参数设置见附录, 本实验无意探究众多模型超参数对结果的影响, 因此仅比较了少数超参数的不同和模型结构的不同对分类结果的影响.

Baseline 模型运行平台硬件配置为 Intel i7-6600u@3.0GHz, 16GB RAM, 软件配置为 Ubuntu 16.04 LTS Desktop, Python 3.6.0. 神经网络模型运行平台为 Amazon Web Services 的 p2.xlarge 实例, 硬件配置为 4 核心 CPU, 64GB RAM, Nvidia K80 GPU(显存 12GB)x1, 软件配置为 Ubuntu 16.04 LTS Server, Python 3.6.0.

5.4 实验结果

5.4.1 词向量维度的影响

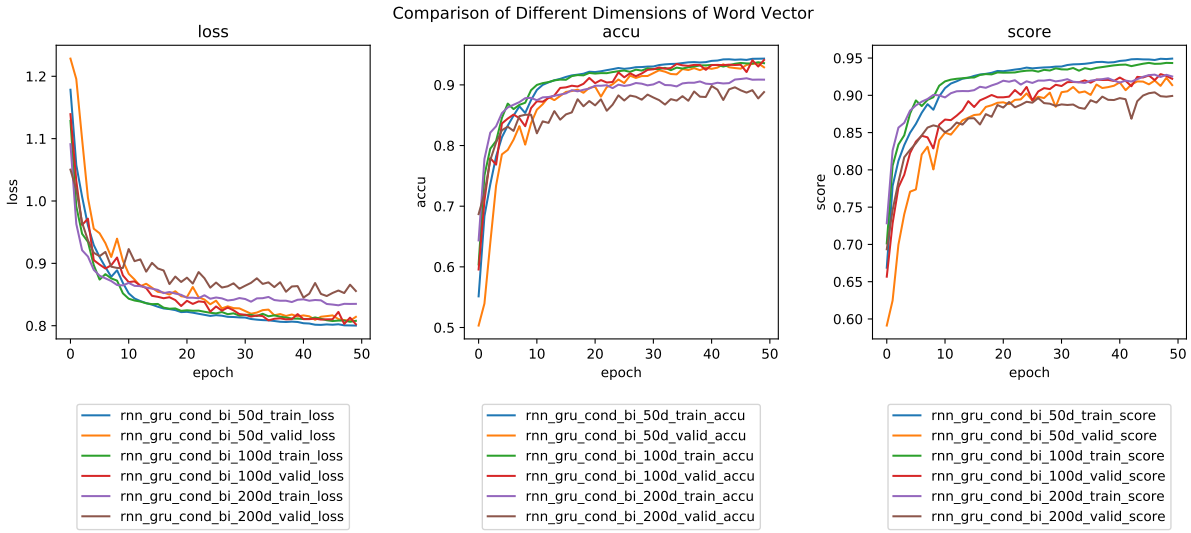

Figure 8: 不同词向量维度下的结果比较

Table 1: 不同词向量维数在测试集上的最好表现

Dim of Word Vector	Accuracy	Score
50	0.9356	0.9235
100	**0.9406**	**0.9285**
200	0.8872	0.9040

GLoVe 模型提供了多种维度的词向量. 词向量维数过低会导致词间关系无法充分体现, 词向量维数过高则会大幅增加模型参数数量, 容易出现过拟合, 同时训练速度下降. 实验中分别选取了长度为 50, 100 和 200 的词向量, 比较它们在 Conditional Bidirectional RNN 模型上的效果. 可以发现长度为 50 和 100 的词向量表现相近, 且均好于长度为 200 的词向量. 原因可能是用于训练的数据量不大, 高维词向量更强的表现力难以在小规模数据集上体现, 反而因难以优化导致表现不好.

出于提高训练速度的考虑, 后续实验均选择 50 维的词向量.

[5]http://pytorch.org/

5.4.2 词向量训练方式的影响

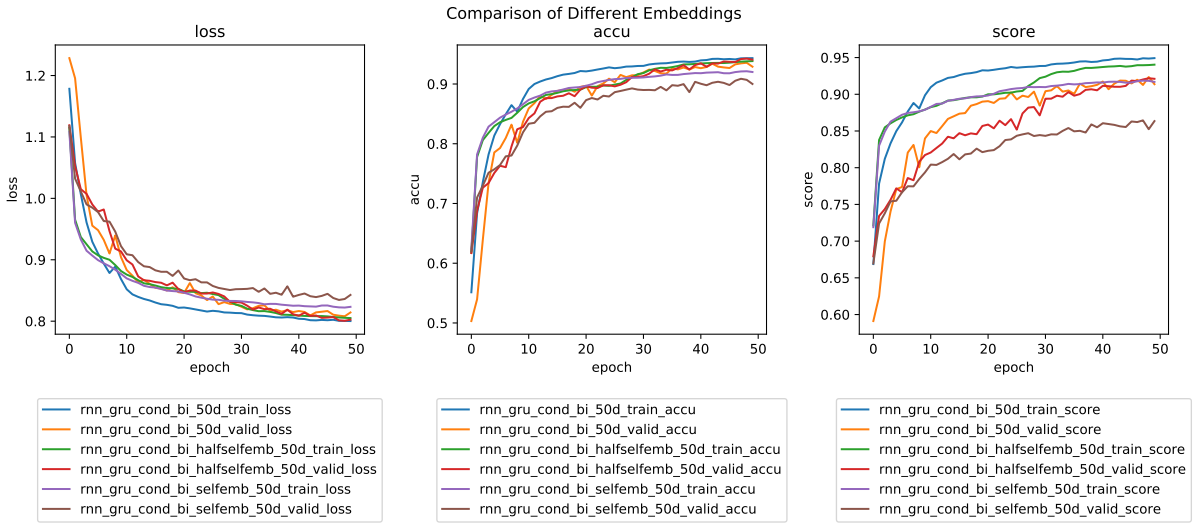

Figure 9: 不同词向量训练方式的结果比较

Table 2: 不同词向量训练方式在测试集上的最好表现

Embedding	Accuracy	Score
GLoVe Fixed	0.9356	**0.9235**
GLoVe Not Fixed	**0.9426**	0.9219
Self Embedding	0.9087	0.8645

本实验比较了 3 种词向量的训练方式: 直接使用预训练的模型 (GLoVe Fixed), 在预训练模型基础上调整 (GLoVe Not Fixed) 以及从零开始训练 (Self Embedding). 由结果可见直接使用预训练模型效果较好, 在预训练模型基础上调整词向量对结果没有改善, 而从零开始训练词向量的结果最差. 原因是用于训练的文本量太小, 无法正确反应词汇之间的关系.

5.4.3 正文截取长度的影响

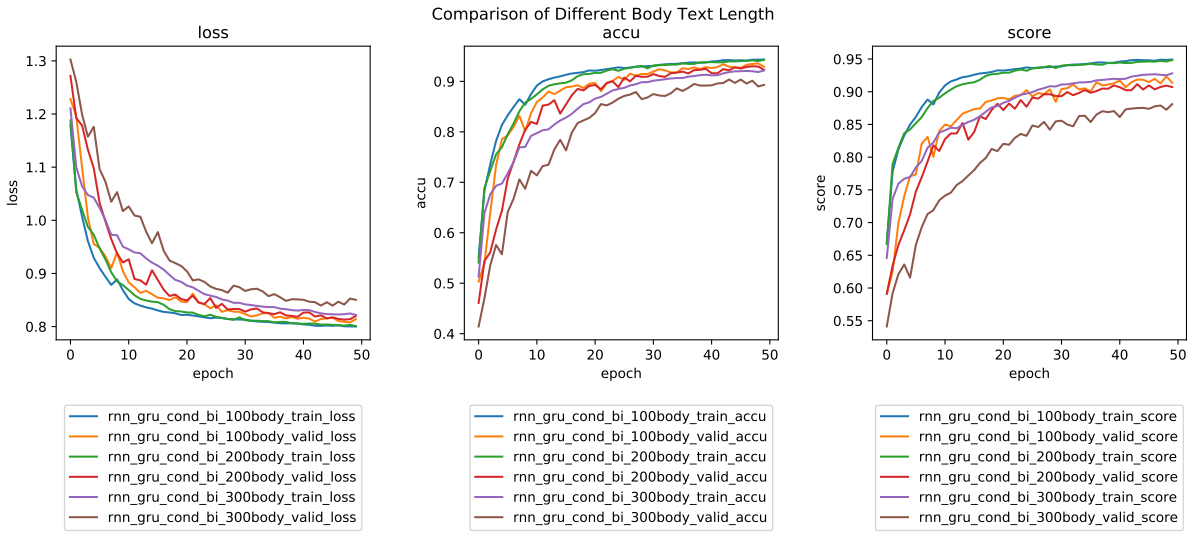

Figure 10: 不同正文截取长度下的结果比较

Table 3: 不同正文截取长度在测试集上的最好表现

Body Text Length	Accuracy	Score
100	**0.9356**	**0.9235**
200	0.9220	0.9116
300	0.8930	0.8811

正文截取长度过短可能导致正文的态度无法被充分提取, 截取长度过长则会大幅增加模型的资源消耗和训练时间. 实验比较了正文截取长度分别为 100, 200 和 300 时 Conditional Bidirectional RNN 模型的表现. 可以发现截取长度为 100 和 200 时模型表现相近, 且均好于截取长度为 300 时的表现. 原因可能是过长的正文使得模型更难训练, 同时正文与标题长度差距过大会减弱模型对标题特征的提取能力.

出于提高训练速度的考虑, 后续实验均选择 100 的正文截取长度.

5.4.4 RNN 单元种类的影响

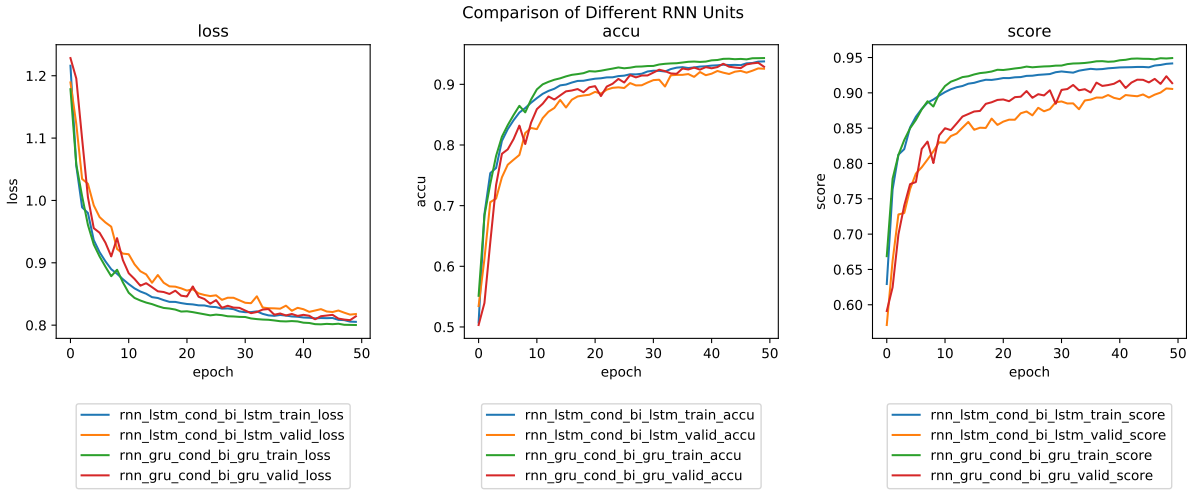

Figure 11: 不同 RNN 单元的结果比较

Table 4: 不同 RNN 单元在测试集上的最好表现

Type of RNN	Accuracy	Score
GRU	**0.9356**	**0.9235**
LSTM	0.9263	0.9063

两种 RNN 单元在测试集上的表现非常接近, 鉴于 GRU 的训练速度略快一些, 后续实验均选用 GRU 作为 RNN 单元.

5.4.5 模型效果比较

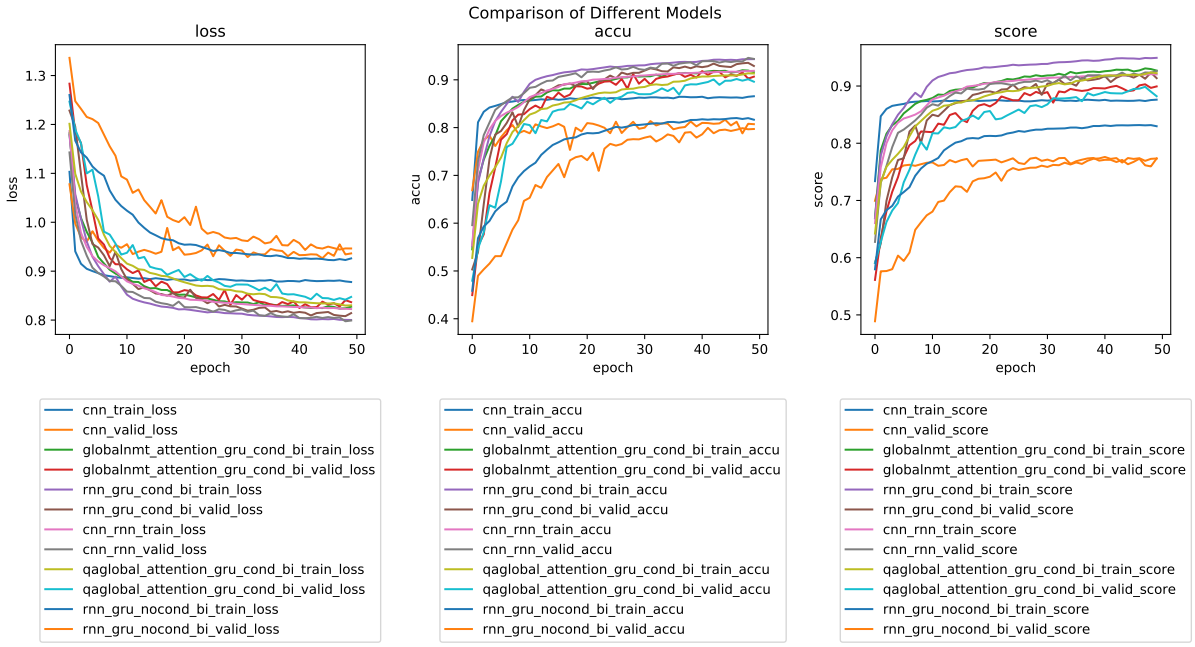

Figure 12: 不同模型的结果比较

Table 5: 不同模型在测试集上的最好表现

Model	**Train**		**Test**	
	Accuracy	**Score**	**Accuracy**	**Score**
Baseline	-	-	0.8774	0.7953
CNN	0.8613	0.8741	0.8122	0.7747
Unconditional Bidirectional RNN	0.8175	0.8311	0.7860	0.7760
Conditional Bidirectional RNN	0.9431	0.9486	0.9356	0.9235
Attention RNN I	0.9200	0.9308	0.9156	0.9030
Attention RNN II	0.9112	0.9211	0.8982	0.8986
CNN-RNN	0.9188	0.9203	**0.9438**	**0.9262**

Table 6: Agree 分类结果

Model	**Train**			**Test**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	-	-	-	0.6051	0.1549	0.2466
CNN	0.8946	0.8654	0.8797	0.4887	0.7203	0.5823
Unconditional Bidirectional RNN	0.8569	0.8143	0.8351	0.4718	0.6913	0.5608
Conditional Bidirectional RNN	0.9361	0.9186	0.9273	0.7507	0.829	0.7879
Attention RNN I	0.8865	0.9136	0.8999	0.5902	0.8391	0.693
Attention RNN II	0.8657	0.9406	0.9016	0.6181	**0.8609**	0.7196
CNN-RNN	0.9216	0.9048	0.9131	**0.8211**	0.7783	**0.7991**

Table 7: Disagree 分类结果

Model	Train			Test		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	-	-	-	0.25	0.0185	0.0345
CNN	0.9186	0.9677	0.9425	0.4595	0.7553	0.5714
Unconditional Bidirectional RNN	0.9141	0.8065	0.857	0.3746	0.6596	0.4778
Conditional Bidirectional RNN	0.95	0.9188	0.9342	**0.6857**	**0.766**	**0.7236**
Attention RNN I	0.9412	0.8598	0.8987	0.6536	0.6223	0.6376
Attention RNN II	0.9691	0.8113	0.8832	0.6269	0.6702	0.6478
CNN-RNN	0.9343	0.8032	0.8638	0.6413	0.6277	0.6344

Table 8: Discuss 分类结果

Model	Train			Test		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	-	-	-	0.6608	0.8483	0.7429
CNN	0.8853	0.7601	0.818	0.6859	0.6957	0.6907
Unconditional Bidirectional RNN	0.8057	0.8071	0.8064	0.6551	0.7183	0.6852
Conditional Bidirectional RNN	0.9366	0.968	0.952	0.8644	0.9206	0.8916
Attention RNN I	0.9041	0.9517	0.9273	0.8286	0.8931	0.8596
Attention RNN II	0.9045	0.954	0.9286	0.7937	0.9035	0.8451
CNN-RNN	0.8909	0.9728	0.93	**0.8725**	**0.9245**	**0.8978**

Table 9: Unrelated 分类结果

Model	Train			Test		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Baseline	-	-	-	0.9564	**0.9849**	**0.9704**
CNN	0.7603	0.8521	0.8036	0.9104	0.8513	0.8798
Unconditional Bidirectional RNN	0.7243	0.8443	0.7797	0.9124	0.8325	0.8706
Conditional Bidirectional RNN	0.95	0.9671	0.9585	**0.9822**	0.9537	0.9678
Attention RNN I	0.9364	0.9394	0.9379	0.9805	0.9237	0.9513
Attention RNN II	0.919	0.9388	0.9288	0.978	0.9063	0.9408
CNN-RNN	0.9182	0.9793	0.9478	0.9747	0.9656	0.9701

实验比较了前述模型在测试集上的效果, 总结如下:

Baseline 模型虽然在测试集上的整体表现一般, 但在进行 related/unrelated 分类时效果最好, 原因是 Baseline 模型中的许多特征是从单词/短语层面直接比较文本的相似程度, 因此能够很好地判断 2 个文本的相关性. 但 Baseline 模型中关于态度的特征很少, 故在区别 agree/disagree/discuss 时表现很差.

CNN 模型和 Unconditional Bidirectional RNN 模型的表现较差, 甚至不如 Baseline 模型, 原因是 2 个模型中只有对标题文本和正文文本分别处理的部分, 标题和正文之间的联系实际上只能靠最后一层的分类器来提取, 极大地影响了分类效果.

Conditional Bidirectional RNN 模型通过将处理标题的 RNN 的输出作为处理正文的 RNN 的输入, 建立了标题与正文之间的联系, 因此在测试集上取得了较高的精确度.

引入 Attention 机制的 RNN 模型的表现与常规的 RNN 相比并没有提高, 甚至有所下降. 观察结果可以发现该模型在预测 related 关系时在测试集上表现较差, 可能是 over sampling 造成的影响.

CNN-RNN 模型在测试集上表现最好. 卷积层的引入使得模型能够充分利用 RNN 中每一步的信息,

提高了模型对文本特征的抓取能力.

分别观察各个模型在 4 种关系上的分类表现可以发现, 尽管 over sampling 等方法可以一定程度上避免模型过早收敛, 但原始数据的不平衡对分类仍然造成了较大的影响. 在原始数据中极少的 agree, disagree 两种类型上分类时, 各个模型在测试集上的表现显著差于训练集; 在稍多一些的 dicuss 上分类时, 测试集表现与训练集表现的差距会缩小; 在最多的 unrelated 类型上分类时, 测试集的表现甚至略好于训练集.

6 结论

本实验尝试在 stance detection 问题上使用不同的深度学习模型, 其中 Conditional Bidirectional RNN 模型和 CNN-RNN 模型都取得了不错的分类效果. 深度学习模型不需要人工提取特征, 也能够在该问题上取得较好的结果, 但缺陷是模型的可解释性较差. 同时也应当注意到基于特征工程的传统方法在特征选取得当时, 同样能够在特定的问题 (如 unrelated 的分类) 上取得很好的效果. 无论是特征工程方法还是深度学习模型, 都需要对问题本身有深入的理解才能提取出恰当的特征/构建合理的网络结构, 从而更好地解决问题.

参考文献

[1] Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. *Proceedings of SemEval*, 16, 2016.

[2] Guido Zarrella and Amy Marsh. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*, 2016.

[3] Wan Wei, Xiao Zhang, Xuqin Liu, Wei Chen, and Tengjiao Wang. pkudblab at semeval-2016 task 6: A specific convolutional neural network system for effective stance detection. *Proceedings of SemEval*, pages 384–388, 2016.

[4] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[5] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273, 2015.

[6] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[7] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[11] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. Using of jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, page 6, 2013.

[12] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[15] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

A 运行参数设置

除特殊说明外, 程序使用的运行参数默认如下:

Table 10: 运行默认参数

Parameter	Meaning	Default Value
datapath	预训练 word2vec 模型所在文件夹	"./data"
word2vecmodelfile	预训练 word2vec 模型文件	"glove.twitter.27B.50d.txt"
fix_embedding	训练过程中不更新词向量	False
train_ratio	训练集比例	0.8
remove_stopwords	是否除去文本中的 stopwords	False
batch_size	单个 batch 大小	384
epoch	在整个训练集上的迭代次数	50
cuda	是否开启 CUDA 加速 (需要 NVIDIA GPU 支持)	False
lr	学习率	0.001
weight_decay	L2 罚项系数	1e-6
title_trunc	标题截取长度	40
body_trunc	正文截取长度	100
seed	随机数种子	0
wechat	开启微信通知功能, 通过文件传输助手实时发送训练进度	False
model	模型名称	必填
modelpath	存储训练中间结果的文件夹	必填
optimizer	优化算法名称	"Adam"
selfemb	指定维数时自行训练词向量, 不使用预训练模型	None
resume	从 modelpath 中最后一个 checkpoint 恢复训练	False

B 程序运行说明

- 运行单个模型: python main.py model modelpath [其他参数]. 参数设置可以参照表格10 或者使用 python main.py -h 命令查看帮助.
- 运行本实验中的全部模型: 运行 python worker.py 即可. 这里维护了一个大小为 2 的进程池, 可以并行训练 2 个模型.