BERT调优技巧与Kaggle竞赛

Yuanhao

wuyhthu@gmail.com

2021.01.15

我的NLP相关Kaggle竞赛

- Toxic Comment Classification Challenge, 2018-3
  - Bronze, 243/4550, top 6%
- Quora Insincere Questions Classification, 2019-2 (kernel only)
  - Silver, 140/4037, Top 4%
- PetFinder.my Adoption Prediction, 2019-4 (NLP+CV+data mining)
  - Gold, 3/2023, Top 1%

前BERT时代

- Gendered Pronoun Resolution, 2019-4 （躺）
  - Silver, 20/838, Top 3%
- Jigsaw Unintended Bias in Toxicity Classification, 2019-7
  - Gold, 10/3165, Top 1%
- TensorFlow 2.0 Question Answering, 2020-1
  - Silver, 16/1233, Top 2%
- Tweet Sentiment Extraction, 2020-6
  - Gold, 7/2227, Top 1%

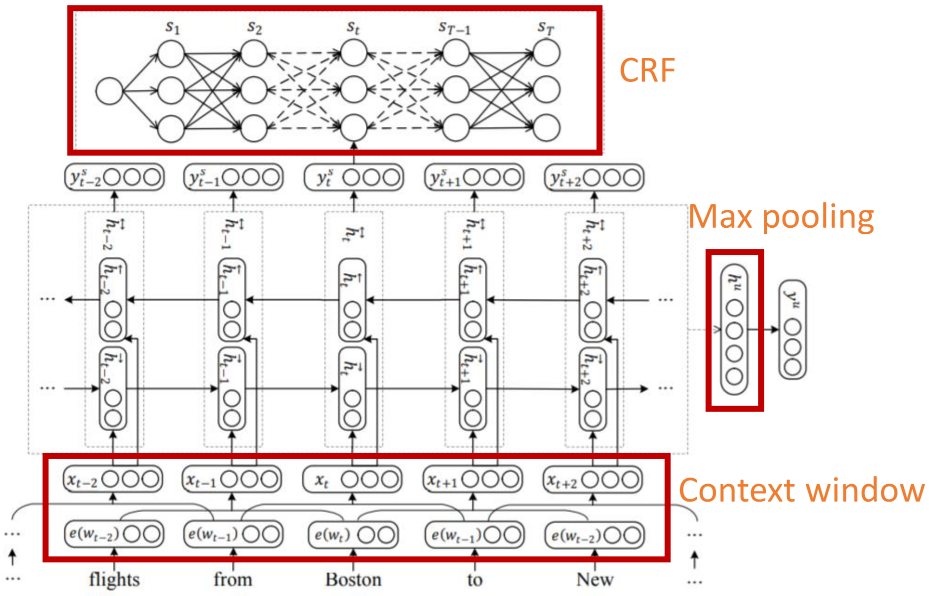
BERT时代

目录

1. 传统语言模型与NLP Pipeline回顾
2. Transformer和BERT
3. BERT Pipeline与调优技巧
4. Kaggle竞赛案例
    1. Jigsaw Unintended Bias in Toxicity Classification
    2. Tweet Sentiment Extraction

传统语言模型与NLP Pipeline回顾

- 原始文本
  - 清洗: 适配embedding
- 得到Token ID
- 得到字词Embedding
- RNN/CNN 模型
- Attention/Pooling
- 分类器/回归器/etc.



https://www.ijcai.org/Proceedings/16/Papers/425.pdf

传统语言模型与NLP Pipeline回顾

Glove :
Found embeddings for 32.77% of vocab
Found embeddings for 88.15% of all text

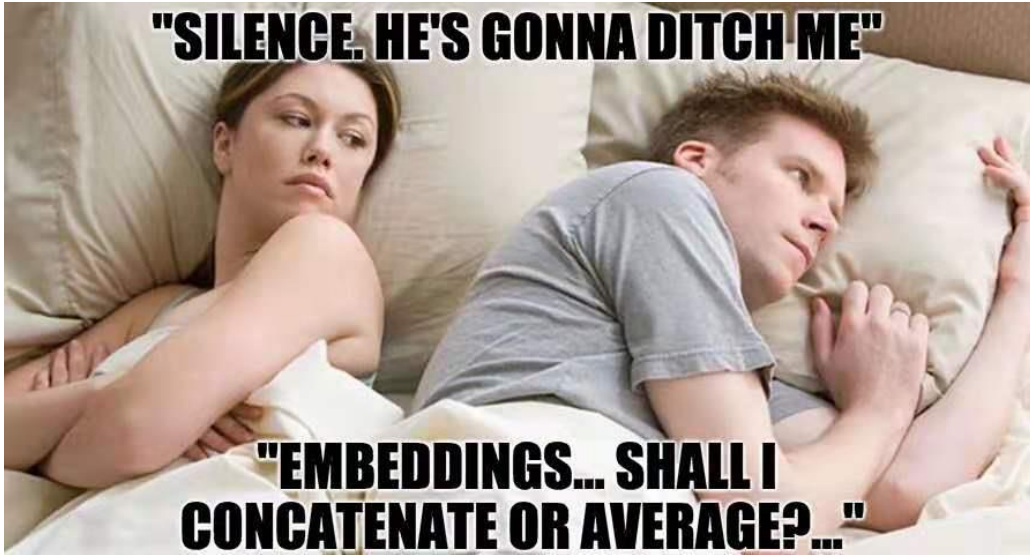
Glove :
Found embeddings for 69.09% of vocab
Found embeddings for 99.58% of all text

```
mispell_dict = {'colour': 'color', 'centre': 'center', 'favourite': 'favorite', 'travelling':
'traveling', 'counselling': 'counseling', 'theatre': 'theater', 'cancelled': 'canceled', 'labou
r': 'labor', 'organisation': 'organization', 'wwii': 'world war 2', 'citicise': 'criticize', 'y
outu ': 'youtube ', 'Qoura': 'Quora', 'sallary': 'salary', 'Whta': 'What', 'narcisist': 'narcis
sist', 'howdo': 'how do', 'whatare': 'what are', 'howcan': 'how can', 'howmuch': 'how much', 'h
owmany': 'how many', 'whydo': 'why do', 'doI': 'do I', 'theBest': 'the best', 'howdoes': 'how d
oes', 'mastrubation': 'masturbation', 'mastrubate': 'masturbate', "mastrubating": 'masturbatin
g', 'pennis': 'penis', 'Etherium': 'Ethereum', 'narcissit': 'narcissist', 'bigdata': 'big dat
a', '2k17': '2017', '2k18': '2018', 'qouta': 'quota', 'exboyfriend': 'ex boyfriend', 'airhostes
s': 'air hostess', "whst": 'what', 'watsapp': 'whatsapp', 'demonitisation': 'demonetization',
'demonitization': 'demonetization', 'demonetisation': 'demonetization'}
```

https://www.kaggle.com/theoviel/improve-your-score-with-some-text-preprocessing

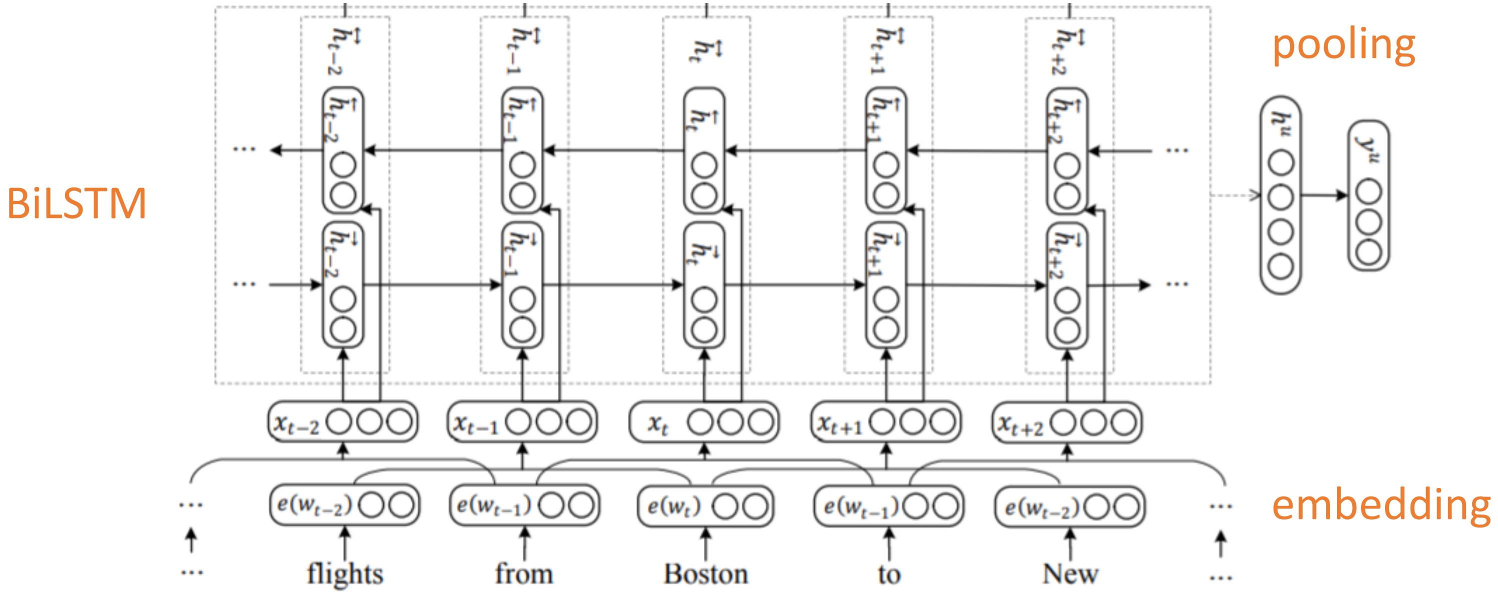
传统语言模型与NLP Pipeline回顾

- Embedding有各种操作..



https://www.kaggle.com/wowfattie/3rd-place

传统语言模型与NLP Pipeline回顾



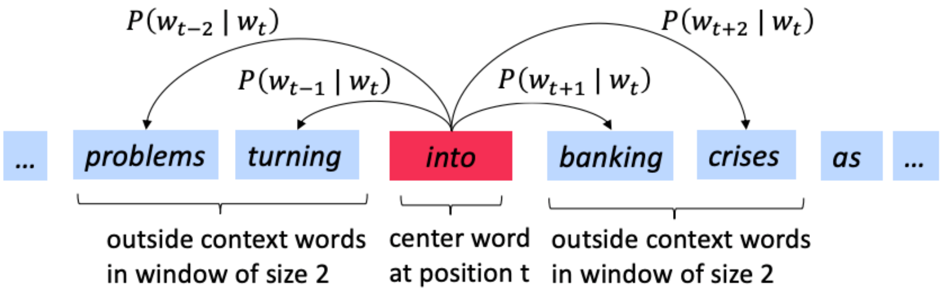
https://www.ijcai.org/Proceedings/16/Papers/425.pdf

传统语言模型与NLP Pipeline回顾

- **繁琐** 数据清洗 & 预处理
- **庞大** Embeddings（Glove 840b 2.2M tokens, Tencent 8M 中文 embedding)
- 从头训练需要 **大量** 数据（freeze embedding）

Transformer 和 BERT

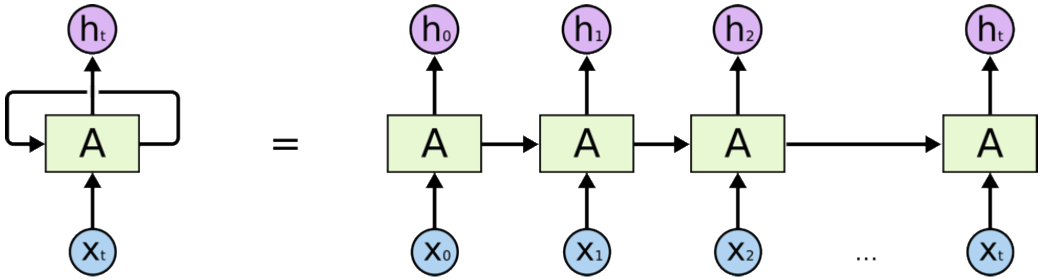
- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- 2018以前最主要的语言模型是字词Embedding
- 在BERT之前, 有许多成功的语言模型，如COVE, ELMO, ULMFIT, GPT
- 已经有一些模型开始尝试更好地利用上下文
- BERT 彻底改变了Kaggle NLP competitions的方式

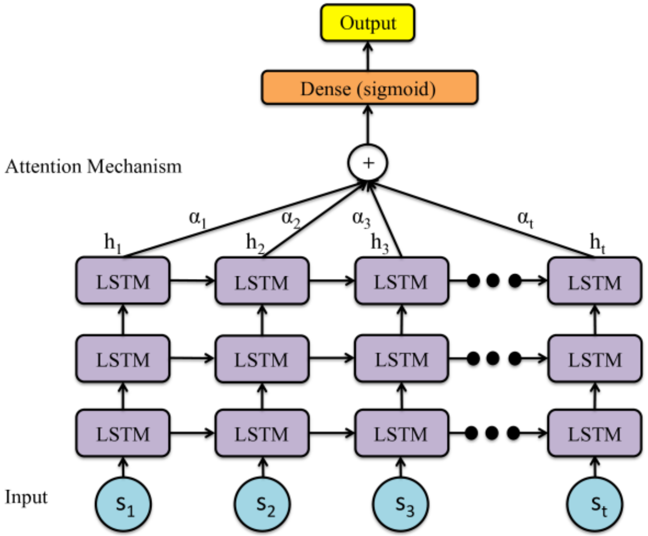


COVE: http://arxiv.org/abs/1708.00107
ELMO: http://arxiv.org/abs/1802.05365

Transformer 和 BERT

- 循环神经网络
  - 注意力机制非常重要





$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{V}\mathbf{h}_{t-1})$$

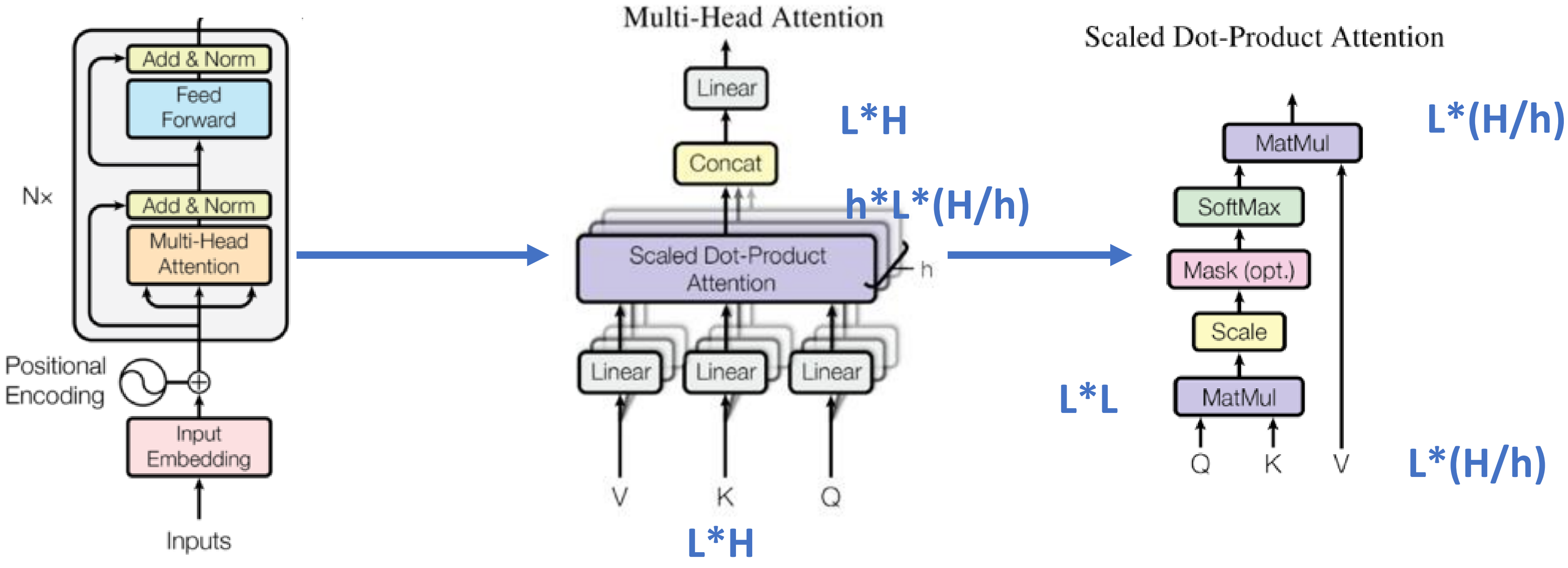
$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{V}(\sigma(\mathbf{U}\mathbf{x}_{t-1} + \mathbf{V}(\sigma(\mathbf{U}\mathbf{x}_{t-2}))))$$

$$\longrightarrow \quad \frac{\partial E_3}{\partial U} = \frac{\partial E_3}{\partial out_3}\frac{\partial out_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial U}$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

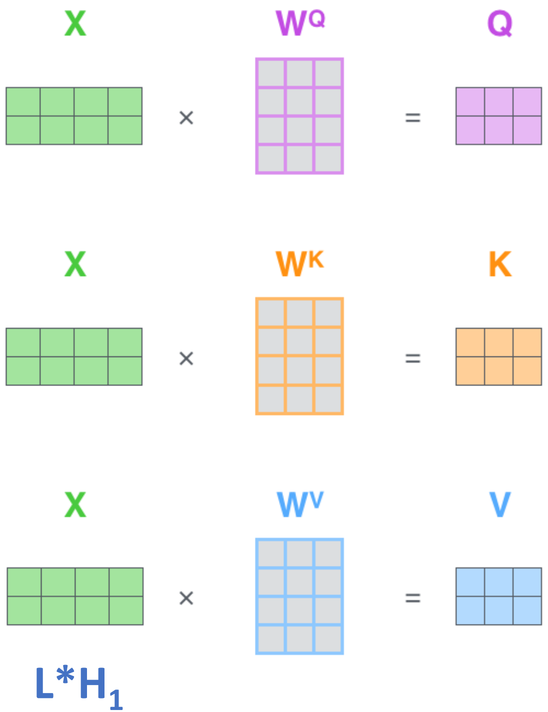
Transformer 和 BERT

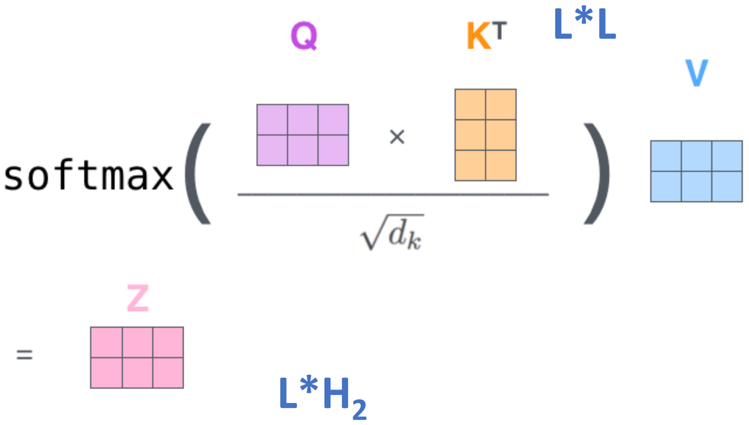
- Transformer



(Transformer) http://arxiv.org/abs/1706.03762

Transformer 和 BERT



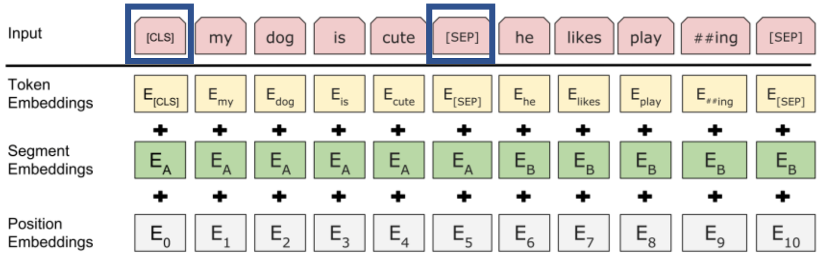


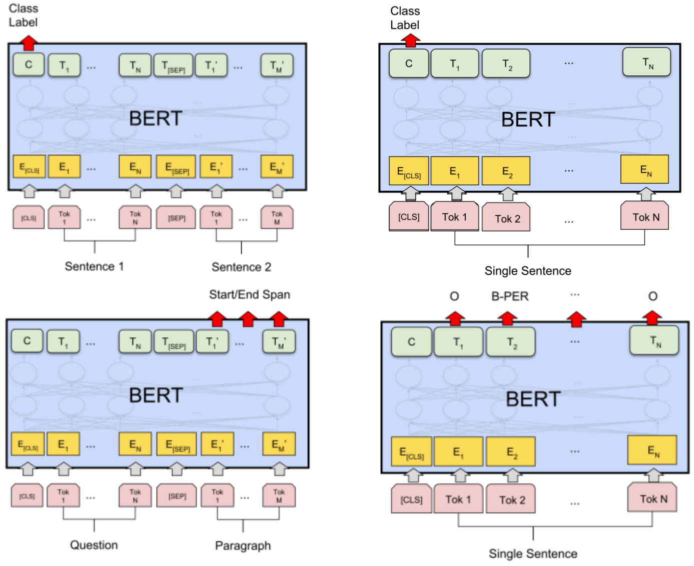
- H_1 embedding维数
- H_2 是隐层维数
- 通常 $H_1 = H_2$

http://jalammar.github.io/illustrated-transformer/

Transformer 和 BERT

- 更好的tokenizer
  - WordPiece (2.2M ---> 30k)
- 更好的模型结构
  - Transformer, layer norm, gelu, position embedding etc.
- 更好的预训练任务
  - Masked LM(MLM)
  - Next sentence prediction(NSP*)
- 更大量的数据 (BERT) https://arxiv.org/abs/1810.04805
  (ALBERT) http://arxiv.org/abs/1909.11942





BERT Pipeline与调优技巧

Transformers

- 原始文本
  - 几乎不需要做清洗
- BERT token id
  - word piece tokenizer, 词典尺寸大大压缩(~30k)
- BERT model
  - Base/large/cased/uncased/Chinese/wwm/ernie
- 任务相关头: 分类器/回归器/etc.
- https://github.com/huggingface/transformers

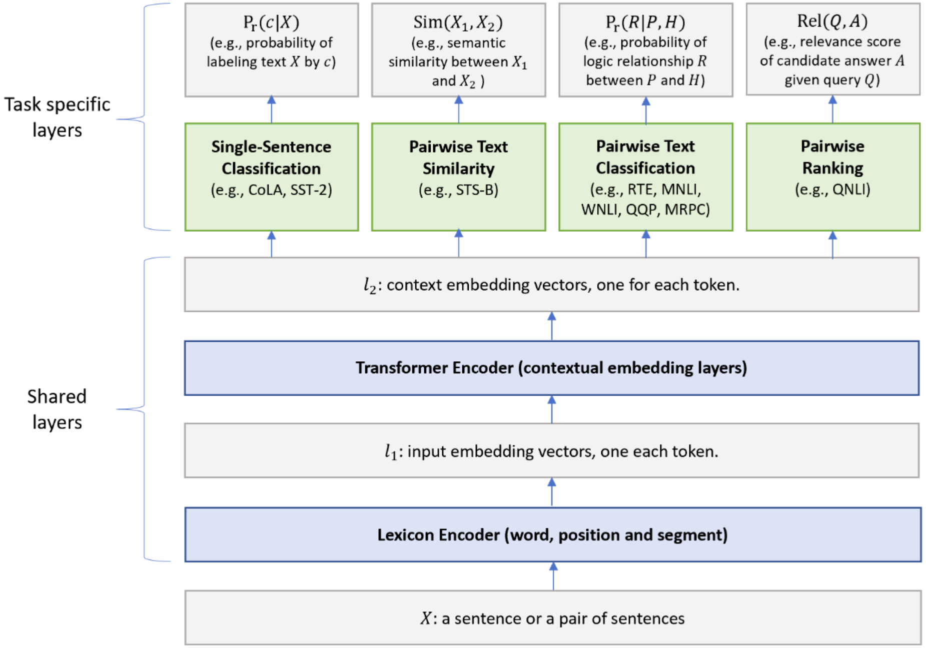
BERT Pipeline与调优技巧

模型的使用已经没有什么门槛 😂

```
import torch
from transformers import *

# Tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# several models for different down-stream tasks
BERT_MODEL_CLASSES = [BertModel, BertForPreTraining, BertForMaskedLM, BertForNextSentencePrediction,
BertForSequenceClassification, BertForMultipleChoice, BertForTokenClassification, BertForQuestionAnswering]

for model_class in BERT_MODEL_CLASSES:
        # load pretrained model
        model = model_class.from_pretrained('bert-base-uncased’)
        # token ---> token id
        input_ids = torch.tensor([tokenizer.encode(“Let‘s see all hidden-states and attentions on this
text”)])
        # get output
        sequence_output, pooled_output, (hidden_states), (attentions) = model(input_ids)
```

BERT Pipeline与调优技巧

- 多任务学习
  - 构建或利用辅助标签
- 更好的截断策略
  - 头 + 尾效果通常较好



(finetune BERT) http://arxiv.org/abs/1905.05583

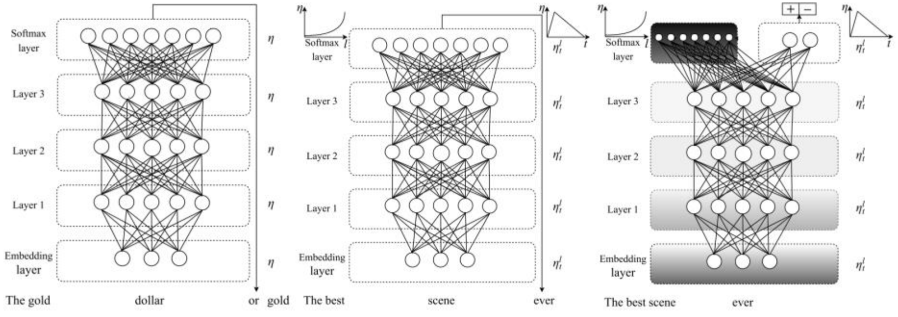
(MT-DNN) http://arxiv.org/abs/1901.11504

BERT Pipeline与调优技巧

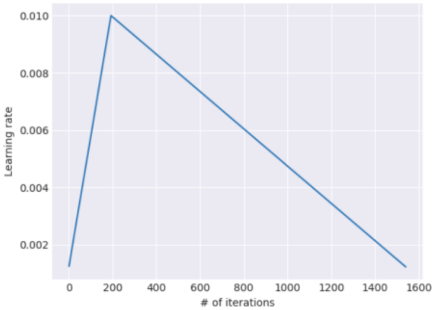
- 用领域数据精调语言模型
  - 当数据量大时提升明显
- 逐层减小学习率
  - 提升不明显
- 更好的学习率
  - 已是默认操作

(finetune BERT) http://arxiv.org/abs/1905.05583

(ULMFit) http://arxiv.org/abs/1801.06146



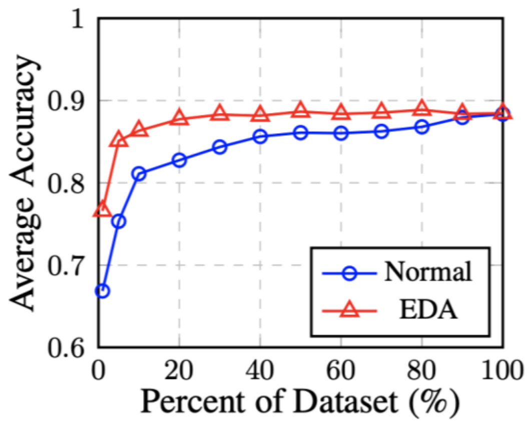
(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning



Warmup-linear learning rate

BERT Pipeline与调优技巧

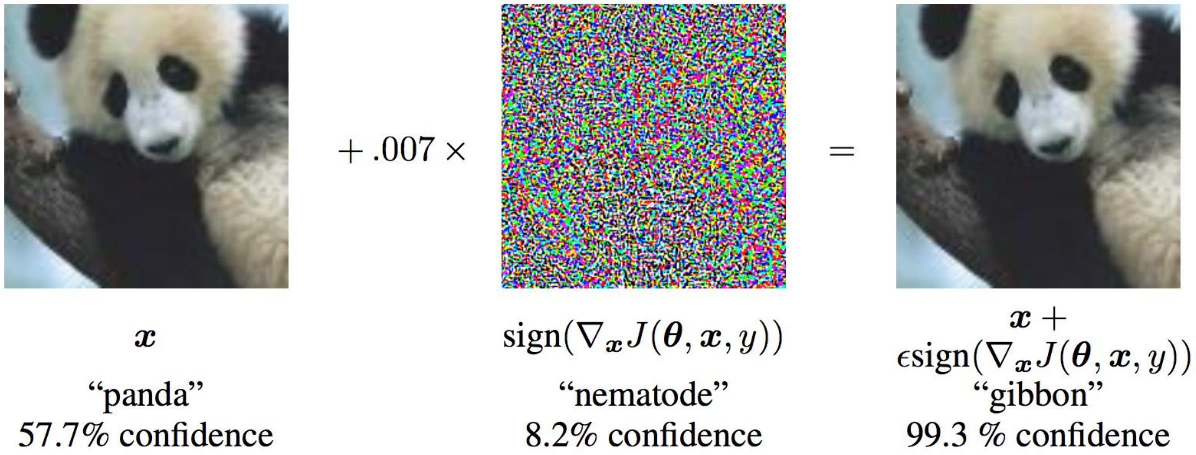
- 对文本数据增强
  - **Synonym Replacement (SR):** Randomly choose *n* words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
  - **Random Insertion (RI):** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this *n* times.
  - **Random Swap (RS):** Randomly choose two words in the sentence and swap their positions. Do this *n* times.
  - **Random Deletion (RD):** For each word in the sentence, randomly remove it with probability *p*.
- 对标签数据增强



https://github.com/jasonwei20/eda_nlp

BERT Pipeline与调优技巧

- 对抗训练
  - FGSM（Fast Gradient Sign Method）和FGM（Fast Gradient Method）
  - $\delta = \epsilon \cdot \mathrm{sign}(g)$
  - $\delta = \epsilon \cdot \ (g/\|g\|)$



BERT Pipeline与调优技巧

- 对抗训练

```
# 初始化
fgm = FGM(model)
for batch_input, batch_label in data:
    # 正常训练
    loss = model(batch_input, batch_label)
    loss.backward() # 反向传播，得到正常的grad

    # 对抗训练
    fgm.attack() # 在embedding上添加对抗扰动
    loss_adv = model(batch_input, batch_label)
    # 反向传播，并在正常的grad基础上，累加对抗训练的梯度

    loss_adv.backward()
    fgm.restore() # 恢复embedding参数
    # 梯度下降，更新参数
    optimizer.step()
    model.zero_grad()
```

```
import torch
class FGM():
    def __init__(self, model):
        self.model = model
        self.backup = {}

    def attack(self, epsilon=1., emb_name='emb.'):
        # emb_name这个参数要换成你模型中embedding的参数名
        for name, param in self.model.named_parameters():
            if param.requires_grad and emb_name in name:
                self.backup[name] = param.data.clone()
                norm = torch.norm(param.grad)
                if norm != 0 and not torch.isnan(norm):
                    r_at = epsilon * param.grad / norm
                    param.data.add_(r_at)

    def restore(self, emb_name='emb.'):
        # emb_name这个参数要换成你模型中embedding的参数名
        for name, param in self.model.named_parameters():
            if param.requires_grad and emb_name in name:
                assert name in self.backup
                param.data = self.backup[name]
        self.backup = {}
```

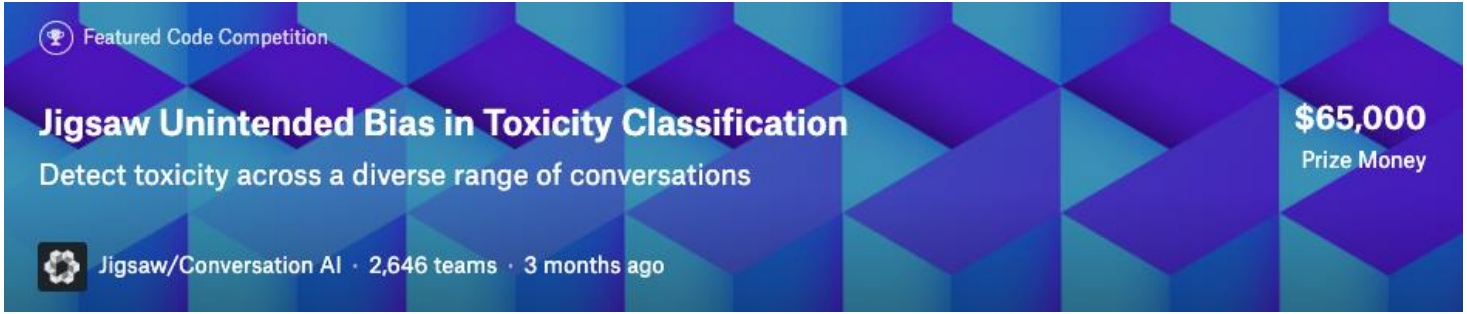
https://zhuanlan.zhihu.com/p/91269728

BERT Pipeline与调优技巧

- 对抗训练
  - https://github.com/huggingface/transformers/tree/master/examples/text-classification

任务	普通训练得分	对抗训练得分	普通训练耗时	对抗训练耗时
RTE	**0.6498(acc)**	0.6173(acc)	0:57	1:46
MPRC	0.8631(f1/acc)	**0.8752(f1/acc)**	1:26	2:40
WNLI	0.5633	**0.5633**	-	-
STS-B	0.8877(cor)	**0.8946(cor)**	2:17	4:08

Kaggle竞赛案例1



- 是一个比较新颖的文本分类比赛，有许多子指标来评估模型的偏见
- 1.8M 训练集, 97.3k 测试集
- 1 个主标签列+7 个辅助标签, 9 个标识（identity）列
- 上一版比赛: 160k 训练集+160k 测试集, 只要预测一个标签

Kaggle竞赛案例1

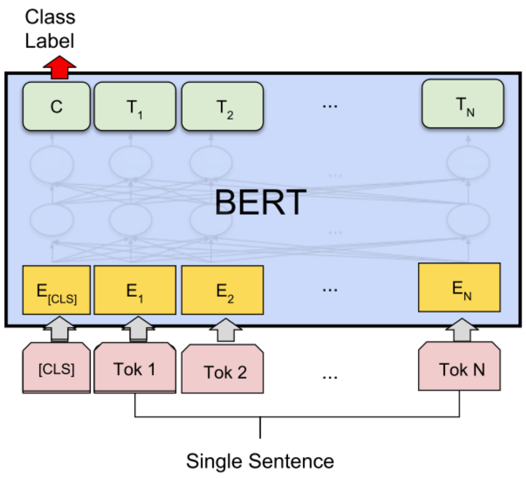
- Identities
  - male, female, homosexual_gay_or_lesbian, Christian, jewish, muslim, black, psychiatric_or_mental_illness
- Auxiliary targets
  - severe_toxicity, obscene, threat, insult, identity_attack, sexual_explicit

Target	Comment_text	Severe_ toxicity	Obscene	Identity_ attack	Sexual_ explicit	insult	threat	black	…
0.0	This is so cool. It's like, 'would you want your mother to read this??' Really great idea well done	0.0	0.0	0.0	0.0	0.0	0.0		
0.0	Thank you!! This would make my life a lot less anxiety-including. Keep it up, and don' let anyone get in your way!	0.0	0.0	0.0	0.0	0.0	0.0		

https://www.kaggle.com/nz0722/simple-eda-text-preprocessing-jigsaw

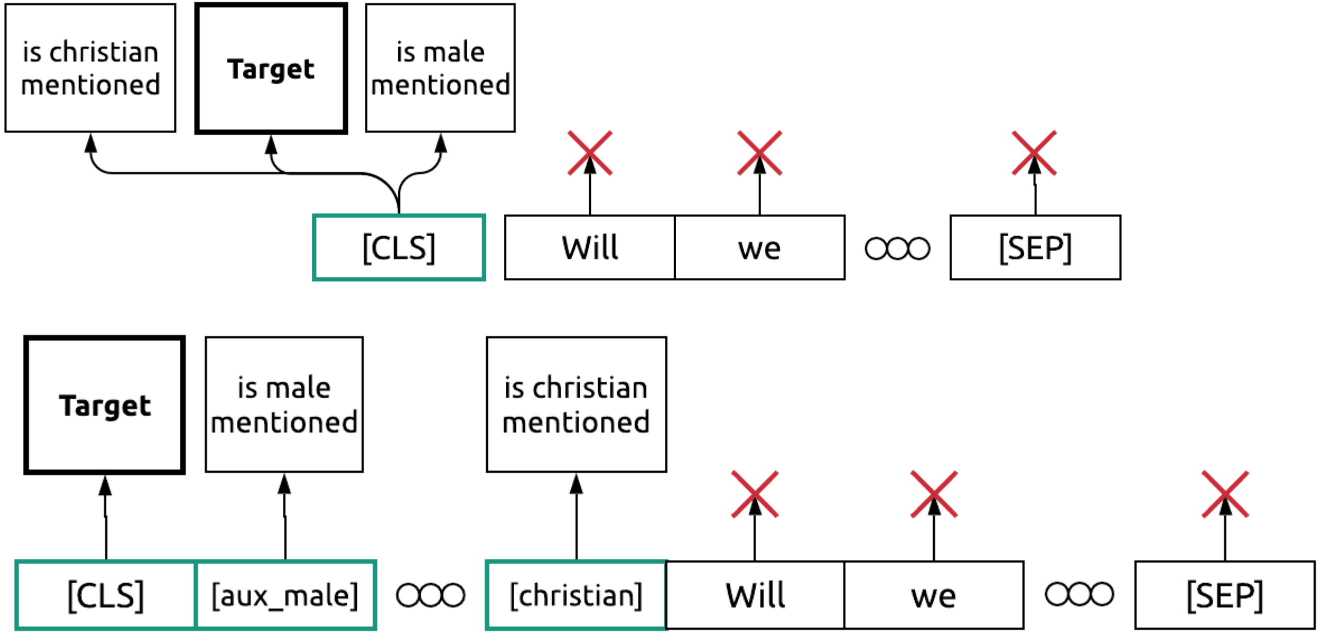
Kaggle竞赛案例1

- 在比赛语料上进一步预训练语言模型
  - 1/2/3/4/10
- 多任务学习
  - 预测标识列和辅助标签
  - 1/2/4/10
- 更好的截断策略
  - 3/10
- 逐层递减学习率
  - 10



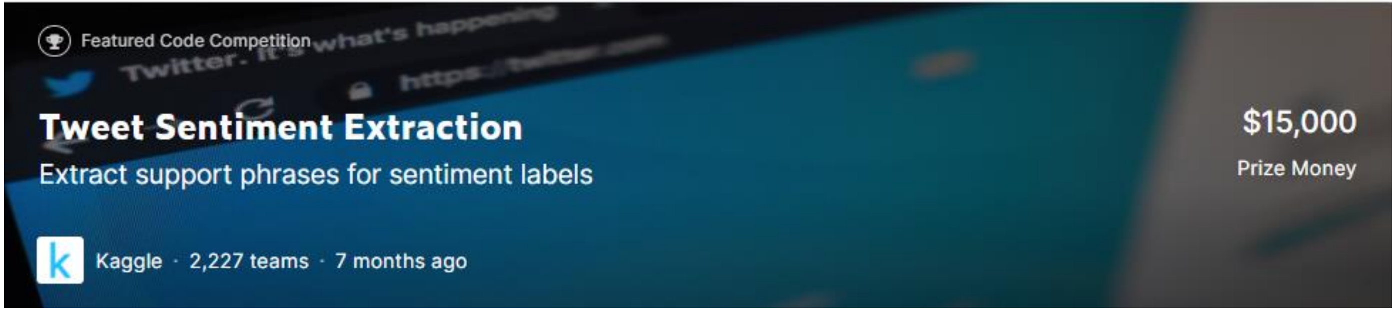
- 模型融合
- 更好的损失函数

Kaggle竞赛案例1



https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/discussion/103280#latest-619135

Kaggle竞赛案例2

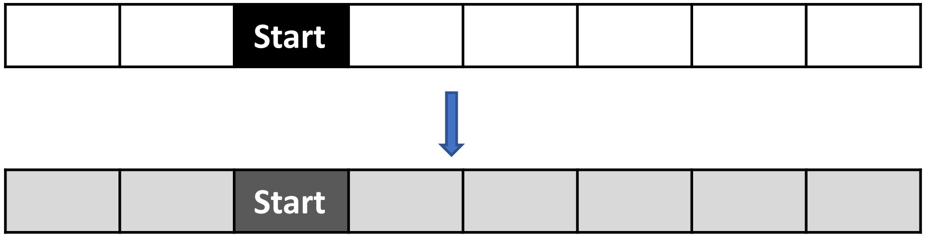


- You're attempting to predict the word or phrase from the tweet that exemplifies the provided sentiment.

- Sooo SAD I will miss you here in San Diego!!! [negative]

Kaggle竞赛案例2

- 典型的区间预测任务
  - 使用CNN来强化局部特征
  - Concat BERT的最后几层来获得更全面的语义表示
- 样本规模小，容易过拟合
  - FGM对抗训练
  - EDA数据增强
  - Freeze embedding
  - Label smoothing



https://github.com/thuwyh/Tweet-Sentiment-Extraction

谢谢

- 实验算力由HP提供
- 更多NLP竞赛、技术相关文章欢迎关注公众号