我删了这些训练数据…模型反而表现更好了！？

原创 Severus 夕小瑶的卖萌屋 2021-08-12 12:05

文 | Severus
编 | Sheryc_王苏

预训练语言模型的训练语料是全网数据，其来着不拒，只要喂过来的数据，统统吃掉，尽可能消化掉。而统计模型，除泛化能力外，另一个重要的能力就是**记忆能力**。

我们知道，~~人类的本质是复读机~~，啊，不是，全网数据中，重复或接近重复的数据是相当多的，尤其是数据爆炸的今天，当我们浏览各个来源的网络资讯的时候，时不时总会有似曾相识的感觉（当然这一定程度也归功于各大自媒体的洗稿）。这种重复的数据在统计模型的眼里，无疑是在告诉它，“*这是老师反复强调的东西，你要加强记忆啊！*”加强了记忆之后，对于理解模型来讲，就是在理解任务上的泛化性能会受限；生成模型中，则会出现逐字copy训练语料作为生成结果的现象。

所以我们可以看到，GitHub发布的Copilot出现了大段copy代码的问题，如果感兴趣的读者尝试了ERNIE3.0/GPT-3，也会发现有大量的生成结果看上去就是在copy训练语料。

所以，本文作者直接 **消除数据中的重复** ，去训练生成模型，最终发现复读机现象大幅减少，而且困惑度也都有所下降。

当然，实际上，去重预训练的语料，实际上也会使得最终的语言模型更加像是一个拥有通用语言知识的模型，而非记忆了部分事实的模型，在实际的应用中想要追求模型什么样的表现，事实上还是要权衡一下。

论文题目：

Deduplicating Training Data Makes Language Models Better

论文链接：

https://arxiv.org/pdf/2107.06499.pdf

Arxiv访问慢的小伙伴也可以在**【夕小瑶的卖萌屋】**订阅号后台回复关键词**【0812】**下载论文PDF~

找到重复数据

首先，什么样的数据是重复数据呢？

最直观的想法就是**部分重复**，即整段整句的复制。在训练样本，即字符串中，则是**一定长度的连续子串的重复**。所以，我们需要使用一些子串匹配算法，快速找到训练样本中重复的子串，这里我们就需要使用到**后缀数组**。

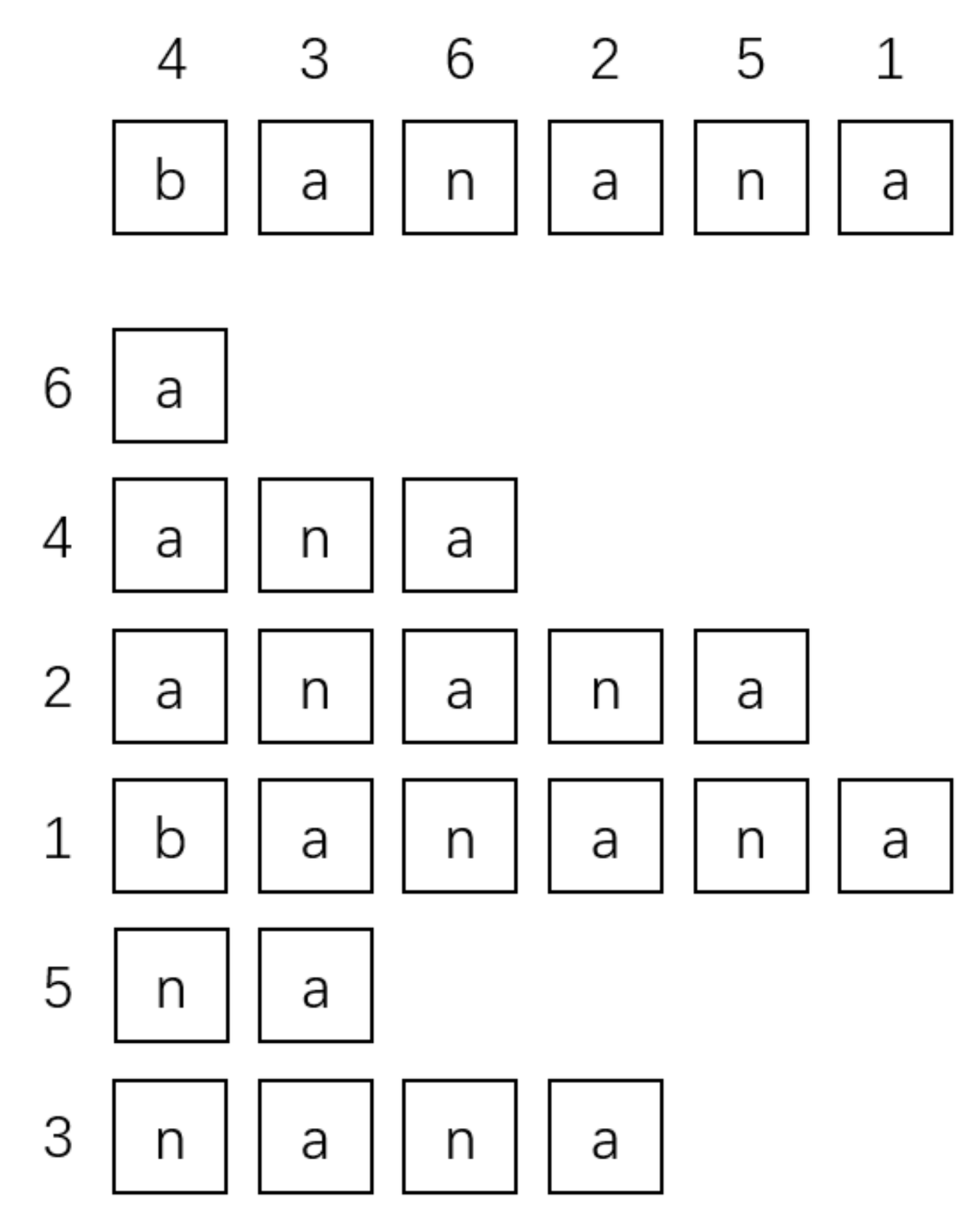
> 预训练语料往往是doc级别的，所以直接比较是不太可行的，毕竟哪怕是海量语料，精确相等的两篇文章也是很少的。

后缀数组

对于一个字符串$\mathcal{S}$，其后缀数组$\mathcal{A}_i$的定义为：*将$\mathcal{S}$的所有后缀按照字典顺序排序后，排名第i的后缀的起始位置的索引*，即：

$$\mathcal{A}(\mathcal{S}) = \arg\text{sort all\_suffixes}(\mathcal{S})$$

以单词banana为例，它的后缀为：["a", "na", "ana", "nana", "anana", "banana"]，排序后如下图：



可以看到，按照上文定义，字符串banana的后缀数组为： `[6, 4, 2, 1, 5, 3]` ，而通常我们也会给它配套一个*rank数组*，$\mathcal{R}_i$，表示*从字符串左数第i位起的后缀在后缀数组$\mathcal{A}$中的排名*。

构造后缀数组的方法有很多，例如直截了当直接对字符串的*后缀排序*（由于有字符串比较的时间，效率比较低），例如*倍增法*（易于实现且效率较好）。

除了这两个数组之外，要找到两个字符串的公共子串，我们还需要另外一个数组，通常称之为*高度（height）数组$\mathcal{H}$*。$\mathcal{H}_i$的定义为：*排名为i的后缀$\mathcal{A}_i$与后缀$\mathcal{A}_{i-1}$的最长公共前缀的长度*。例如上例中，后缀nana与后缀na的最长公共前缀为na，则$\mathcal{H}_6 = 2$。高度数组也可以用很高效的方法求取出来，本文则不再赘述。

可以看到，“**后缀的最长公共前缀**”实际上就是字符串中重复出现的连续子串了，而如果高度数组的取值在某个范围之内，则代表长度在某个范围之内的子串重复出现在字符串之中了。若将若干条字符串使用特殊符号连接，拼接到一起，则就可以得到多个字符串在某个长度范围内的重复子串，这样即可定位到**含有特定长度的重复子串的重复数据**，作者称之为**EXACTSUBSTR**。

作者也比较了不用的预训练数据集中，不同长度的子串的重复情况，如下图：

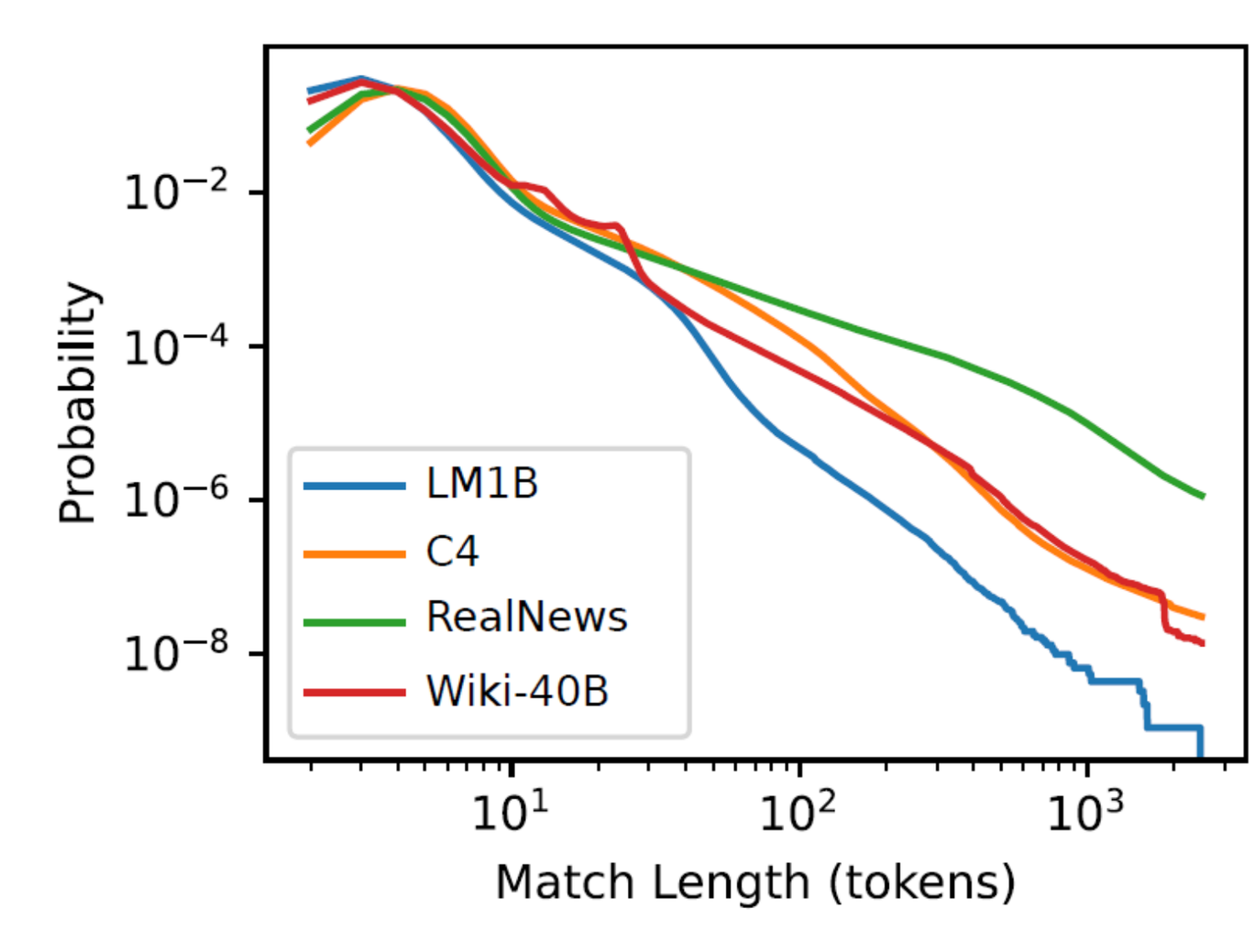


Figure 1: For each substring of length k, we plot the probability that there exists a second identical length-k substring in the same train set. Matches with length under 10 tokens are common, and account for 90% of tokens. We choose a threshold of 50 for experiments.

其中，长度为k的重复子串的概率计算如下：

$$m(k) = \Pr_{i\in[N]}[\exists i \neq j : \mathcal{S}_{i..i+k} = \mathcal{S}_{j..j+l}]$$

最终作者选择了50作为阈值，用以定位重复数据。

最小哈希（MinHash）

除精确的子串匹配之外，作者还使用了**相似性哈希算法MinHash**直接近似计算整个训练样本的相似度，作为精确字符串匹配的补充，作者称之为**NEARDUP**。对于两个输入样本x_i和x_j，其各自的n-gram集合d_i和d_j，则二者的相似度可以近似使用**Jaccard系数**计算，即：

$$\text{Jaccard}(d_i, d_j) = \frac{|d_i \cap d_j|}{|d_i \cup d_j|}$$

MinHash使用哈希函数将n-gram集合重排，*只保留最前的k个n-gram来计算文档的签名，用以计算文档的相似性*。本文选择了5-gram以及k=9000，用于计算文档签名，使用下式来计算文档的相似概率：

$$\Pr(d_i, d_j | \text{Jaccard}(d_i, d_j) = s_{i,j}) = 1 - (1 - s_{i,j}^b)^r$$

其中$b = 20, r = 450$是用户可以设置的超参。

作为补充，在使用MinHash计算潜在相似性之后，还可以使用**编辑相似度**来做进一步的过滤，编辑相似度定义如下：

$$\text{EditSim}(x_i, x_j) = 1 - \frac{\text{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)}$$

本文使用编辑相似度大于0.8来当作辅助判定。

当两个样本使用上述方法判定为相似之后，则将二者连边，最终*将语料集构造为一个图*，然后即可**使用图论算法，计算图中的连通分量，用以确认相似文档的聚簇，用于去重**。

作者也在C4数据集上使用NEARDUP分析了一下，可以看到最终的聚簇分布如下：

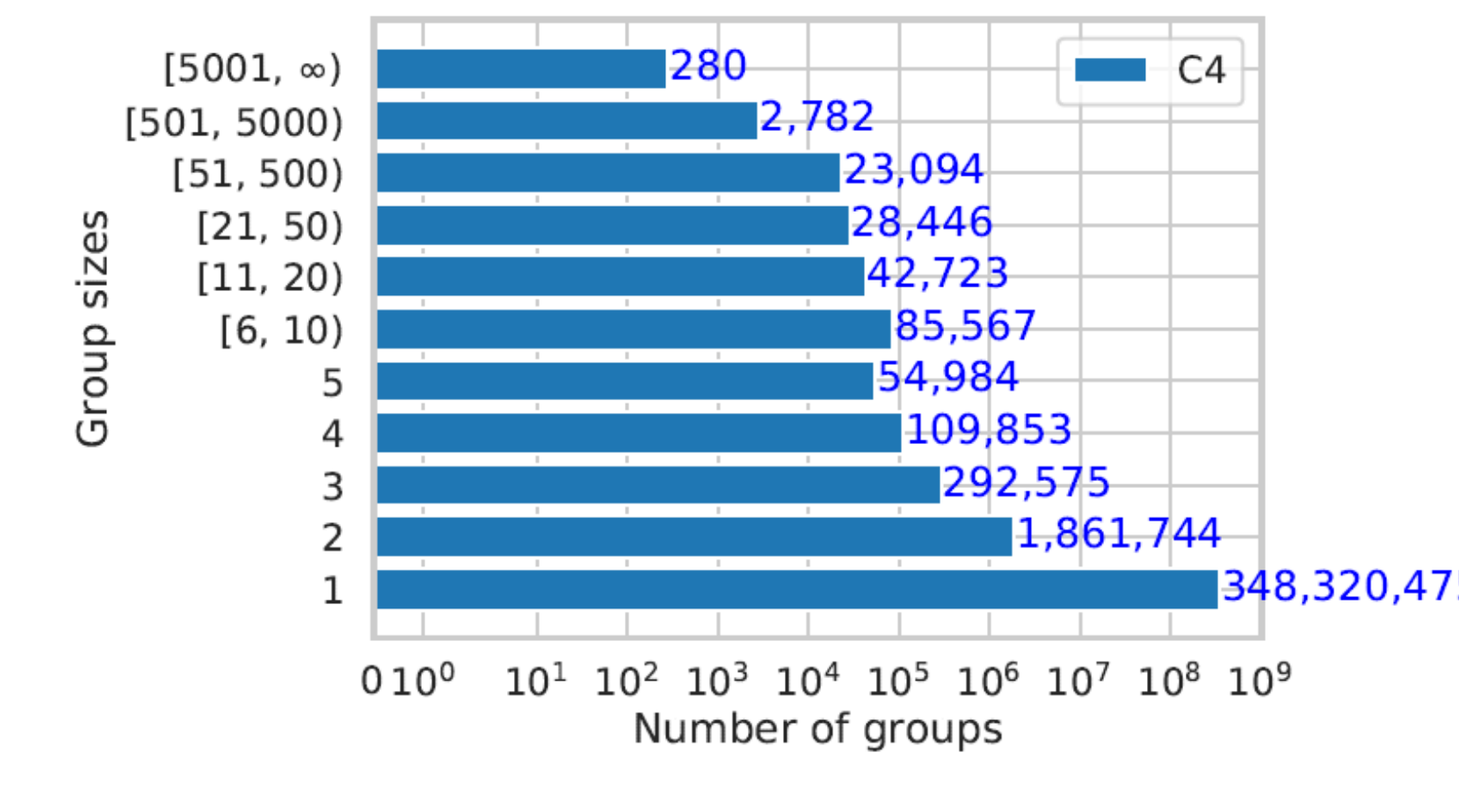


Figure 2: The distribution of near-duplicate cluster sizes from running NEARDUP on C4.

同时也用使用以上两种方法计算的重复case：

Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
RealNews	KUALA LUMPUR (Reuters) - Roads in Southeast Asia have been getting a little louder lately as motorcycle makers, an aspiring middle class and easy bank credit come together to breed a new genus of motorcyclists – the big-bike rider. [...]	A visitor looks at a Triumph motorcycle on display at the Indonesian International Motor Show in Jakarta September 19, 2014. REUTERS/Darren Whiteside\n KUALA LUMPUR (Reuters) - Roads in Southeast Asia have been getting a little [...] big-bike rider. [...]
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

Table 1: Qualitative examples of near-duplicates identified by NEARDUP from each dataset. The similarlity between documents is highlighted. Note the small interspersed differences that make exact duplicate matching less effective. Examples ending with “[...]” have been truncated for brevity.

以及在各个数据集上，使用上述两种方法分别计算出的重复文本的占比：

	% train examples with		% valid with
	dup in train	dup in valid	dup in train
C4	3.04%	1.59%	4.60%
Real News	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

Table 2: The fraction of examples identified by NEARDUP as near-duplicates.

	% train tokens with		% valid with
	dup in train	dup in valid	dup in train
C4	7.18%	0.75 %	1.38 %
Real News	19.4 %	2.61 %	3.37 %
LM1B	0.76%	0.016%	0.019%
Wiki40B	2.76%	0.52 %	0.67 %

Table 3: The fraction of tokens (note Table 2 reports the fraction of *examples*) identified by EXACTSUBSTR as part of an exact duplicate 50-token substring.

实验效果

作者分析过每个数据集之后，也按照比例删除了各个数据集里面的数据，细节本文不再赘述。作者分析发现，**重复的数据多数为互联网上的相同新闻或机器生成的数据**，而**某些短且相似的文本，精确字符串匹配则会定位不到**，重复文本的case可以见上面的case对照表。

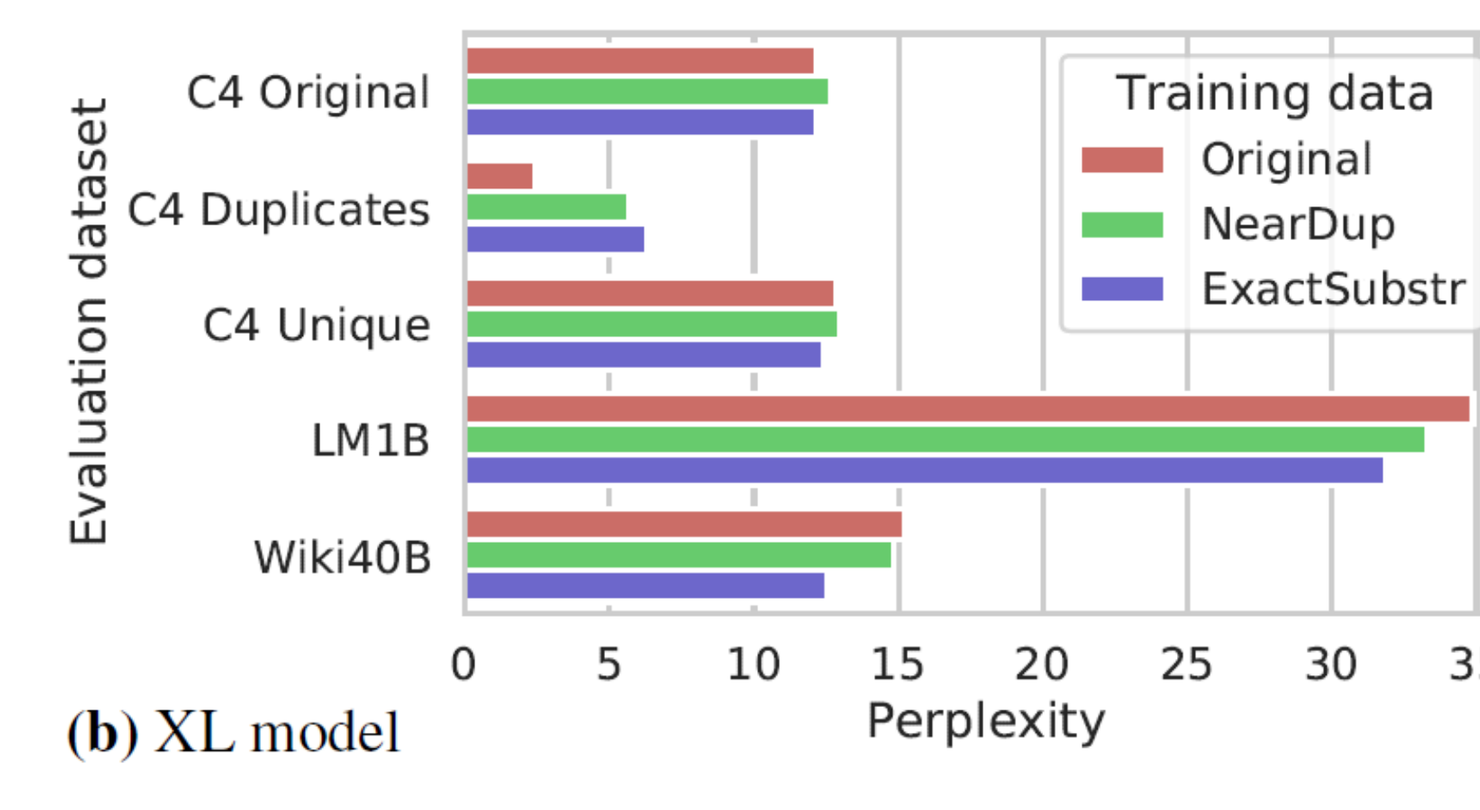
作者分别在下面3个数据集上训练了模型：

- C4-ORIGNAL：原本的C4数据集
- C4-NEARDUP：使用NEARDUP策略去重的数据集
- C4-EXACTSTR：使用EXACTSTR策略去重的数据集

分别训练了T5结构的XL模型，之后在4个评估集上验证困惑度：

- C4 Original：原本的C4数据集
- C4 Duplicates：使用NEARDUP计算出来的带有重复的子集
- C4 Unique：使用NEARDUP去重之后的子集
- LM1B：主要是新闻的句子
- Wiki40B：维基百科数据集

结果如下：



(b) XL model

可以看到，的确，*使用了去重之后，在LM1B及Wiki40B数据集上，困惑度的降低比较明显*，说明得到的语言模型能够适应更加广泛的文本，而自然**在重复的子集上困惑度会有所上升；在训练样本自身分布上，困惑度没有很明显的变化**。

作者也尝试了只用去重数据训练的模型在生成上的复读情况：

Model	1 Epoch	2 Epochs
XL-ORIGINAL	1.926%	1.571%
XL-NEARDUP	0.189%	0.264%
XL-EXACTSUBSTR	0.138%	0.168%

Table 4: When generating 100k sequences with no prompting, over 1% of the tokens emitted from a model trained on the original dataset are part of a 50-token long sequence copied directly from the training dataset. This drops to 0.1% for the deduplicated datasets.

可以看到，重复的比例也减少了很多。

最后，作者也看了一下，将验证集去重之后，对已有模型的困惑度的影响：

Model	Dataset	Orig	Dups	Unique
Transformer-XL	LM1B	21.77	10.11	23.58
GROVER-Base	RealNews	15.44	13.77	15.73
GROVER-XL	RealNews	9.15	7.68	9.45

Table 5: For each model, the perplexity of the official validation set (*Orig*), valid set examples which were identified by NEARDUP as matches of train set examples (*Dups*), and valid set examples identified by NEARDUP as unique (*Unique*). Due to the size of the RealNews validation set, we evaluated on only the first 25k examples meeting each condition.

也可以看到，**将验证集去重之后，已有模型在验证集的重复数据子集上困惑度相对低了一些，而在去重之后的验证集上，困惑度会相对偏高**，说明已有的语言模型也存在部分复读的情况。

发散地想一想

这篇文章从字面的角度上定位了训练样本中的重复数据，以及*去重之后，得到了泛化能力更好，并且在生成的时候不会整段地抄训练语料的语言模型*。那么既然有字面上的重复数据，我们自然也可以去发散，**是否可以从更加高层次的角度上，去定义数据重复呢**？

例如**主题上的重复**，比如通常，资讯中比较高频的是娱乐新闻，相对来讲其他的新闻占比就比较低；而2020~2021年，受新冠疫情影响，疫情相关的报道又一定多于其他的报道；而即便是百科数据，也有数据构造上的侧重，也有高质量词条和低质量词条的差别，这些主题上的相似不一定会表现在字面上，如使用自然分布去训练语言模型，最终的结果往往又会偏向这些热点事件。

又例如**通用知识上的重复**，或可称之为表达上的重复，在之前的文章（在错误的数据上，刷到SOTA 又有什么意义？）中，我提出，一个人，哪怕他不懂某一个领域，或者看一个从来没有见过的文本，虽然其中的专名他可能不知道是什么东西，但是他能大体看懂这段文本。也就是说，人与人之间，一定存在一个**共用的知识体系**，我将之称作是**通用知识**。而**如果抽取出文本中的通用知识，以通用知识的相似程度去定义文本重复，以此来去重数据，增加数据的多样性，以及使用对抗的方式剥离模型中的一些“事实”知识，是否能够让模型更加的泛化呢**？

那么，有没有一个工具，可以帮助我们实现找到文本中的通用知识呢？

实际上，今年百度开源的项目——**PaddleNLP - 解语**[1]就是在文本与通用知识关联上的一个尝试，通过定义语义词类的方式，将中文文本转换为词类知识序列，为上文提到的利用通用知识来聚合文本提供了一层丰富、覆盖全面且相对稳定的特征（Beyond 预训练语言模型，NLP还需要什么样的知识？）。或许，通过这种方式，在今天的主题上，也可以完成一种延续。

当然，我们也希望，在这个方向上能够有更多的工作，最终的理想是，让算法对文本的理解能力逼近人类。

参考文献

[1].PaddleNLP - 解语. https://www.paddlepaddle.org.cn/textToKnowledge