论语言编码器对语法错误的鲁棒性。

**尹范**1**，龙泉宇**2**，孟涛**3**，张凯伟**3 1北京大学。

2上海交通大学3加州大学洛杉矶分校1600012975@pku.edu.cn；oscar.long@sjtu.edu.cn；tmeng@cs.ucla.edu；[kwchang@cs.ucla.edu。](mailto:kwchang@cs.ucla.edu)

# 摘要

我们进行了一项彻底的研究，以诊断预训练的语言编码器（ELMo、BERT和RoBERTa）在面对自然语法错误时的行为。具体来说，我们从非母语者那里收集真实的语法错误，并进行广告攻击，在干净的文本数据上模拟这些错误。我们使用这种方法来对下游应用程序进行调试模型。结果表明，所有测试模型的性能都会受到影响，但影响的程度有所不同。为了解释模型behav- iors，我们进一步设计了一个语言学接受性任务，以揭示他们在识别非语法句子和错误位置的能力。我们发现，带有简单分类器的ﬁxed上下文编码器在句子正确性的预测上进行了训练，能够定位错误位置。我们还为BERT设计了一个loze测试，并发现BERT捕捉到了上下文中错误和speciﬁc tokens之间的in- teraction。我们的结果为理解- ing语言编码器对语法错误的鲁棒性和行为提供了启示。

# 介紹

预训练的语言编码器在促进各种下游自然语言处理（NLP）任务方面取得了巨大的成功（[Peters等](#_bookmark36)，[2018](#_bookmark36)；[Devlin等](#_bookmark24)，[2019](#_bookmark24)；[Liu等](#_bookmark33)，[2019b](#_bookmark33)）。How- ever, they usually assume training and test cor- pora are clean and it is unclear how the models behave when confronted with noisy input.Gram- matical error是一种重要的噪声类型，因为它自然而频繁地发生在自然语言中，特别是在来自非母语者的口语和书面材料中。处理这样的噪声会重新影响模型在表示语言和语法知识时的鲁棒性。如果语言编码器能处理这些噪音，也会产生积极的社会影响。

可以适当地模拟非母语者的文本。

最近关于评估模型针对语法错误的行为的工作采用了各种meth- ods，包括（1）在speciﬁc语言表象- ena上手动构建mini- mal编辑对（[Marvin和Linzen](#_bookmark35)，[2018](#_bookmark35)；[Goldberg](#_bookmark26)，[2019](#_bookmark26)；[Warstadt等人，2019a，b）；（2）标记或创建可接受性判断资源（Linzen等人，2016；Warstadt和Bowman，2019；Warstadt等人。](#_bookmark50)，[2019a](#_bookmark50),[b](#_bookmark51)）；（2）标记或创建可接受性判断资源（[Linzen等人](#_bookmark31)，[2016](#_bookmark31)；[Warstadt和Bowman](#_bookmark49)，[2019](#_bookmark49)；[Warstadt等人](#_bookmark50)，[2019a](#_bookmark50)）；（3）为speciﬁc NLP任务模拟噪声，如神经机器翻译（[Lui等人](#_bookmark34)，[2018](#_bookmark34)；[Anastasopoulos](#_bookmark13)，[2019](#_bookmark13)），情感clas- siﬁcation（[Baldwin等人](#_bookmark14)，[2017](#_bookmark14)）。这些研究要么专注于speciﬁc现象，主要在指定的语料库上进行实验，要么严重依赖人类注释和语言学的专家knowl- edge。相比之下，我们的工作自动模拟自然发生的数据和各种类型的语法错误，并系统地分析这些噪音如何影响下游的应用。这对于理解几种语言编码器对语法错误的鲁棒性具有更多的实际意义。

具体来说，我们首先提出了一个有效的方法来模拟不同的语法错误，它应用了基于NUS Cor- pus of Learner English (NUCLE)([Dahlmeieretal.](#_bookmark22) , [2013](#_bookmark22))上观察到的真实错误的黑盒对抗性攻击算法，这是一个语法错误纠正基准。这种方法将干净的语料库转化为有错误的语料库，有利于调试下游任务中的语言编纂者。我们通过评估四个语言理解任务和一个序列标记任务上的模型来证明其灵活性。

接下来，我们通过语言接受能力任务探究预训练编码器的各个层级，来量化模型识别- ing语法错误的能力。我们为八种错误类型分别构建了数据集。然后，我们冻结编码器层

3386

*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3386-3403 July 5 - 10, 2020.Oc 2020计算语言学协会。

并在每层 位置。这个探测任务假设如果一个简单的分类器在指定的错误类型上表现良好，那么编码器层很可能包含该错误的知识([Conneauetal.](#_bookmark20) , [2017](#_bookmark20); [Adi et al.](#_bookmark11) , [2017](#_bookmark11))。

最后，我们研究了模型如何捕捉语法错误和con-文本之间的相互作用。我们以BERT为例，设计了一个无监督的cloze测试来评估其作为掩蔽语言模型（MLM）的内在功能。

我们的贡献总结如下：

* 1. 我们提出了一种模拟各种语法错误的新方法。所提出的方法是可行的，可以用来验证语言编码器对语法错误的鲁棒性。
  2. 我们通过研究模型在各种语法错误类型的下游任务上的表现，对语言编码器的ro- bustness进行了系统分析，并加强了之前的工作。
  3. 我们证明。(1)现有的语言编码器对语法错误的鲁棒性是不同的；(2)语言编码器的上下文层在识别和定位语法错误方面的能力要强于标记嵌入层；(3)BERT捕捉到了错误和上下文中speciﬁc标记之间的相互作用，特别是错误的邻近标记。

重现我们实验的代码可在以下网站获取[：https://github.com/uclanlp/ ProbeGrammarRobustness。](https://github.com/uclanlp/ProbeGrammarRobustness)

# 相关工作

**探究预训练的**语言编码器 最近，预训练的语言编码器在不同的下游任务中取得了成功，激发了人们了解其优势的巨大兴趣。过去分析预训练编码器的工作主要是基于干净的数据。正如[Tenney等人](#_bookmark44)（[2019a](#_bookmark44)）中提到的，这些stud- ies可以大致分为两类。(1)设计受控任务，以探究一个spe- ciﬁc语言现象是否被模型捕获([Conneau等人](#_bookmark21)，[2018](#_bookmark21)；[Peters等人](#_bookmark37)，[2019](#_bookmark37)；[Tenney等人](#_bookmark45)，[2019b](#_bookmark45)；[Liu等人](#_bookmark32)，[2019a](#_bookmark32)；[Kim等人](#_bookmark29)，[2019](#_bookmark29))，或者。

(2)分解模型结构，探索编码的语言属性是什么([Tenneyetal](#_bookmark44) ,

[2019a](#_bookmark44)；[Jawahar等人](#_bookmark27)，[2019](#_bookmark27)；[Clark等人](#_bookmark19)，[2019](#_bookmark19)）。)然而，这些研究并没有分析gram- matical误差如何影响模型行为。

我们的工作与分析人工创建噪声的mod- els的研究有关。例如，[Linzen等人](#_bookmark31)（[2016](#_bookmark31)）评估LSTMs是否通过使用verbal inﬂection违反主语-动词协议来捕获语言的层次结构。[Marvin和Linzen](#_bookmark35)([2018](#_bookmark35))提出了一个新的数据集，由最小编辑对与oppo- site语言接受性组成的三个speciﬁc lin- guistic现象，并使用它来评估RNN的句法能力。[Goldberg](#_bookmark26)（[2019](#_bookmark26)）调整了之前的方法来评价BERT。[Warstadt等人](#_bookmark50)（[2019a](#_bookmark50)）进一步比较了sin- gle现象下的五种分析方法。尽管方法的多样性- ogy，这些研究有共同的局限性。首先，他们只采用了语言知识的单一或speciﬁc方面；其次，他们的实验主要基于构建的数据集，而不是现实世界的下游应用。相比之下，我们提出了一种覆盖更广泛的语法错误的方法，并对下游任务进行评估。一项同时进行的工作（[Warstadt 等人](#_bookmark51) ，[2019b](#_bookmark51)）通过使用语言学家制作的模板，为英语中 67 个孤立的语法范式创建- ing 语言最小对数据集，为诊断语言模型提供便利。相比之下，我们并不严重依赖人工词汇和模板。

**合成误差** 为了评估和促进神经模型对噪声的鲁棒性，一些研究手动创建了具有speciﬁc语言现象的新数据集（[Linzen等](#_bookmark31)，[2016](#_bookmark31)；[Marvin和Linzen](#_bookmark35)，[2018](#_bookmark35)；[Goldberg](#_bookmark26)，[2019](#_bookmark26)；[Warstadt等](#_bookmark50)，[2019a](#_bookmark50)）。其他人已经引入了各种方法来在干净的下游数据集上生成合成错误，特别是机器翻译语体。[Belinkov和Bisk](#_bookmark15)（[2018](#_bookmark15)）；[Anastasopoulos](#_bookmark13)（[2019](#_bookmark13)）证明，由字符操作和单词substitu- tion合成的语法错误会降低NMT系统的性能。[Baldwin等人](#_bookmark14)([2017](#_bookmark14))用句法上(重新排序- ing)和语义上(单词替换)的噪声句子来增强原始情感分类数据集，并实现了更高的性能。我们的方法部分受到[Lui等人](#_bookmark34)（[2018](#_bookmark34)）的启发，他们通过维护五个简单错误类型的混淆矩阵来合成半自然的非语法句子。

另一线研究使用黑盒对抗性攻击方法来创建对抗性的例子。

用于调试 NLP 模型（[Ribeiro 等人](#_bookmark40)，[2018](#_bookmark40)；[Jin等人](#_bookmark28)，[2019](#_bookmark28)；[Alzantot等人](#_bookmark12)，[2018](#_bookmark12)；[Burstein等人](#_bookmark17)，[2019](#_bookmark17)）。与上述数据生成程序相比，这些方法为目标模型创造了更具挑战性的场景。我们提出的模拟受益于对抗性攻击算法和半自然语法错误。

# 方法

我们首先解释我们如何模拟非语法场景。然后，我们描述目标模型和评估设计。

## 语法错误模拟

大多数下游数据集只包含干净和语法句子。尽管最近的lan-guage编码器取得了很好的性能，但目前还不清楚它们是否在有语法错误的文本数据上同样表现良好。

因此，我们在干净的语料库上合成语法错误，以测试语言编码器的鲁棒性。我们使用一种可控的基于规则的方法来收集和模仿在NUCLE上观察到的错误，遵循之前的工作（[Lui等人](#_bookmark34)，[2018](#_bookmark34)；[Sperber等人](#_bookmark43)，[2017](#_bookmark43)），并应用两种方法将er- rors引入清洁语料库。(1)我们根据NUCLE的频率分布对错误进行采样，并将其引入到合理的位置；(2)受对抗性攻击的文献([Ribeiro等人](#_bookmark40)，[2018](#_bookmark40)；[Jin等人](#_bookmark28)，[2019](#_bookmark28)；[Alzantot等人](#_bookmark12)，[2018](#_bookmark12))的启发，我们进行搜索算法，引入在给定下游任务上造成最大性能下降的语法错误。

**模仿NUCLE上的错误**分布 我们首先描述如何提取NUCLE上的错误分布（[Dahlmeier等人](#_bookmark22)，[2013](#_bookmark22)）。NUCLE是由自然发生的数据（NUS的学生论文）和错误标签组成的。每一个不符合语法的句子都与其仅在局部编辑中不同的修正配对。这两个句子组成了一个*最小的编辑对*。一个例子是像。

* + 1. 孩子**长大**后会不会责怪父母？

*×*

* + 1. 孩子会不会责怪父母后，他

**长大了**？√

NUCLE语料库包含约59,800个句子。

平均长度为20.38。每个句子中约有6%的标记包含语法错误。共有27个错误标签，包括Prep（表示prepo- sition错误）、ArtOrDet（表示article或de- terminer错误）、Vform（表示动词不正确）、Vform（表示动词不正确）。

形式）等等。

我们考虑了NUCLE中8种经常发生的、令牌级的错误类型，如表[1](#_bookmark1)所示。

这些错误类型从句法（SVA，Worder）、语义（Nn，Wchoice，Trans）和两者（ArtOrDet，Prep，Vform）方面扰动一个句子，因此涵盖了自然语言中广泛的噪声。然后，我们根据对NUCLE的观察，为每一种错误类型构建一个混淆集。混淆集的每个成员都是一个token。我们在同一集合中的token *ti*和*tj*之间分配一个权重*wij*来表示*ti*被*tj*替换的概率。特别是，对于 ArtOrDet、Prep 和 Trans，混淆集由一组在 NUCLE 上经常出现错误或更正的令牌组成。对于集合中的每个标记 *ti*，我们根据在 NUCLE 上最小编辑对中 *ti 被 tj* 取代的次数来计算 *wij*。

请注意，我们添加了一个特殊的标记ø来重新发送删除和插入。对于 Nn，当我们定义一个名词时，我们将它和它的单数 (SG) 或复数 (PL) 对应的名词添加到集合中。对于SVA，当我们找到一个现在时态的动词时，我们将它和它的第三人称-单数(3SG)或非第三人称(非3SG)对应词添加到集合中。对于Worder，我们将副词的位置与其邻近的动词、分词或情态交换。对于Vform，我们使用NLTK([Bird and Loper](#_bookmark16), [2004](#_bookmark16))来提取动词的现在、过去、进行时和完成时，并添加到集合中。对于Wchoice，我们从WordNet中选择十个目标词的同义词。Vform和Wchoice的subi- tution权重都设置为统一。

**语法错误介绍** 我们以两种方式介绍错误。ﬁrst称为*proba- bilistic转换*。与[Lui等人](#_bookmark34)（[2018](#_bookmark34)）类似，我们ﬁrst使用伯克利句法分析器（[Petrov等人](#_bookmark38)，[2006](#_bookmark38)）获得目标句子的解析树。然后，我们从NUCLE估计的错误类型分布中抽取一个错误类型，并根据解析树随机选择一个可以应用这种错误类型的位置。最后，我们根据采样错误类型的混淆集的权重来采样一个错误标记，并将错误标记引入到所选位置。

然而，*概率变换*只代表了平均情况。为了调试和分析语言编码器的鲁棒性，我们考虑了另一个更具挑战性的环境--*最坏情况下的转换*，在这里我们利用搜索算法

## type 误差 Description 错乱设置

ArtOrDetArticle/determiner errors*{* a, an, the, ø*}。*

*{* on, in, at, from, for, under, over, with, into,

PrepPreposition错误

TransLink单词/短语错误

期间，直到，反对，之间，整个，到，由，关于，像，之前，跨，后面，但，出，向上，后，自，向下，关闭，的，Ø

而且，但是，所以，然而，作为，那，因此，也，b-原因，因此，如果，虽然，哪，哪里，此外，除了，的，ø*}。*

*{*

*}*

Nn名词数量错误*{*SG, PL*}。*

SVAS主语-动词协议错误*{*3SG，不是3SG*}。*

Vform动词形式错误*{*现在、过去、渐进、完美*}。*Wchoice单词选择错误*{*来自WordNet Synsets的十个同义词*}。*WorderWord定位错误*{*形容词/形容词、分词、情态*}。*

表1：目标错误类型及相应的混淆集。

黑盒对抗式攻击的结果来确定错误位置。更具体地说，我们通过考虑基于所有混淆集的所有可能的替换来获得句子中每个标记的操作集。请注意，有些混淆集是不适用的，例如Nn对动词的混淆集。操作集中的每个操作都是为了替换目标标记或改变其位置。然后，我们采用一种搜索算法，从这些操作集中选择改变测试模型预测的操作，并应用它们来生成错误句。我们考虑了三种搜索算法：*贪婪搜索*、*波束搜索*和*遗传算法*。

*贪婪搜索*攻击是一个两步程序。首先，我们评估一个句子中标记的重要性。当一个标记被删除时，它的重要性由模型预测的似然下降来表示。降低的幅度越大，标记就越重要。在对所有标记进行比较后，我们得到一个按重要性降序排列的标记列表。然后，我们走过这个列表。对于列表中的每个标记，我们尝试与该标记相关联的操作集中的所有操作，然后练习降低模型预测可能性最大的操作。我们不断重复第二步，直到预测结果发生变化或达到预算（例如，每句话的操作数量）。

*波束搜索*与*贪婪搜索*类似。唯一不同的是，当我们走过排序后的tokens列表时，我们会保持一个大小为*k*的波束，其中包含全局降级最高的前*k个*操作流。

*遗传算法*是一种基于群体的迭代方法，用于寻找更合适的例子。我们首先通过随机选择操作来建立一代，然后使用交叉和突变的组合来寻找更好的候选者。我们向读者推荐[Alzantot等人](#_bookmark12)（[2018](#_bookmark12)），以了解对抗性攻击中遗传算法的细节。所有方法的Com- prehensive描述见附录[C](#_bookmark58)。

## 目标机型

我们评估了以下三种预训练的lan-guage编码器。模型和训练设置的详细描述见附录[B](#_bookmark55)。

**ELMo**（[Peters et al.](#_bookmark36) , [2018](#_bookmark36)）是一个基于LSTM的三层模型，在1B Word Benchmark（[Chelbaetal.](#_bookmark18) , [2014](#_bookmark18)）的双向语言建模任务上进行预训练。我们将ELMo作为上下文嵌入，并在其上添加两层带有注意力机制的BiLSTM。

**BERT** ([Devlin et al.](#_bookmark24) , [2019](#_bookmark24))是一个基于变压器的([Vaswaniet al.](#_bookmark46) , [2017](#_bookmark46))模型，它在掩蔽语言建模和下句预词典任务上进行了预训练。它使用16GB英语文本，并通过ﬁne-tuning适应下游任务。We use *BERT- base-cased* for Named Entity Recognition (NER) and *BERT-base-uncased* for other tasks and per- form task-specific c ﬁne-tuning.

**RoBERTa** ([Liu et al.](#_bookmark33) , [2019b](#_bookmark33))是一个稳健的预训练BERT模型，它使用了更大的预训练数据(共160GB)、更长的预训练时间、dy- namic masking策略和其他优化的预。

训练方法。我们使用*RoBERTa-base*和每一个形式的任务规格调整。

## 评价方法

我们设计了以下三种评价方法来系统分析语言编码器在输入中受到语法错误的影响。

**模拟下游任务上的错误** 使用第 [3.1](#_bookmark0) 节中讨论的模拟方法，我们能够对现有的基准体进行评估。在我们的实验中，我们独立考虑目标模型。整个过程是：给定一个数据集，目标模型被ﬁrst训练（ﬁne-tuned），并在干净的训练和开发集上进行评估。然后，我们从开发集中丢弃那些错误预测的例子，并应用模拟方法来扰动每个剩余的例子。我们计算了攻击成功率（被攻击的例子/所有例子）作为模型对语法错误的鲁棒性指标。攻击成功率越小，模型越健壮。

*§*

**语言学可接受性探测** 我们设计了一个语言学可接受性探测任务来评估每一个单独类型的错误。我们考虑两个方面。(1)如果模型能够判断一个句子在语法上是否正确(即二元分类任务)；(2)如果模型能够在token层面定位错误位置。我们对目标模型进行编码，并训练一个自注意力分类器来执行这两个探测任务。

**BERT的cloze测试** 我们设计了一个无监督的cloze测试来评估基于最小编辑对的BERT的屏蔽语言模型组件。对于每一个只在一个方面有差异的最小编辑对，我们量化了在句子的其余部分预测一个掩蔽标记的概率如何受到这个语法错误的影响。这种方法分析了错误token如何影响干净的语境，这与[Goldberg](#_bookmark26)（[2019](#_bookmark26)）的观点相辅相成，他专注于SVA错误，并讨论了干净的语境如何影响被掩盖的错误token的预测。

# 语法错误如何影响下游性能？

在本节中，我们模拟语法错误并分析下游任务的性能下降。

我们比较了ELMo、BERT、RoBERTa和一个基线模型InferSent（[Conneau等](#_bookmark20)，[2017](#_bookmark20)）。

Infersent ELMo BERT RoBERTa

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MRPC | 75.42 | 80.30 | 86.48 | 89.88 |
| MNLI-m | 68.62 | 74.91 | 83.77 | 87.70 |
| MNLI-mm | 69.12 | 75.50 | 84.80 | 87.40 |
| QNLI | 77.39 | 78.23 | 90.58 | 92.50 |
| SST-2 | 83.14 | 90.37 | 92.08 | 94.72 |
| NER | - | 91.21 | 95.20 | 95.45 |

表2.目标模型在语言理解和顺序标签任务上的原始性能。目标模型在语言理解和顺序标记任务上的原始性能。

**数据集** 我们使用四个语言理解数据集。MRPC（[Dolan和Brockett](#_bookmark25)，[2005](#_bookmark25)），MNLI（[Williams等人](#_bookmark52)，[2018](#_bookmark52)），QNLI（[Rajpurkar等人](#_bookmark39)，[2016](#_bookmark39)）和SST-2（[Socher等人](#_bookmark42)，[2013](#_bookmark42)）来自GLUE（[Wang等人](#_bookmark47)，[2019a](#_bookmark47)）和序列标记基准。NER的CoNLL-2013。这些语料库的详细de- scriptions在附录[A](#_bookmark54)中。我们不使用GLUE的其他数据集，因为它们要么规模小，要么只包含短句。

**攻击设置** 对于所有的任务，我们将句子中允许修改的最大百分比限制为15%的代币，根据真实数据估计的统计数据，这是一个合理的比例。如表[3](#_bookmark3)所示，在这样的限制下，*最坏情况下的转换*只修改了约9%的tokens。对于MNLI和QNLI，我们只修改第二句，即分别修改假设和答案。对于MRPC，我们只修改第五句话。由于MNLI和QNLI的开发集中的例子相对较多，诱导了极长的攻击时间，因此我们不对MNLI和QNLI应用遗传算法。对于NER，我们保留命名实体，只修改剩余的tokens。

**结果和讨论**表[2](#_bookmark2)列出了四个目标模型在每个任务的标准开发集的测试性能。表[3](#_bookmark3)总结了语言理解任务上的攻击成功率，NER上F1得分的下降，以及修改后的代币的平均百分比（括号内的数字）。所有的数字都以百分比格式化。如表[3](#_bookmark3)所示，随着*概率论*的*形成*，攻击成功率下降2%（RoBERTa，QNLI）和10%（ELMo，MRPC）之间。随着

*最坏情况下*的*转换*，我们得到了81.1%的高攻击率（ELMo，遗传算法，MRPC），而所有任务的平均攻击率为29%，只扰动了大约9%的标记。这个结果证实了所有的模型都受到了非语法输入的影响。NER任务是

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **型号** | **Alg.** | **MRPC** | **MNLI (m/mm)** | **QNLI** | **SST-2** | **NER** |
| Infersent | 遥远的 | 6.51 (14.53) | 8.30 (13.98) / 8.80 (14.23) | 4.76 (12.53) | 5.79 (14.38) | - |
|  | 贪婪 | 53.42 (9.02) | 36.52 (10.35) / 40.71 (10.06) | 44.92 (7.61) | 43.44 (8.02) | - |
|  | 梁 | 54.39 (9.08) | 36.66 (10.37) / 40.87 (10.06) | 45.16 (7.62) | 43.86 (8.03) | - |
|  | 基因 | 79.15 (8.60) | - | - | 59.86 (8.39) | - |
| BiLSTM | 遥远的 | 9.99 (14.53) | 7.76 (13.98) / 7.83 (14.23) | 5.34 (12.53) | 4.64 (14.38) | 3.29 (13.75) |
| + ELMo | 贪婪 | 60.84 (8.19) | 29.58 (10.28) / 32.92 (9.89) | 39.12 (7.25) | 37.55 (8.24) | 17.81 (7.67) |
| + Attn | 梁 | 61.49 (8.29) | 29.74 (10.29) / 33.12 (9.91) | 40.38 (7.33) | 38.32 (8.32) | 18.33 (7.85) |
|  | 基因 | 81.14 (7.41) | - | - | 59.25 (8.25) | 39.78 (8.19) |
| BERT | 遥远的 | 3.69(14.53) | 6.59 (13.98) / 6.95 (14.23) | 2.33 (12.53) | 4.73 (14.38) | 3.07 (13.75) |
|  | 贪婪 | 31.25 (7.95) | 28.76 (10.28) / 32.04 (10.01) | 25.43 (7.38) | 33.54 (7.96) | 17.12 (7.51) |
|  | 梁 | 31.81 (8.01) | 29.03 (10.30) / 32.44 (10.04) | 26.42 (7.48) | 34.28 (8.01) | 18.27 (7.74) |
|  | 基因 | 59.01 (8.84) | - | - | 58.53 (7.83) | 38.83(7.64) |
| RoBERTa | 遥远的 | 3.04 (14.53) | 5.66 (13.98) / 5.88(14.23) | 1.92 (12.53) | 3.53 (14.38) | 2.52 (13.75) |
|  | 贪婪 | 20.45 (8.11) | 20.65 (10.43) / 21.47 (10.02) | 19.82 (7.18) | 31.06 (8.20) | 15.84 (8.12) |
|  | 梁 | 20.73(8.14) | 20.89 (10.44) / 21.91 (10.06) | 20.52 (7.29) | 31.91 (8.27) | 16.51 (7.47) |
|  | 基因 | 38.93 (9.17) | - | - | 56.41 (8.39) | 35.11(7.55) |

表3.模型在下游任务上的鲁棒性评价结果。评估模型在下游任务上的鲁棒性的结果。每一列代表一个数据集，每一行代表一个带有攻击算法的受害者模型（dist.表示*概率转换*）。在每个单元中，我们显示了数据集上的平均攻击成功率（百分比）和修改词的平均百分比（括号中的数字）。

BERTRoBERTa

MRPC MNLI SST MRPC MNLI SST

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **准备工作** | 16 | 178 | 36 | 15 | 103 | 43 |
| **艺术/设计** | 5 | 270 | 20 | 7 | 228 | 28 |
| **Wchoice** | 93 | 1129 | 233 | 64 | 772 | 195 |
| **Vform** | 8 | 231 | 26 | 9 | 314 | 37 |
| **SVA** | 57 | 538 | 83 | 31 | 388 | 83 |
| **Nn** | 14 | 128 | 13 | 3 | 84 | 13 |
| **Worder** | 0 | 62 | 28 | 0 | 43 | 28 |
| **Trans** | 5 | 70 | 25 | 5 | 31 | 25 |

表4：每个错误类型在成功攻击中被选择的次数。我们发现，Wchoice和SVA的危害性更大。

一般来说，较难受到语法错误的影响。在*概率转换*方面，F1分数的下降幅度从2%到4%不等。对于*最坏情况下的转换*，NER的最高下降幅度为18.33%（ElMo，波束搜索）。

考虑到不同的目标模型，我们发现语法错误对不同模型的影响有所不同。特别是在MRPC和MNLI上，RoBERTa对语法错误的影响表现出很强的鲁棒性，在所有的任务中，尤其是在MRPC和MNLI上，攻击成功率（平均20.28%）和F1得分去掉率（平均17.50%）一直较低。另一方面，BERT、ELMo和InferSent在NLU任务上的平均年龄攻击率分别为26.03%、33.06%、36.07%重新谱化。鉴于预训练策略的差异，我们推测预

用更多的数据进行训练，可能会对噪声数据的模型 ro- bustness 有好处。这个推测与（[Warstadt等人](#_bookmark51) ，[2019b](#_bookmark51)）一致，作者也在 LSTM 和 Transformer-XL（[Dai等人](#_bookmark23) ，[2019](#_bookmark23)）上给出了不同训练数据的轻量级演示。我们将这一推测的进一步利用- ration和模型架构的详细分析留给未来的工作。

需要注意的是，在实验环境中，对于每个模型，我们按照文献中的说法，只计算模型做出正确预测的情况下的at- tack成功率。因此，不同模型的攻击成功率不具有可比性。为了比较不同编码器的鲁棒性，我们进一步检查了所有模型做出正确预测的开发集中共同部分的攻击成功率。我们发现总体趋势与表中的趋势相似

1. 例如，RoBERTa、BERT、ELMo在MRPC和SST- 2上的贪婪攻击成功率分别为14.4%、22.1%、46.8%和28.2%、30.0%、33.9%。

为了更好地理解语法错误的影响，我们还分析了(1)哪种错误类型对性能的危害最大，(2)不同的错误率如何影响性能。对于第五个问题，我们用成功的贪婪攻击实例中选择的总时间来表示错误类型的危害。我们在MRPC、MNLI-m和SST-2的开发集上进行实验来分析BERT和RoBERTa，如表所示。

**句子级 Acc 代币级 Acc**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  | InferSent  ELMo |  |
|  |  |  |  |  | BERT |  |
|  |  |  |  |  | RoBERTa |  |

0.8

攻击成功率

0.7

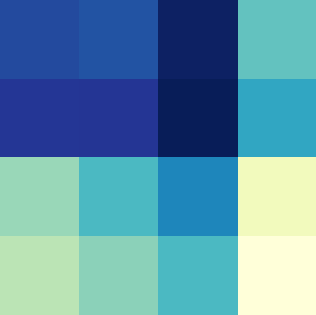
0.6

0.5

0.4

0.3

**第十二层**

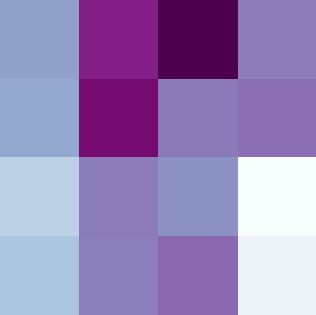
**第八层**

|  |  |  |  |
| --- | --- | --- | --- |
| 85.6 | 84.3 | 93.3 | 68.1 |
| 88.3 | 88.4 | 94.5 | 73.8 |
| 63.4 | 70.3 | 78.4 | 52.2 |
| 60.8 | 64.6 | 70.3 | 48.2 |

**第一层**

**0层**

**第十二层**

**第九层**

|  |  |  |  |
| --- | --- | --- | --- |
| 50.5 | 79.8 | 93.2 | 58.8 |
| 48.0 | 85.3 | 59.6 | 62.2 |
| 33.9 | 59.3 | 54.4 | 13.1 |
| 39.1 | 58.7 | 63.6 | 17.5 |

**第一层**

**0层**

0.12 0.14 0.16 0.18 0.20 0.22

扰动率

图1：当一个句子中被修改的代币数量增加时，攻击成功率。

1. 其中，Wchoice是危害最大的类型，而Worder是危害最小的类型。SVA 排名第二，是最有害的类型。请注意，虽然Nn改变一个token的方式与SVA类似（在大多数情况下都是增加或删除-s或-es），但它们对模型有不同的影响。至于与函数词有关的错误，一般来说，Prep扮演着更重要的角色，但ArtOrDet对MNLI的危害更大。

对于第二种，我们将贪婪攻击的允许修改的代币从一句话中的15%增加到45%，导致修改代币的实际百分比-年龄低于20%。我们在MNLI-m的开发集上评估所有模型。Re-sults如图[1](#_bookmark5)所示。我们发现，所有的攻击成功率几乎是线性增长的，因为我们增加了modiﬁcations。ELMo和BERT的表现几乎相同，而InferSent在开始- ning时增长较快，RoBERTa在到达终点时增长较慢。当错误率在20%左右时，平均攻击成功率达到70%。

# 模型能在多大程度上识别语法错误？

我们本节的目标是评估预训练的编码器识别语法错误的能力。我们使用二进制语言接受性任务来测试模型在判断句子的语法正确性方面的能力。我们进一步研究了模型是否能够精确定位错误位置，这重新体现了token层面的能力。

**数据** 我们为每一个spe- ciﬁc类型的语法错误构建了单独的数据集。对于每个数据集，我们从1B Word Benchmark（[Chelbaetal.](#_bookmark18)，[2014](#_bookmark18)）中提取10,000个句子，其长度落在10到60个tokens之内。然后，我们将目标错误类型引入到这些句子的一半，使用*proba- bilistic转换*，并将每个数据集的错误率保持在3%左右（导致一到两个



50 60 70 80 90 20 30 40 50 60 70 80 90

图2：在四种错误类型上探究四层BERT。左侧显示了二进制语言接受性任务的准确性。右侧显示了定位错误位置的准确性。每一行repre- sents一个speciﬁc层，每一列代表一种错误类型，从左到右分别是ArtOrDet、Nn、SVA、Worder。完整的结果在附录[D](#_bookmark60)中给出

每个句子中的错误）。)句子分为训练（80%）、开发（10%）和测试（10%）。

**模型** 我们研究ELMo（2层）、BERT-base-uncased（12层）和RoBERTa-base（12层）的各个层。特别是，我们对每一层进行量化，并在其上附加一个可训练的自注意力层，以获得一个句子表示。句子表示被输入到线性分类器中，以输出该句子在语言上是否可接受的概率。关于自注意力层和线性分类器的细节请参见附录。

[B.3](#_bookmark56).接下来我们从训练的自注意力层中提取权重最大的前两个位置。如果包含有错误token的位置，我们认为错误是由模型在token层面正确定位的。这表明上下文编码器是否为分类器提供了足够的信息，以识别错误位置。为了进行比较，我们还评估了每个模型的输入 em- bedding 层（非语境化，0 层）作为基线。我们计算了句子级和token级评估的准确性。

**结果和讨论** 我们在图[2](#_bookmark6)中直观地显示了四层BERT对四种误差类型，ArtOrDet，Nn，SVA和Worder的重新结果。所有层和其他错误类型的完整结果见附录[D](#_bookmark60)。我们发现，BERT、ELMo和RoBERTa的最佳上下文层在不同错误类型上的平均句子级准确率分别为87.8%、84.3%和90.4%，而输入嵌入层则达到64.7%、65.8%和66.0%。在token-level中，尽管只训练了?

0.5

0.4

0.3

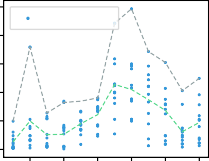
Acc.

0.2

0.1

0.0

**预备**



10

Attn Head10

10

7

4

6

6

6 12 4

2

3

2 4 6 8 10 12

Acc.

0.6

0.4

0.2

0.0

**sva**



2 6

5

12 12

4

9

12

1

5

4 5

收件人：负责人

2 4 6 8 10 12

Prep Art Wci Tras Nn SVA.

Vform

-6 -5 -4 -3 -2 -1 1 2 3 4 5 6

图3：BERT的每个注意力头在token级评价上的准确性。灰色线代表表现最好的头。绿线代表一层中各头的平均表现。

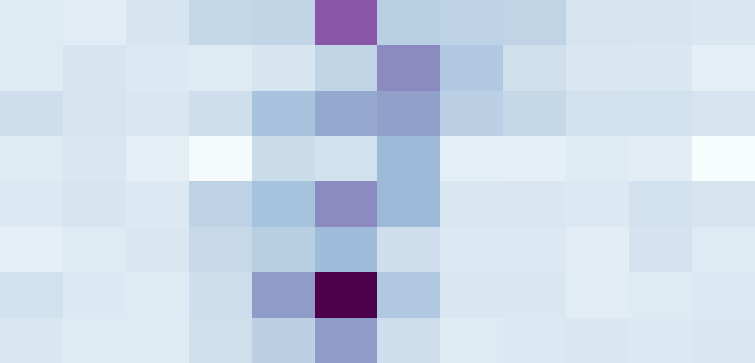
句子是否可以接受的预测，BERT、ELMo和RoBERTa的最佳层上分类器的平均准确率分别为79.3%、63.3%。

和80.3%，而输入嵌入层分别为48.6%、18.7%和53.4%。这两个事实说明，与输入嵌入层相比，这些预训练的编码器层具有更强的语法错误检测和定位能力。

我们还观察到了与speciﬁc模型相关的模式。具体来说，BERT的中间层（第7-9层）比低层或高层更好地识别错误，如图[2](#_bookmark6)所示。但更高层的BERT对与长程依赖和动词（如SVA和Vform）相关的错误定位更准确。为了进一步研究BERT对错误位置的认知。我们对BERT中的144个注意头进行同样的token级评估。图[3](#_bookmark7)中直观地显示了Prep和SVA的结果。我们发现，即使是在完全无监督的情况下，一些注意力头在定位错误方面的准确率也达到了50%-60%。与自注意层一致，中层的注意头表现最好。所有错误类型见附录[F](#_bookmark62)。

由于篇幅有限，我们在附录[D](#_bookmark60)中介绍了RoBERTa和ELMo的结果，并在下文中总结了观察结果。与 BERT 相比，RoBERTa 在低层的错误检测和定位能力更强，在顶层（第 10-11 层）的表现最好。它也擅长捕捉动词和后置性错误。另一方面，ELMo的第5层始终给出最高的句子级分类精度。但其定位错误的最佳表现层取决于错误类型，并在第五层和第二层之间变化。特别是，ELMo的第二层在定位Nn方面具有很强的能力，在准确率上优于BERT。考虑到Nn在字符嵌入中并不明显，这一点令人惊讶。

Vt

图4：探测作为MLM的BERT。每一行代表一个目标错误类型。每一列代表与错误位置的距离。每个数字代表所有对的平均似然降。我们发现，speciﬁc tokens受错误tokens的影响更大。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0.00 | -0.00 | 0.01 | 0.02 | 0.02 | 0.09 | 0.02 | 0.02 | 0.02 | 0.01 | 0.01 | 0.00 |
| 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.02 | 0.06 | 0.03 | 0.01 | 0.00 | 0.00 | -0.00 |
| 0.01 | 0.01 | 0.00 | 0.01 | 0.03 | 0.05 | 0.05 | 0.02 | 0.02 | 0.01 | 0.01 | 0.01 |
| 0.00 | 0.00 | -0.00 | -0.02 | 0.01 | 0.01 | 0.04 | -0.00 | -0.01 | 0.00 | -0.00 | -0.02 |
| 0.00 | 0.01 | 0.00 | 0.02 | 0.03 | 0.06 | 0.04 | 0.00 | 0.00 | 0.00 | 0.01 | 0.01 |
| -0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.04 | 0.01 | 0.00 | 0.00 | -0.00 | 0.01 | 0.00 |
| 0.01 | 0.00 | 0.00 | 0.01 | 0.06 | 0.14 | 0.03 | 0.00 | 0.00 | -0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.01 | 0.02 | 0.06 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |

从ELMo的0层。我们进一步注意到，对于所有模型来说，Worder是句子层面最难检测的类型，而ArtOrDet和Worder是令牌层面最难定位的类型。我们推测这与这些错误的位置性有关，它为模型识别这些错误诱导了一个弱信号。附录 [E](#_bookmark61) 展示了 BERT 令牌级评估的一些示例。

# 当错误出现时，BERT6 如何捕捉代币之间的交互作用。

我们旨在通过研究BERT的屏蔽语言模型(MLM)组件来揭示语法错误与其附近标记之间的相互作用。我们研究BERT，因为它是一个典型的基于变压器的编码器。我们的分析可以扩展到其他模型。

**实验设置** 我们对 NUCLE 的最小编辑对进行实验。我们对带有错误标签ArtOrDet, Prep, Vt, Vform, SVA, Nn, Wchoice, Trans的对子进行分类，并保留那些只有一个标记被改变的对子。这样我们就得到了8个最小编辑对的集合，大小分别为586、1525、1817、943、2513、1359、3340。

和452。

给定一个最小的编辑对，我们考虑距离错误token六token以内的token。我们每次用[MASK]替换语法句和非语法句中的同一个token，并使用BERT作为MLM来预测其似然。然后我们计算非语法句中的似然下降，并获得所有最小编辑对的平均下降。

**结果和讨论**结果可视化在图[4](#_bookmark8)。在一般情况下，我们发现，减少对speciﬁc位置的相似性比别人大

这样一来，根据收入上限，就可以减轻**这部分**人的经济负担。这样一来，就可以减轻这部分人的经济负担。

√

0.9

腐败

l

Ori gi na

**这些**人根据他们的收入上限。

*x*

本（这些）组of 0.01 0.09 - 0.410.02。

廉价的燃料成本和庞大的数量。

√**核反应堆**产生的能量远远超过研发成本。

廉价的燃料成本和庞大的数量。

0.8

0.7

指数

0.6

0.5

0.0 0.2 0.4 0.6 0.8 1.0

比例

核反应堆weighs the cost of research and development . produced 所产生的能量远远超过核反应堆。

*x*

0.05 -0.02 - 0.31 0.42

表5：ArtOrDet的例子。 我们显示了最小的imal编辑对和每个tokens距离错误的两个tokens内的似然减少。错误的去术语和它们的修正被标记为红色。定语-名词依赖关系中的词头用蓝色标记。如表所示，定语-名词依赖关系中的词头得到最大的似然下降。

在存在错误的情况下。鉴于事实上，如主语-动词和定语-名词依赖性等标记之间的某些依赖性由BERT准确建模，正如先前的工作（[Jawahar等人](#_bookmark27)，[2019](#_bookmark27)）所证明的那样，我们怀疑错误标记的存在将主要影响其邻近的- ing标记（在句法和物理邻居方面）。这与我们在图[4](#_bookmark8)中的观察一致，即在SVA的情况下，主语大多是动词的前一个令牌（尽管主语和动词之间可以存在协议吸引物），错误位置的进行令牌总体上得到最大的可能性下降。在ArtOrDet的情况下，文章或确定词可以是后续名词的指示器和从属词，预测错误位置的下一个标记变得更加困难。我们在表[5](#_bookmark9)中提供了两个用ArtOrDet运行的例子来进一步说明这一点。

# 对抗性培训

最后，我们探索了一种基于提出的语法错误模拟的数据增强方法。我们应用贪婪搜索算法将语法错误引入到目标任务的训练考试中，并在原始例子和通用例子的组合上重新训练模型。我们以MRPC([Dolan and Brockett](#_bookmark25), [2005](#_bookmark25))数据集为例来演示结果。我们在训练集上增加了

图5：数据增强防御的结果。比例表示与原始量相比，对训练集的对抗性考试- ples增强的数量。虚线和实线分别显示了不同比例下对损坏和原始开发集的准确率。

MRPC与不同比例的对抗性ex-amples，在增强的训练集上重新调整BERT，然后在原始开发集和损坏的开发集上进行评估。

结果如图[5](#_bookmark10)所示，我们发现通过增加少量的对抗性例子，准确率从46%恢复到82%。随着增强的对抗性例子比例的增加，在被破坏的集合上的准确率继续增加，而对origi- nal验证准确率的变化可以忽略不计。这一事实也证明了我们的模拟例子对于降低语法错误的影响是有潜在帮助的。

# 结论

在本文中，我们对语言编码器对语法错误的鲁棒性进行了彻底的研究评估。我们提出了一种新的方法来模拟语法错误和facili- tating我们的评价。我们研究了三种预先训练好的语言编码器，ELMo、BERT和RoBERTa，并集中研究了它们对语法错误的适应性的三个方面：面对无声文本时在下游任务中的表现，识别错误的能力，以及在存在错误的情况下捕捉标记之间的相互作用。本研究揭示了对语言编码器对抗语法错误的行为的理解，并鼓励未来的工作以提高这些模型的鲁棒性。

**Acknowledgements** 我们要感谢匿名审稿人的反馈。这项工作是由美国国家科学基金会赠款＃IIS-1927554支持。

# 参考文献

Yossi Adi、Einat Kermany、Yonatan Belinkov、Ofer Lavi和Yoav Goldberg。2017.[Fin-grained anal- ysis of sentence embeddings using auxiliary pre- diction tasks](https://openreview.net/forum?id=BJh6Ztuxl). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Pro- ceedings*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang.2018.[Generating natural language adver- sarial examples](https://www.aclweb.org/anthology/D18-1316/).In *Proceedings of the 2018 Con- ference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - Novem- ber 4, 2018*, pages 2890-2896.

Antonios Anastasopoulos.2019.An analysis of source- side grammatical errors in nmt.In *Proc.Black- boxNLP*.

Timothy Baldwin, Trevor Cohn, and Yitong Li.2017.[Robust training under linguistic adversity](https://www.aclweb.org/anthology/E17-2004/).In *Pro- ceedings of the 15th Conference of the European Chapter of the Association for Computational Lin- guistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 21-27.

Yonatan Belinkov和Yonatan Bisk。2018.[Synthetic and natural noise both break neural machine trans- lation](https://openreview.net/forum?id=BJ8vJebC-).In *6th International Conference on Learn- ing Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Steven Bird和Edward Loper。2004.[NLTK:natu- ral language toolkit](https://www.aclweb.org/anthology/P04-3031/).In *Proceedings of the 42nd An- nual Meeting of the Association for Computational Linguistics, Barcelona, Spain, July 21-26, 2004 - Poster and Demonstration*.

Jill Burstein, Christy Doran, and Thamar Solorio, ed- itors.2019.[*Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:Human Language Technologies, NAACL-HLT 2019, Minneapolis,MN, USA, June 2-7, 2019,Volume1 (Long and ShortPa- pers)*](https://www.aclweb.org/anthology/volumes/N19-1/).Association for Computational Linguistics.

Ciprian Chelba、Tomas Mikolov、Mike Schuster、Qi Ge、Thorsten Brants、Phillipp Koehn和Tony Robin- son。2014.[One billion word benchmark for measur- ing progress in statistical language modeling](http://www.isca-speech.org/archive/interspeech_2014/i14_2635.html).In *IN- TERSPEECH 2014，15th Annual Conference of the International Speech Communication Association，Singapore，September 14-18，2014*，pages 2635-。

2639.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning.2019.What does bert look at? An analysis of bert's attention.In *Black- BoxNLP@ACL*.

Alexis Conneau、Douwe Kiela、Holger Schwenk、Lo¨éc Barrault和Antoine Bordes。2017.[监督的](https://www.aclweb.org/anthology/D17-1070/)

[learning of universal sentence representations from natural language inference data](https://www.aclweb.org/anthology/D17-1070/).In *Proceedings of the 2017 Conference on Empirical Methods in Nat- ural Language Processing，EMNLP 2017，Copen- hagen，Denmark，September 9-11，2017*，pages 670-。

680.

Alexis Conneau、Germa´n Kruszewski、Guillaume Lam- ple、Lo¨c Barrault和Marco Baroni。2018.[What you can cram into a single $&！#\* vector:Probing sentence embeddings for linguistic properties](https://doi.org/10.18653/v1/P18-1198).In *Proceedings of the 56th Annual Meeting of the As- sociation for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1:*

[*\*](https://doi.org/10.18653/v1/P18-1198)

*长篇论文*，第2126-2136页。

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu.2013.[Building a large annotated corpus of learner english:The NUS corpus of learner english](https://www.aclweb.org/anthology/W13-1703/).In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL-HLT 2013, June 13, 2013, Atlanta,*

*美国乔治亚州*，第22-31页。

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Car- bonell, Quoc Le, and Ruslan Salakhutdinov.2019.[Transformer-XL。Attentivelanguagemodelsbeyond a ﬁxed-length context](https://doi.org/10.18653/v1/P19-1285).In *Proceedings of the 57th Annual Meeting of the Association for Computa- tional Linguistics*, pages 2978-2988, Florence, Italy.Association for Computational Linguistics.

Jacob Devlin，Ming-Wei Chang，Kenton Lee，and Kristina Toutanova.2019.[BERT: pre-training of deep bidirectional transformers for language under- standing](https://www.aclweb.org/anthology/N19-1423/).In ([Burstein et al.](#_bookmark17) , [2019](#_bookmark17)), pages 4171- 4186.

William B. Dolan和Chris Brockett。2005.[Automati- callyconstructingacorpusofsententialparaphrases](https://www.aclweb.org/anthology/I05-5002/).In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*.

Yoav Goldberg.2019.[Assessing bert's syntactic abili- ties](http://arxiv.org/abs/1901.05287).*CoRR*，abs/1901.05287.

Ganesh Jawahar, Benoˆıt Sagot, and Djame'Seddah.2019.[What does BERT learn about the structure of language?](https://www.aclweb.org/anthology/P19-1356/)In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Vol- ume 1: Long Papers*, pages 3651-3657.

金迪、金志敬、周天一和Peter Szolovits.2019.[IsBERTreally robust? natural languageattackontextclassiﬁcationandentailment](http://arxiv.org/abs/1907.11932).*CoRR*，abs/1907.11932.

Najoung Kim、Roma Patel、Adam Poliak、Patrick Xia、Alex Wang、Tom McCoy、Ian Tenney、Alexis Ross、Tal Linzen、Benjamin Van Durme、Samuel R. Bow-man和Ellie Pavlick。2019.[Probing what differ- ent NLP tasks teach machines about function word comprehension](https://www.aclweb.org/anthology/S19-1026/).In *Proceedings of the Eighth Joint*

*Conference on Lexical and Computational Seman- tics, \*SEM@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 235-249.

Diederik P. Kingma 和 Jimmy Ba.2015.[Adam: A method for stochastic optimization](http://arxiv.org/abs/1412.6980).In *3rd Inter- national Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015,*

*会议记录*：

Tal Linzen、Emmanuel Dupoux和Yoav Goldberg。2016.[Assessing the ability of lstms to learn syntax- sensitive dependencies](https://transacl.org/ojs/index.php/tacl/article/view/972).*TACL*，4：521-535。

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith.2019a.Lin- guistic knowledge and transferability of contextual representations.In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics:Human Language Tech- nologies*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Man- dar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.2019b.[Roberta。ArobustlyoptimizedBERTpretrainingap- proach](http://arxiv.org/abs/1907.11692).*CoRR*，abs/1907.11692.

Alison Lui, Antonios Anastasopoulos, and David Chi- ang.2018.[Neural machine translation of text from non-native speakers](http://arxiv.org/abs/1808.06267).*CoRR*，abs/1808.06267。

丽贝卡-马文和塔尔-林岑。2018.[Targetedsyn- tactic evaluation of language models](https://www.aclweb.org/anthology/D18-1151/).In *Proceed- ings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1192-1202.

Matthew E. Peters、Mark Neumann、Mohit Iyyer、Matt Gardner、Christopher Clark、Kenton Lee和Luke Zettlemoyer。2018.[Deep contextualized word rep- resentations](https://www.aclweb.org/anthology/N18-1202/).In *Proceedings of the 2018 Confer- ence of the North American Chapter of the Associ- ation for Computational Linguistics:Human Lan- guage Technologies，NAACL-HLT 2018，New Or- leans，Louisiana，USA，June 1-6，2018，Volume 1。*

*长篇论文）*，第2227-2237页。

马修-E-彼得斯、塞巴斯蒂安-鲁德和诺亚-A-史密斯。2019.[Totune or not to tune? adapting pre-trained representations to diverse tasks](https://www.aclweb.org/anthology/W19-4302/).In *Proceed- ings of the 4th Workshop on Representation Learn- ing for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7-14.

Slav Petrov、Leon Barrett、Romain Thibaux和Dan Klein。2006.[学习准确的、紧凑的、和可互读的树注释](https://www.aclweb.org/anthology/P06-1055/)。In *ACL 2006, 21st Interna- tional Conference on Computational Linguistics and 44th Annual Meeting of the Association for Compu- tational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang.2016.[Squad:100，000+问题的机器理解文本](https://www.aclweb.org/anthology/D16-1264/)。In *Proceedings of*

*2016年11月1-4日，美国德克萨斯州奥斯汀，2016年11月1-4日召开的2016年自然语言处理经验方法会议，EMNLP 2016，*第2383-2392页。

Marco Ribeiro, Sameer Singh, and Carlos Guestrin.2018.[Semantically equivalent adversarial rules for debugging nlp models](https://doi.org/10.18653/v1/P18-1079).Page 856-865.

Erik F. Tjong Kim Sang和Fien De Meulder。2003.[介绍conll-2003共享任务。Language-independent named entity recognition](http://arxiv.org/abs/cs.CL/0306050).*CoRR*，cs.CL/0306050。

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts.2013.[Recursive deep mod- els for semantic compositionality over a sentiment treebank](https://www.aclweb.org/anthology/D13-1170/).In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Process- ing, EMNLP 2013, 18-21 October 2013, Grand Hy- att Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631-1642.

Matthias Sperber, Jan Niehues, and Alex Waibel.2017.Toward robust neural machine translation for noisy input sequences.

Ian Tenney, Dipanjan Das, and Ellie Pavlick.2019a.[BERTrediscovers the classical NLP pipeline](https://www.aclweb.org/anthology/P19-1452/).In *Proceedings of the 57th Conference of the Associ- ation for Computational Linguistics, ACL 2019, Flo- rence, Italy, July 28- August 2, 2019, Volume 1:*

*长篇论文*，第4593-4601页。

Ian Tenney、Patrick Xia、Berlin Chen、Alex Wang、Adam Poliak、R Thomas McCoy、Najoung Kim、Benjamin Van Durme、Sam Bowman、Dipanjan Das和Ellie Pavlick。2019b.[What do you learn from context? probing for sentence structure in contextu- alized word representations](https://openreview.net/forum?id=SJzSgnRcKX).In *International Con- ference on Learning Representations*.

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Lukasz Kaiser和Illia Polosukhin。2017.[注意力是你所需要的](http://papers.nips.cc/paper/7181-attention-is-all-you-need)。In *Advances in Neural Information Pro- cessing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 Decem-。*

*ber 2017, Long Beach, CA, USA*, pages 5998-6008.

Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy和Samuel R. Bowman.2019a.[GLUE：A multi-task benchmark and analysis plat- form for natural language understanding](https://openreview.net/forum?id=rJ4km2R5t7).In *7th International Conference on Learning Representa- tions, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Alex Wang，Ian F. Tenney，Yada Pruksachatkun，Katherin Yu，Jan Hula，Patrick Xia，Raghu Pappa- gari，Shuing Jin，R.Thomas McCoy，Roma Pa- tel，Yingghui Huang，Jason Phang，Edouard Grave，Haokun Liu，Najoung Kim，Phu Mon Htut，Thibault F'evry，Berlin Chen，Nikita Nangia，Anhad Mo- hananey，Katharina Kann，Shikha Bordia，Nicolas

Patry, David Benton, Ellie Pavlick, and Samuel R. Bowman.2019b. jiant 1.2: A software toolkit for research on general-purpose text understanding models. http://jiant.info/.

Alex Warstadt和Samuel R. Bowman.2019.[Grammatical analysis of pretrained sentence en- coders with acceptability judgments](http://arxiv.org/abs/1901.03438).*CoRR*，abs/1901.03438.

Alex Warstadt、Yu Cao、Ioana Grosu、Wei Peng、Ha- gen Blix、Yining Nie、Anna Alsop、Shikha Bordia、Haokun Liu、Alicia Parrish、Sheng-Fu Wang、Jason Phang、Anhad Mohananey、Phu Mon Htut、Paloma Jeretic和Samuel R. Bowman。2019a.[Investi- gating bert's knowledge of language:Five analysis methods with npis](http://arxiv.org/abs/1909.02597).*CoRR*，abs/1909.02597.

Alex Warstadt，Alicia Parrish，Haokun Liu，Anhad Mo- hananey，Wei Peng，Sheng-Fu Wang，and Samuel R Bowman.2019b.Blimp。A benchmark of lin- guistic minimal pairs for english.*arXiv preprint arXiv:1912.00582*.

Adina Williams, Nikita Nangia, and Samuel R. Bow-man.2018.[A broad-coverage challenge corpus for sentence understanding through inference](https://www.aclweb.org/anthology/N18-1101/).In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computa- tional Linguistics:Human Language Technologies，NAACL-HLT 2018，New Orleans，Louisiana，USA，June 1-6，2018，Volume 1（Long Papers）*，pages

1112–1122.

托马斯-沃尔夫、莱桑德-德布特、维克多-桑赫、朱利安-肖蒙、克莱门特-德朗格、安东尼-莫伊、皮埃尔-里克-西斯塔克、蒂姆-罗尔特、R'emi-卢夫、摩根-芬托-伊兹和杰米-布鲁。2019.Huggingface 的 trans- formers。最先进的自然语言处理- ing.*ArXiv*, abs/1910.03771.

# 下游任务详情

我们在四个语言理解和一个Se-quence标签数据集上进行测试。这些数据集的统计数字列在表[6](#_bookmark57)中。

**MRPC** Microsoft Research Paraphrase Cor- pus (MRPC) ([DolanandBrockett](#_bookmark25), [2005](#_bookmark25))是一个para-短语检测任务，其目的是预测两个句子在语义上是否等同的二元标签。

**MNLI** 多类型自然语言推理语料库（Multi-genre Natural Language In- ference Corpus，MNLI）（[Williams 等，2018](#_bookmark52)）是一个广域自然语言推理任务，用于预测前提和假设之间的关系（包含、矛盾、中性）。MNLI包含匹配（域内）和错误匹配（跨域）两个部分。

**QNLI** 问题回答NLI任务（QNLI）是从斯坦福问题回答数据集（[Rajpurkar et al.](#_bookmark39) , [2016](#_bookmark39)）重铸而来，其目的是确定一个上下文句子是否包含问题的答案（entailment, not entailment）。

**SST-2** Stanford Sentiment Treebank 双向类拆分（SST-2；（[Socher et al.](#_bookmark42)，[2013](#_bookmark42)））是 句子分配积极或消极的标签。

CoNLL-2003 **- NER** CoNLL-2003命名实体识别(NER)([Sang和Meulder](#_bookmark41), [2003](#_bookmark41))的共同任务是一个令牌级序列标签任务，以识别四种类型的命名实体：每子、位置、组织和不属于前三组的混淆实体的名称。

# 型号详情

## 预先培训的编码器详细信息

我们研究了BERT(base，uncased)、BERT(base，cased)(仅适用于NER)、RoBERTa(base)和ELMo。BERT（base）和RoBERTa（base）具有相同的架构。它们都是深度跨前模型，有12层和12个注意头，每层有768个隐藏大小。它们包含一个可学习的输出层，用于[CLS]或<s>上的ﬁne-tuning。我们使用来自[Wolf等人](#_bookmark53)（[2019](#_bookmark53)）的BERT和RoBERTa的PyTorch实现，并在下游任务上对它们进行ﬁne-tune。对于ELMo，我们将ELMo表示作为上下文嵌入的标记，并将它们馈送到一个两层，1500D BiL- STM，具有跨句子注意机制，如在*jiant*中实现。([Wanget al.](#_bookmark48) , [2019b](#_bookmark48))。

## 培训和微调细节

对于BERT和RoBERTa，我们将所有任务的最大输入长度设置为128，最大epochs数为3，dropout为0.1。我们使用Adam（[Kingma和Ba](#_bookmark30)，[2015](#_bookmark30)），初始学习率为2e-5，批次大小为16，训练时没有预热步骤。对于ELMo，我们使用Adam（[Kingma和Ba](#_bookmark30)，[2015](#_bookmark30)）训练BiLSTM，初始学习率为1e-4，批次大小为32。我们将 dropout 设置为 0.2，最大 epochs 数为 10，当性能在 2 个 epochs 内没有改善时，将学习率除以 5。

## 探测模型 详情

我们在第5节中使用一个自注意力层和一个线性分类器来组成探测组件。自注意层将编码器的*第i*层的隐藏表示作为输入，表示为*h* = *{hi ，hi ，...，hi }*，并输出一个sen-。

1

2

*n*

tence代表*si*。

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集 | 火车 | 开发 | 平均列数 |
| MRPC | 3.7k | 409 | 22.4 |
| MNLI | 393k | 19k | 10.1 |
| QNLI | 105k | 5.5k | 27.6 |
| SST-2 | 67k | 873 | 19.5 |
| CoNLL2003 | 15k | 3k | 14.8 |

*si* = Σ*n*

*j*=1

*αjhi*

*j*

(1)

*αj* = *softmax*(*vT tanh*(*Wahi* ))(2)

*b j*

表6.数据集统计表6：MRPC、MNLI、QNLI、SST-2和CoNLL2003的数据集统计。Train和Dev代表训练集和开发集的句子数量。Avg Len代表被攻击的目标句子的平均句子长度（以- ken为单位）。

其中*Wa*是权重矩阵，*vb*是参数向量 *si*被反馈给分类器，输出句子在语言上被接受的概率。自注意层的隐藏dim为100，dropout为0.1。分类器有1层和0.1个dropout。用Adam（[Kingma和Ba](#_bookmark30)，[2015](#_bookmark30)）训练探测模型，学习率为0.001，批次大小为8，L2权重衰减为

0.001为10个纪元，早停耐心为2。

# 攻击算法

我们基于NUCLE上的真实错误（[Dahlmeier et al.](#_bookmark22) ，[2013](#_bookmark22)），进行了三种搜索算法，*贪婪搜索*、*波束搜索*、*遗传算法*在Adver- sarial攻击中的应用。对于*波束搜索*，我们设置波束大小为5，对于*遗传算法*，我们设置每一代的人口为60，设置最大代数为对应句子长度的23%。例如，如果一个句子有100个tokens，遗传算法最多迭代23次。算法[1](#_bookmark59)、[2](#_bookmark63)、[3](#_bookmark64)分别是贪婪攻击、波束搜索攻击和遗传算法攻击的详细描述。

**算法1** 贪婪攻击 10cm

*{}*

**输入。**原句 *Xori* = w1*,* w2*, ..., wn* , 地面。

真值预测*Yori*，目标模型*F* ，所有混淆集*P* ，预算*b*。

**输出。***Xadv*. 1：*Xadv Xori*

*←*

2：**对于***Xori*中的每一个*wi，***做**

3: 删除*wi*，并计算*Yori*上的似然下降，得到*wi*的重要性得分，表示为*Sw』* 。

4: 将*P*的所有替换应用于*wi*。获得

*wi*的操作池，表示为*Wsub*。

*i*

5：**结束**

6:

7：按照令牌重要性的递减顺序得到*Xori*的索引列表：*argsortw』。XóT*(*Sw"* )

8:*←e*

**对于**每一个*i*在*I***做**

9:*pori ← F* (*Xadv* )*|Y* =*YóT』。*

10: **对于***Wsub*中的每个*w′***，做**

*i*

11: 在*Xadv*中用*w′*代替*wi*(或交换它们的位置)

# 探究模型在识别错误方面的能力。

12:

13:

职位）。)

*Yadv argmaxF* (*Xadv*), *padvF* (*Xadv* ) *Y* =*YóT』。*

*←|*

*←*

**如果**不是*Yori* = *Yadv***，则返回***Xadv。*

## 句子级二进制分类任务

表[7](#_bookmark65)显示了在句子级二进制分类ﬁcation任务中探究ELMo、BERT和RoBERTa的individ- ual层跨越8种错误类型的完整结果。我们对预先训练好的编码器的参数进行量化，并为每一层训练一个自注意力分类器来判断句子的二元语言接受能力。We ﬁnd that layer 1 of ELMo, middle layers of BERT, and top layers of RoBERTa perform the best in this evaluation.

## 令牌级错误定位任务

表[8](#_bookmark66)显示了在token-level的8种错误类型中探究individ- ual层的ELMo、BERT和RoBERTa的完整结果。我们首先对预先训练好的编码器的参数进行量化，并为每个层训练一个自注意力分类器，以判断句子的二元语言可接受性。然后，我们提取自注意力层的两个具有最高 atten- tion 权重的位置，看是否包含错误标记。

# 定位错误位置的案例分析

我们在第5节展示了一些令牌级评估的例子。我们为每一种错误类型随机选择一个考题，并将BERT不同层的自注意力层的注意力权重可视化。每个token下的紫色越深，意味着自注意力层在这个token上投入了更多的注意力。

14:**其他**

15:**if** *padv < pori* **then**

16: *wselect w′*, *pori 垫子*

*←←*

17:**结束如果**

18:**结束如果**

19:**结束**

20:**如果**迭代次数超过*b***，则返回***Xori*。

21:**结束如果**

在*Xadv*中用*wselect*22: 代替*wi*，23：**结束。**

24：**归还***Xori*

# BERT的注意力头的代币级评估

如第5节所述。我们也对BERT的144个注意头进行了同样的token级别的探测。在这个实验中，BERT中的参数是完全冻结的。我们观察到，即使在这种无监督的方式下，一些注意力头仍然能够精确地定位错误位置。BERT的中层表现最好。我们进一步观察到，一些注意力头可能与特定类型的错误有关。例如，第9层的2号头和第10层的6号头擅长捕捉SVA和Vform。这两个错误都与动词有关。

**算法2** 波束搜索攻击

**输入。**原句*Xori* = w1*,* w2*, ..., wn* , 地面真相预测*Yori*, 目标模型*F*, 所有混淆集*P*, 预算*b*, 束大小*bm*.

*{}*

**输出。**对抗性例子*Xadv*.

1：初始化：*bestBeam*复制*Xori*为*bm*次。2: **对于***Xori*中的每一个*wi，***执行**

*←*

3：删除*wi*，并计算*Yori*上的似然下降，得到*wi*的重要性得分，表示为*Sw』* 。

4: 将*P*的所有替换应用于*wi*。获得

*wi*的操作池，表示为*Wsub*。

*i*

5：**结束**

6:

7：根据令牌重要性的递减顺序，得到*Xori*的索引列表。*I ← argsortw』eXóT』。*(*Sw"* )

8: **对于***Wsub*中的每个*w′，***做**

*I*[0]

9: 将*Xori*中的*wi*改为*w′*(或将其换掉)

**算法3** 遗传攻击

**输入。**原句*Xori* = w1*,* w2*, ..., wn* , 地面真相预测*Yori*, 目标模型*F*, 所有混淆集*P*, 预算*b*, 人口规模*ps*, 代数*G*.

*{}*

**输出。**对抗性例子*Xadv*.

1：用空集初始化第五代。2：**对于***Xori*中的每一个*wi，*都**要这样做**

*← 份*

3: 将*P*的所有替换应用于*wi*。得到*wi*的操作池，表示为*Wsub*。

*i*

4：**结束**

5：**对于***i*=1*，*2*，*3*，...，ps***做**

6: 随机选择一个位置*j*和一个操作，从

*Wsub*来修改*Xori*。然后加到*P* 0。

*j*

7：**结束**

8:

9：**对于***g*=1*，*2*，*3*，...，*G1**做**

*-*

10:**对于***i* = 1*，*2*，*3*，...，ps***做**

11：*Yadv ← argmaxF*（*P*）。

*g\_*1

职务)

10:*Yadv ← argmaxF*（*Xori*）。

11:*←|*

*i*

*padvF* (*Pg\_*1) *Y* =*Y。*

*←|*

*ióT*

12:**if** not *Yadv* = *Yori* **then return** *Pg\_*1

*padvF*(*Xori*)*Y* =*YóT」。*

**如果**不是*Yori* = *Yadv***，则返回***Xori。*

12:**其他**

13:**其他**

14:*Xelite ← argmin*(*p)*

*i*

*预言* )

15:*Pg ← {Xelite}。*

13:topBeamRecord*顶层建筑*的例子，*padv*最低。

*←*

14:**结束如果**

16:

1

*prob ←* 将样本概率标准化，用

*F* (*Pg\_*1)

*i*

15：**结束**

16:

17:**对于***i* = 2*，*3*，...，ps***做**

18: 从*Pg\_*118: 中取样parent1，有问题。

17: *bestBeam TopBeam*

*←*

18: **对于***I/I*[0]中的每个*i，***做**

19: *pori ← F* (*Xadv*)*|Y* =*YóT*

19:

20:

*试探*

从*P概率*中抽取父母2样本

*←*

*←*

*j*

*g\_*1

有问题

20:*plist*

*← {}*

21:**对于***bestBeam*中的每个*Xbeam***，做**

22: **对于***Wsub*中的每个*w′，***做**

*i*

21:

childCrossover(parent1, parent2) *childmut*从*Wsub*中随机选择一个位置和一个操作来修改。

23: 在*Xbeam*位置用*w′*代替*wi*)

24:*Yadv ← argmaxF*（*Xbeam*）。

*padv ← F* (*Xbeam*)*|Y* =*Y。*

掉换

*孩子*

22:*Pgchildmut*

*i*

*←*

23:**结束**

24:**结束如果**

25:**如果**不是*Y*

26:**其他**

*鸟类*

*óT*

= *Yadv* **然后返回**

*Xbeam*

25:**结束**

26：**结束**

27: 增加*任择议定书* (*w′ , padv, Xbeam*) to *oplist* 28:**end if**

*←*

29:**结束**

30：**结束**

31:**如果**迭代次数超过*b***，则返回***Xori。*

32:**结束如果**

33: 选择*op.padv*最低的*op.bm*操作。

每一次*op.Xbeam*更新*bestBeam*。

34：**结束**

35：**返回***Xori*

27：**返回***Xori*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **准备工作** | **Artordet** | **Vform** | **Nn** | **Wchoice** | **Trans** | **SVA** | **Worder** |
| ELMo，0层 | 62.6 | 65.0 | 69.6 | 67.7 | 74.5 | 67.5 | 72.1 | 47.6 |
| ELMo，第1层 | **90.6** | **84.7** | **87.2** | **82.9** | **83.9** | **80.6** | **93.1** | **71.2** |
| ELMo，第2层 | 84.7 | 77.0 | 79.4 | 79.7 | 82.6 | 74.4 | 89.9 | 68.5 |
| BERT，0层 | 62.5 | 60.8 | 67.4 | 64.6 | 73.9 | 69.5 | 70.3 | 48.2 |
| BERT，第1层 | 68.0 | 63.4 | 69.3 | 70.3 | 75.0 | 71.5 | 78.4 | 52.2 |
| BERT，第2层 | 74.4 | 67.0 | 75.3 | 74.8 | 76.7 | 73.1 | 84.4 | 62.0 |
| BERT，第3层 | 80.5 | 75.0 | 83.4 | 73.7 | 78.5 | 76.3 | 89.2 | 69.8 |
| BERT，第4层 | 82.7 | 80.7 | 83.6 | 77.7 | 82.6 | 79.6 | 90.6 | 72.4 |
| BERT，第5层 | 85.2 | 83.8 | 85.4 | 84.3 | 84.5 | 81.8 | 91.7 | 71.9 |
| BERT，第6层 | 88.2 | 86.6 | 85.8 | 86.7 | 84.5 | 82.6 | 90.9 | 73.4 |
| BERT，第7层 | 91.3 | 88.1 | 90.2 | 86.5 | **86.9** | 83.9 | **95.3** | 73.4 |
| BERT，第8层 | **92.5** | **88.3** | **91.4** | **88.4** | 86.3 | **85.5** | 94.5 | **73.8** |
| BERT，第9层 | 91.4 | 86.3 | 89.9 | 87.4 | 85.6 | 84.9 | 94.4 | 72.4 |
| BERT，第10层 | 90.8 | 87.4 | 88.2 | 87.0 | 86.1 | 84.8 | 94.9 | 71.8 |
| BERT，第11层 | 90.0 | 84.9 | 88.1 | 86.6 | 85.6 | 84.3 | 94.2 | 69.5 |
| BERT，第12层 | 88.4 | 85.6 | 88.1 | 84.3 | 84.0 | 82.6 | 93.3 | 68.1 |
| RoBERTa，0层 | 61.9 | 65.9 | 69.7 | 67.1 | 75.1 | 69.1 | 68.3 | 50.9 |
| RoBERTa，第1层 | 78.3 | 74.7 | 84.6 | 77.6 | 80.2 | 75.9 | 88.4 | 67.8 |
| RoBERTa，第2层 | 85.2 | 79.4 | 88.7 | 83.0 | 83.3 | 78.8 | 90.9 | 71.8 |
| RoBERTa，第3层 | 89.3 | 85.7 | 90.6 | 86.9 | 87.0 | 84.1 | 94.3 | 72.6 |
| RoBERTa，第4层 | 90.2 | 88.7 | 91.8 | 88.7 | 86.2 | 86.4 | 94.5 | 74.5 |
| RoBERTa，第5层 | 91.4 | 89.1 | 92.9 | 90.5 | 89.0 | 87.1 | 95.5 | 74.5 |
| RoBERTa，第6层 | 93.4 | 91.3 | 91.9 | 91.4 | 88.9 | 86.8 | 95.0 | 75.3 |
| RoBERTa，第7层 | 93.9 | 90.5 | 91.8 | 90.4 | 88.2 | 86.9 | 94.6 | 74.7 |
| RoBERTa，第8层 | 93.9 | 91.1 | **93.4** | 92.3 | 88.0 | 87.2 | 94.4 | 75.9 |
| RoBERTa，第9层 | 94.3 | 90.6 | 92.5 | 92.1 | 89.4 | 88.0 | **95.7** | 74.7 |
| RoBERTa，第10层 | 94.4 | **92.0** | 93.3 | **92.3** | **89.9** | 88.1 | 95.0 | 75.1 |
| RoBERTa，第11层 | **95.3** | 91.5 | 93.3 | 89.4 | 88.8 | **88.2** | 95.2 | **76.0** |
| RoBERTa，第12层 | 94.5 | 91.1 | 92.7 | 88.3 | 87.3 | 87.9 | 95.3 | 74.8 |

表7：ELMo、BERT和RoBERTa各个层级在二元语言可接受性探测任务上的准确率结果。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **准备工作** | **Artordet** | **Vform** | **Nn** | **Wchoice** | **Trans** | **SVA** | **Worder** |
| ELMo，0层 | 23.2 | 14.3 | 22.3 | 9.8 | 21.8 | 10.2 | 18.4 | 29.6 |
| ELMo，第1层 | 56.5 | **42.6** | 51.8 | 82.0 | 72.0 | **69.4** | 30.6 | 55.1 |
| ELMo，第2层 | **68.0** | 34.2 | **55.4** | **85.9** | **73.0** | 42.8 | **49.2** | **62.7** |
| BERT，0层 | 24.1 | 39.1 | 66.7 | 58.7 | 62.3 | 56.4 | 63.6 | 17.5 |
| BERT，第1层 | 56.6 | 33.9 | 66.9 | 59.3 | 69.4 | 71.1 | 54.4 | 13.1 |
| BERT，第2层 | 58.7 | 27.4 | 75.8 | 58.4 | 76.3 | 83.3 | 60.0 | 34.1 |
| BERT，第3层 | 64.5 | 55.2 | 56.2 | 62.4 | 79.3 | 83.0 | 64.2 | 67.8 |
| BERT，第4层 | 68.9 | 54.1 | 69.2 | 62.9 | 81.7 | 66.0 | 67.3 | 59.7 |
| BERT，第5层 | 67.4 | 52.4 | 76.9 | 60.8 | 83.8 | 80.7 | 62.2 | 62.3 |
| BERT，第6层 | 68.2 | 51.5 | 76.5 | 58.7 | 84.9 | **83.9** | 71.7 | 66.9 |
| BERT，第7层 | 70.4 | **52.3** | 93.0 | 61.8 | 82.8 | 81.9 | 61.3 | 61.2 |
| BERT，第8层 | 69.9 | 51.7 | 93.0 | 65.4 | 80.2 | 80.2 | 60.9 | **63.9** |
| BERT，第9层 | **71.7** | 48.0 | 91.6 | **85.3** | **84.9** | 79.6 | 59.6 | 62.2 |
| BERT，第10层 | 70.7 | 50.4 | 90.5 | 80.5 | 82.3 | 78.2 | 92.4 | 58.7 |
| BERT，第11层 | 70.1 | 49.2 | **96.3** | 80.5 | 81.0 | 80.7 | 90.5 | 60.3 |
| BERT，第12层 | 71.4 | 50.5 | 86.7 | 79.8 | 79.1 | 81.6 | **93.2** | 58.8 |
| RoBERTa，0层 | 44.8 | 26.5 | 74.8 | 62.8 | 71.3 | 71.1 | 61.7 | 14.3 |
| RoBERTa，第1层 | 68.3 | 12.1 | 90.7 | 62.5 | 80.9 | 75.9 | 93.5 | 48.9 |
| RoBERTa，第2层 | 69.9 | 35.3 | 71.0 | 61.9 | 83.9 | 84.1 | 60.5 | 58.2 |
| RoBERTa，第3层 | 71.9 | 54.4 | 92.2 | 60.7 | 85.5 | 84.4 | **96.2** | 59.3 |
| RoBERTa，第4层 | 71.2 | 48.9 | 92.0 | 83.3 | 85.6 | **85.3** | 95.9 | 60.8 |
| RoBERTa，第5层 | **71.9** | **53.6** | 92.5 | 84.9 | **88.5** | 83.9 | 95.3 | 61.2 |
| RoBERTa，第6层 | 70.2 | 52.9 | 92.5 | 87.0 | 87.3 | 83.9 | 95.7 | 59.0 |
| RoBERTa，第7层 | 70.6 | 50.6 | 92.1 | 87.8 | 87.2 | 83.9 | 94.8 | 58.4 |
| RoBERTa，第8层 | 71.6 | 51.5 | 92.2 | **89.5** | 87.0 | 79.6 | 95.2 | 58.8 |
| RoBERTa，第9层 | 71.3 | 53.2 | 91.9 | 87.7 | 86.7 | 81.3 | 95.8 | 61.1 |
| RoBERTa，第10层 | 69.6 | 50.3 | **92.8** | 86.8 | 87.1 | 78.8 | 96.0 | **64.2** |
| RoBERTa，第11层 | 69.3 | 49.6 | 92.7 | 88.4 | 86.5 | 75.6 | 95.5 | 62.0 |
| RoBERTa，第12层 | 69.6 | 48.9 | 90.1 | 86.8 | 84.9 | 79.6 | 94.1 | 62.8 |

表8：ELMo、BERT和RoBERTa各层定位误差位置的精度结果。

层1层2层3层4层5层6层7层8层9层10层11层12层。

[CLS]

11

攻击

,

逝世

对于

的

平面

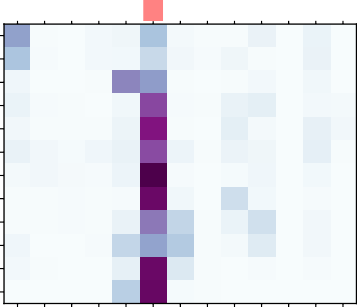
崩溃

靠近

水牛城

.

[SEP]

准备工作错误

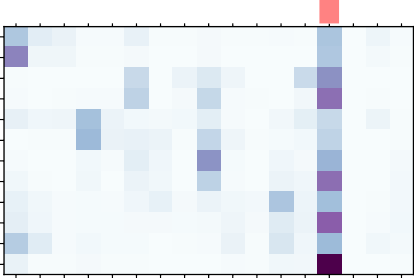
[CLS]

饮食

消化

[SEP]

层1层2层3层4层5层6层7层8层9层10层11层12层。

ArtOrDet错误

[CLS]

那

可能

帮助

减少

的

风险

的

a

银行

运转

出

的

的

资本

.

[SEP]

层1层2层3层4层5层6层7层8层9层10层11层12层。

还有

之后

他们

的

食物

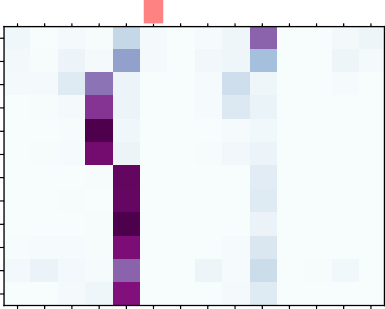
,

他们

的

食物

.

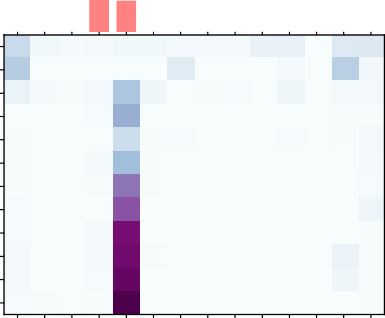
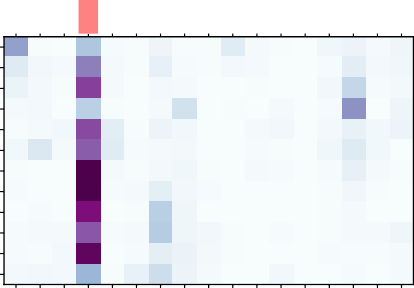
Vform错误

证书

消费者

可持续

层1层2层3层4层5层6层7层8层9层10层11层12层。

Nn错误

[CLS]

[CLS]

他的

它

只是

有

对手

,

一直

执政的

甚至

右翼

争论不休

候选人

该

罗德里戈

人

阿维拉

是

,

制作

不久

它

之后

更差

认输

打败

.

.

[SEP]

[SEP]

层1层2层3层4层5层6层7层8层9层10层11层12层。

[CLS]

我们

需要

a

新的

形成

的

到

帮助

到

选择

海鲜

.

[SEP]

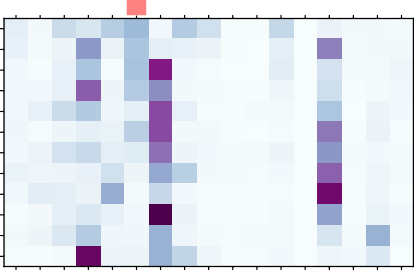
第1层第2层第3层第4层第5层

[CLS]

他

SVA错误

层1层2层3层4层5层6层7层8层9层10层11层12层。

第1层第2层第3层第4层第5层

[CLS]

它

顺序错误

是

在...中

的

小型

百分比a

的

俾格米人

谁

可以

阅读

不过

撰写

.

[SEP]

会

做

鲁尼

's

开始

会议

与

的

世间

's

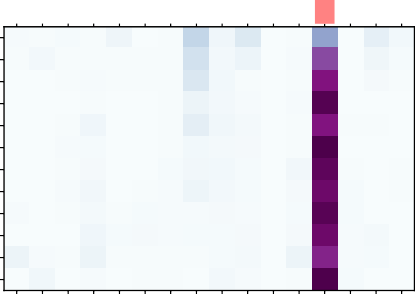
最

魅力四射

团队

.

[SEP]

层6层6

第7层第8层第9层第10层第11层第12层

传输错误

第7层第8层第9层第10层第11层第12层

Wchoice错误

图6：自我注意力层的注意力权重可视化。ﬁgure代表一个句子与specifiﬁc错误类型。一个句子中的错误用红色高亮显示。每一列代表自注意力层建立在BERT的一层。

0.4

Acc.

0.2

预备

0.6

0.4

Acc.

0.2

sva

0.0

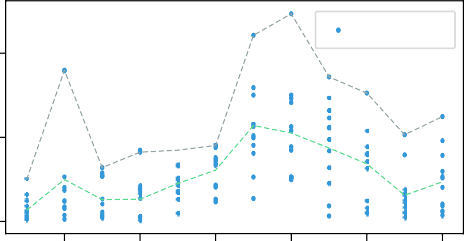
0.3

0.2

Acc.

0.1

2 4 6 8 10 12



10

10

收件人：负责人

10

7

4

6

6

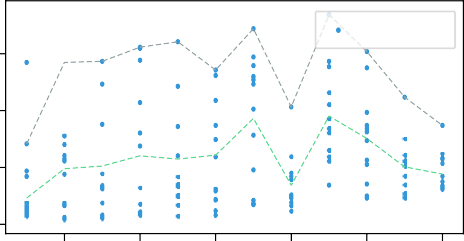
12 4

6

3

2

偈语



3

7

12 2

7

2

收件人：7号首长

3

1

6

2

3

0.0

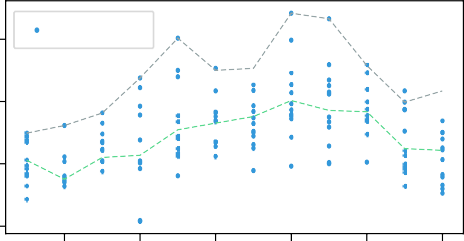
0.6

0.4

Acc.

0.2

2 4 6 8 10 12



2

6

Attn Head5

9

12 12

4

5

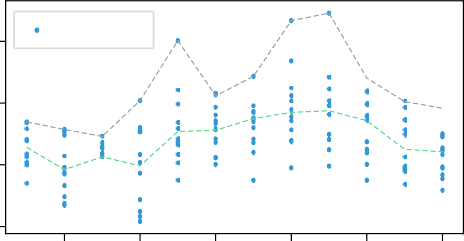
1

12

4

5

vform



2

6

收件人：负责人

5

4

12

9

10

6

4

12

5

5

0.0

0.4

0.3

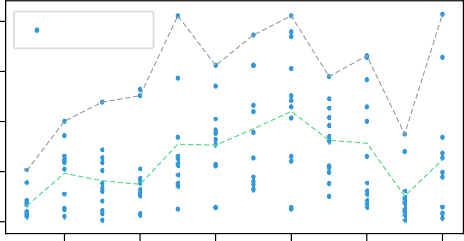
Acc.

0.2

0.1

2 4 6 8 10 12

转运



10

1

3

收件人：负责人

3

6

11

1

5

4

2

6

5

0.0

0.6

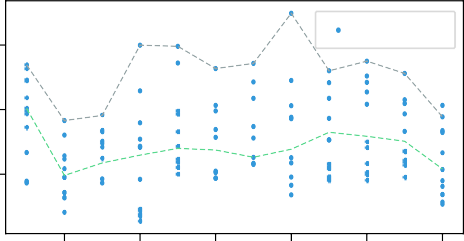
0.4

Acc.

0.2

2 4 6 8 10 12

nn



11

11 1

收件人：负责人

2

1

6

6

8

11

3

11

6

0.0

0.6

0.4

Acc.

2 4 6 8 10 12

选择

0.6

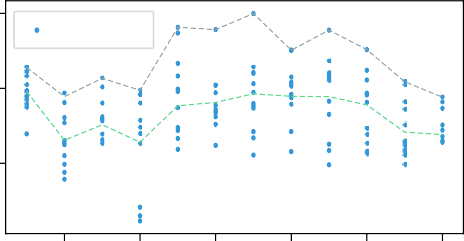
0.4

Acc.

2 4 6 8 10 12

词人

0.2



收件人：负责人

110

5

6

210

7

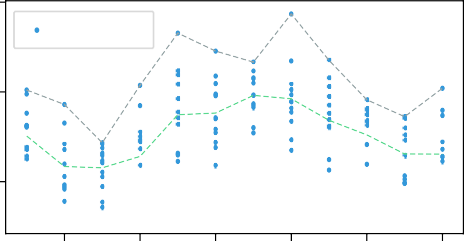
7

12

2

10

6



4

Attn Head7

8

4

9

12

9

11

2

10

10

1

0.2

2 4 6 8 10 12 2 4 6 8 10 12

图7：BERT的每个注意头定位每种类型错误的可视化。ﬁgure中的一个点代表一个注意头的性能。顶部的灰色线代表每层中表现最好的头（用它的编号注释）。中间的绿线代表这一层中所有头的平均表现。