DLSF:基于双重语义过滤的文本对抗攻击方法

熊熙, 丁广政, 王娟, 张帅

引用本文

熊熙, 丁广政, 王娟, 张帅. DLSF:基于双重语义过滤的文本对抗攻击方法[J]. 计算机科学, 2025, 52(10): 423-432.

XIONG Xi, DING Guangzheng, WANG Juan, ZHANG Shuai. DLSF:A Textual Adversarial Attack Method Based on Dual-level Semantic Filtering [J]. Computer Science, 2025, 52(10): 423-432.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700202

DLSF:基于双重语义过滤的文本对抗攻击方法

熊 熙[1,2,3] 丁广政[1,2,3] 王 娟[1,2,3] 张 帅[4]
1 成都信息工程大学网络空间安全学院(芯谷产业学院) 成都 610225
2 先进密码技术与系统安全四川省重点实验室(芯谷产业学院) 成都 610225
3 先进微处理器技术国家工程研究中心(工业控制与安全分中心) 成都 610225
4 北京理工大学信息与电子学院 北京 100081
(flyxiongxi@gmail.com)

摘 要 在商业应用领域,基于深度学习的文本模型发挥着关键作用,但其亦被揭示易受对抗性样本的影响,例如通过在评论中夹杂混肴词汇以使模型做出错误响应。好的文本攻击算法不仅可以评估该类模型的鲁棒性,还能够检测现有防御方法的有效性,从而降低对抗性样本带来的潜在危害。鉴于目前黑盒环境下生成对抗文本的方法普遍存在对抗文本质量不高且攻击效率低下的问题,提出了一种基于单词替换的双重语义过滤(Dual-level Semantic Filtering,DLSF)攻击算法。其综合了目前存在的候选词集合获取方法,并有效避免了集合中不相关单词的干扰,丰富了候选词的类别和数量。在迭代搜索过程中采用双重过滤的束搜索策略,减少模型访问次数的同时,也能保证获取到最优的对抗文本。在文本分类和自然语言推理任务上的实验结果显示,该方法在提升对抗文本质量的同时,显著提高了攻击效率。具体来说,在IMDB数据集上的攻击成功率高达99.7%,语义相似度达到0.975,而模型访问次数仅为TAMPERS的17%。此外,目标模型在经过对抗样本进行对抗增强训练后,在MR数据集上的攻击成功率从92.9%降至65.4%,进一步验证了DLSF有效提升了文本模型的鲁棒性。

关键词:文本对抗攻击;黑盒攻击;束搜索;鲁棒性;文本模型
中图分类号 TP391

DLSF:A Textual Adversarial Attack Method Based on Dual-level Semantic Filtering

XIONG Xi[1,2,3],DING Guangzheng[1,2,3],WANG Juan[1,2,3] and ZHANG Shuai[4]
1 School of Cybersecurity(Xin Gu Industrial College),Chengdu University of Information Technology,Chengdu 610225,China
2 Advanced Cryptography and System Security Key Laboratory(Xin Gu Industrial College),Chengdu 610225,China
3 SUGON Industrial Control and Security Center,Chengdu 610225,China
4 School of Information and Electronics,Beijing Institute of Technology,Beijing 100081,China

Abstract In the field of commercial applications,deep learning-based text models play a crucial role but are also susceptible to adversarial samples,such as the incorporation of confusing vocabulary into reviews leading to erroneous model responses. A strong attack algorithm can assess the robustness of such models and test the effectiveness of existing defense methods,thereby reducing potential harms from adversarial samples. Considering the prevalent issues of low-quality adversarial texts and inefficient attack methods in black-box settings,this paper proposes a dual-level semantic filtering attack algorithm based on word substitution. This algorithm amalgamates existing methodologies for assembling candidate word sets,effectively eliminates interference from irrelevant words,and thereby enriches the variety and quantity of candidate words. It employs a dual-filter beam search strategy during the iterative search process,which not only reduces the frequency of model access,but also guarantees the acquisition of optimal adversarial texts. Experimental results on text classification and natural language inference tasks demonstrate that this method significantly enhances the quality of adversarial texts and attack efficiency. Specifically,the attack success rate on the IMDB dataset reaches 99.7%,semantic similarity reaches 0.975,with the number of model accesses being only 17% of those required by TAMPERS. Furthermore,after adversarial augmentation training with adversarial samples,the target model's attack success rate on the MR dataset decreases from 92.9% to 65.4%,further confirming that DLSF effectively enhances the robustness of the target model.

Keywords Textual adversarial attack,Black-box attack,Beam search,Robustness,Text model

到稿日期:2024-10-10 返修日期:2025-03-15
基金项目:四川省科技计划项目(2024NSFSC2043,2024NSFSC1744,2024NSFSC1185);教育部人文社会科学研究基金(22YJAZH120)
This work was supported by the Sichuan Province Science and Technology Program(2024NSFSC2043,2024NSFSC1744,2024NSFSC1185) and Foundation for Humanities and Social Sciences of Ministry of Education of China(22YJAZH120).
通信作者:王娟(jjmao2009@163.com)

1 引言

深度学习模型虽然在可解释性方面存在局限，但其凭借强大的建模能力，已在计算机视觉[1]、自然语言处理[2]、语音识别[3]等多个领域实现迅猛发展。深度神经网络的稳定性和鲁棒性亦成为研究焦点。Szegedy 等[4]首次提出对抗样本攻击理论，该攻击通过在输入数据中添加细微的扰动，致使模型预测失误。此攻击手段最初在视觉领域[5-7]受到广泛关注，并在攻击与防御手段[8]上均取得显著进展。然而，文本模型亦面临对抗性样本的威胁，如通过略微修改低质量有害短语即可规避谷歌的有害评论检测系统[9]，这揭示了文本模型在鲁棒性与可解释性方面的不足。此外，恶意软件检测、舆情分析等关键应用系统均依赖于文本模型。因此，深入研究文本领域的对抗攻击具有重要意义。

根据攻击实施环境的不同，对抗文本攻击可细分为白盒攻击和黑盒攻击两大类。在黑盒攻击场景中，攻击者除了能够获得模型的输出结果外，无法获取模型参数、激活单元、梯度等内部信息。Papernot 等[10]尝试应用目标模型的前向导数来预估修改后的输入文本对模型输出的影响。然而，由于文本数据的离散性和高度易感知性，此方法产生的对抗文本常常缺乏流畅性，并可能包含语法错误。因此，视觉领域的相关方法并不适合直接应用于文本攻击场景。对此，研究者逐步将关注点转向基于单词级替换的文本攻击方法，该领域研究的核心问题是确保对抗文本的质量和攻击效率[11-12]。这意味着对抗文本需要具备一定的语句流畅性，并保持完整的语义信息；同时，还要求整个攻击流程简洁高效，减少对目标模型的查询次数。本文将基于单词替换的成功攻击归结为两大关键步骤。(1)候选词搜索空间的构建：每个待替换的单词都对应一个候选词集合，所有单词的候选集合构成一个离散的搜索空间，这一空间的妥善构建是确保对抗文本质量的关键。(2)搜索策略的设计：搜索策略利用上述搜索空间来筛选出符合要求的对抗文本，常见的搜索策略包括利用模型的梯度信息、贪心算法、遗传算法等，这些策略的目的是在提高对抗文本质量的同时确保一定的搜索效率。具体步骤如图 1 所示，候选词搜索空间由原始文本中各待替换单词的候选词集合构成，在每轮迭代仅替换一个单词的前提下，搜索策略从这些候选词的组合中选取一个满足约束条件的最佳组合作为原始文本的对抗文本。

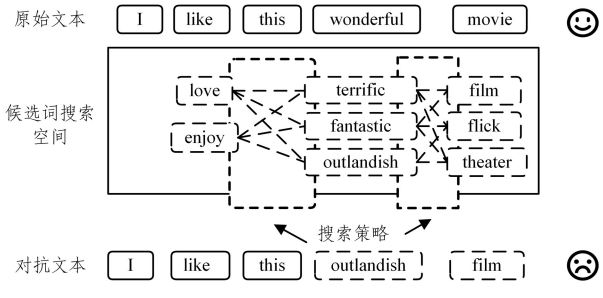


图 1 黑盒文本攻击的关键点：候选词搜索空间与搜索策略

Fig. 1 Key points of black-box textual attacks: candidate word search space and search strategy

本文在黑盒环境下提出一种双重语义过滤(DLSF)文本攻击方法。该方法首先通过整合不同来源的候选词，扩充了每个单词的候选词集合，接着在候选对抗文本生成的过程中使用束搜索(Beam Search)策略来扩大搜索范围。此外，该方法还在候选单词和候选文本两个不同的层面进行了过滤，以确保对抗文本在质量和攻击效率方面的最优化。本文的主要贡献包括以下 3 个方面：

(1)在构建输入文本中每个单词的候选词集合时，综合运用了外部词典、预训练词向量以及掩码语言模型 3 种资源，并有效解决了以往依赖掩码语言模型获取候选词时易产生不相关及大量无意义拼接单词的问题。

(2)提出了在束搜索过程中实施双重筛选的策略，这一搜索策略不仅减少了冗余候选文本的数量，还显著提升了迭代搜索的效率。

(3)在文本分类和自然语言推理(Natural Language Inference，NLI)两项任务中均取得了显著成效，不仅在攻击成功率、扰动率以及语义相似度等关键指标上保证了对抗文本的质量，还有效减少了对目标模型的访问次数。此外，通过对抗训练的实验验证，进一步证实了该攻击方法生成的对抗文本的有效性，即其能通过数据增强的方式显著提升目标模型的鲁棒性。

2 相关工作

在黑盒环境下，针对目标模型的文本攻击可细分为硬标签(Hard-Label)[11,13]和软标签(Soft-Label)[12,14-15]两大场景。硬标签攻击中，目标模型仅提供关于输入文本的决策结果(如 0 或 1)，而软标签攻击则提供文本的置信度评分，进而揭示了语义层面的信息。Yoo 等[16]将对抗文本攻击视为目标函数、单词扰动方式、语义约束以及搜索算法的有机结合。因此，候选词集合的获取和恰当的搜索策略对于攻击流程至关重要。

2.1 贪心搜索

该策略在每次迭代中遍历所有候选文本，并选择对攻击成功最有利的单一候选项，这通常依赖于相似性度量或模型输出的置信度。例如，BERT-Attack[14]，TextFooler[17]，PWWS[18]等方法均采用贪心搜索策略进行对抗文本攻击，但其获取候选词的方式各异，分别通过预训练好的词向量、预训练词向量和外部词典实现。首先，这些方法获取候选词的来源较为单一，导致生成的对抗文本质量一般。其次，尽管其迭代搜索的效率较高，但也很可能导致其无法充分探索候选词对应的全局搜索空间。

2.2 启发式搜索

在对抗文本攻击的领域内，启发式方法通常是指基于种群的优化[19-20]方法。与贪心搜索相比，此类方法具有更大的搜索范围，生成的对抗文本更流畅且隐蔽性更强。Alzantot 等[19]率先采用遗传算法(Genetic Algorithm，GA)处理单词替换过程中的语义丢失和句子流畅性问题。此外，遗传算法不受软硬标签场景的限制。例如，在硬标签场景下，HLGA[13]通过遗传算法增强搜索过程中候选文本的语义相似度。而在软标签场景下，TAMPERS[12]利用遗传算法降低对抗文本的扰动率。具体而言，TAMPERS 先通过贪心搜索获得初始对抗文本，以有效控制搜索空间的扩张，然后采用遗传算法对其进行优化，最终减少了对抗文本中所需替换的单词数量。尽管基于种群的搜索策略是一种全局的搜索策略，但其搜索设计流程涉及种群初始化、交叉和变异等复杂操作，导致其可控性较差。

2.3 束搜索

束搜索在贪心搜索的基础上引入束宽 b(Beam Width)的设置，每次迭代保留 b 个候选项，从而在保持搜索流程简洁的同时，相较于贪心搜索扩大了搜索范围。BeamAttack[15] 首先简单综合了上述 3 种候选词来源，随后采用束搜索策略在每次迭代过程中扩大搜索范围，最终获得了较高的攻击成功率。然而，该方法未能充分考虑到使用较多候选词来源，以及束搜索策略在扩大搜索范围的同时也会产生大量冗余候选文本，从而增加不必要的模型访问次数。TABS[21] 仅将束搜索应用于攻击代码类目标模型，并未成功将其迁移至文本领域。

综上所述，目前对抗文本攻击面临的主要挑战包括：(1)候选词来源较为单一，导致候选词集合资源匮乏，难以保证对抗文本的高质量；(2)常用的搜索策略在扩大搜索范围的同时，难以兼顾攻击算法的效率与对抗文本的质量。例如，贪心搜索虽效率较高，但每次迭代仅考虑单一的最优候选项，可能导致最终生成的对抗文本并非搜索空间中的最优解；而启发式搜索虽通过种群基础扩大了搜索范围，但其设计过程复杂，需精细调整种群大小、适应度函数等参数或操作，且搜索范围的扩大亦会增加模型访问次数，最终影响攻击效率。

3 问题定义

对于任意指定的输入文本，假定其由 N 个单词组成，即输入 $X=\{w_1\cdots,w_i,\cdots,w_N\}$，同时，存在一个经过微调的文本模型 F，使得输入 X 在模型 F 中对应的标签为 y，形式化表示为 $F:X\rightarrow y$。在软标签场景中，每个生成的对抗文本通过向目标模型 F 发起查询，均能获得一个相应的概率值。

对抗文本 X_{adv} 与原始输入 X 相对应，其构成为 $X_{adv}=\{w_1',w_2',w_3',\cdots,w_i',\cdots,w_N'\}$，其中 $w_i'\in D_i$，而 D_i 由原始单词 w_i 及其候选词集合 C_i 组成，即 $D_i=\{w_i\}\cup C_i$。候选词集合 C_i 包含了 w_i 的所有候选词，形式为 $C_i=\{c_i^1,c_i^2,\cdots,c_i^j,\cdots,c_i^n\}$。因此，对抗文本中的每个单词 w_i 要么保持原始形式，要么从其候选词集合中选取一个候选词替换。下面是对文本攻击的形式化定义：

$$\begin{aligned}&\exists X_{adv}\in S(X_{adv})\\&\text{s.t. } F(X_{adv})\neq F(X)\end{aligned}\tag{1}$$

其中，$S(X_{adv})=\{X_{adv}\}$，集合 $S(X_{adv})$ 包含了输入文本 X 所有潜在的对抗文本，攻击的目标是至少寻求一个 X_{adv}，使其对应的输出标签与 X 的输出标签不一致。

然而，仅保证生成的对抗文本 X_{adv} 与原始文本 X 标签不一致是不够的，还需要在此基础之上施加一定的限制。TextFooler[17] 提出了一个合格的对抗文本应该满足 3 个关键条件：(1)人类预测一致性，即从人类视角出发，原始文本和对抗文本对应的模型预测结果应保持一致；(2)语义相似性，对抗文本与原始文本之间必须保持一定程度的相似性，且这种相似性不能过低；(3)语句流畅性，对抗性样本应当语句通顺，无语法错误。为此，除了使用攻击成功率(Attack Success Rate，ASR)来衡量攻击方法的效果，本文引入的额外限制性指标包括语义相似度(Semantic Similarity，SIM)、单词扰动率(Perturbation Rate of Word，PRW)。此外，在攻击过程中，还对模型的查询次数(Number of Queries，Queries)进行了限制，以确保攻击的效率。

$$\begin{aligned}&\text{mininum } queries\\&\text{s.t. } sim(X,X_{adv})\geqslant\epsilon,prw(X,X_{adv})<\delta\end{aligned}\tag{2}$$

具体地，在对目标模型的查询次数尽可能少的情况下，也需保证对抗文本与原始文本之间有一定的语义相似度以及较低的单词扰动率，这样的方法设计旨在平衡攻击的隐蔽性与有效性，确保对抗文本在达到预定攻击目的的同时，还具备较高的自然性且难以被察觉。其中，$sim(\cdot,\cdot)$ 是用来计算两个句子之间相似度的函数；$prw(\cdot,\cdot)$ 是一个衡量标准，根据 X_{adv} 与 X 之间不同单词的个数来计算扰动率。

4 攻击算法 DLSF

DLSF 算法适用于目标模型输出为软标签的场景，与 TextFooler[17] 等相比，该方法不仅致力于提升对抗样本的质量，还注重保持高效的搜索效率。

DLSF 的实施流程如图 2 所示，具体分为以下 3 个阶段。

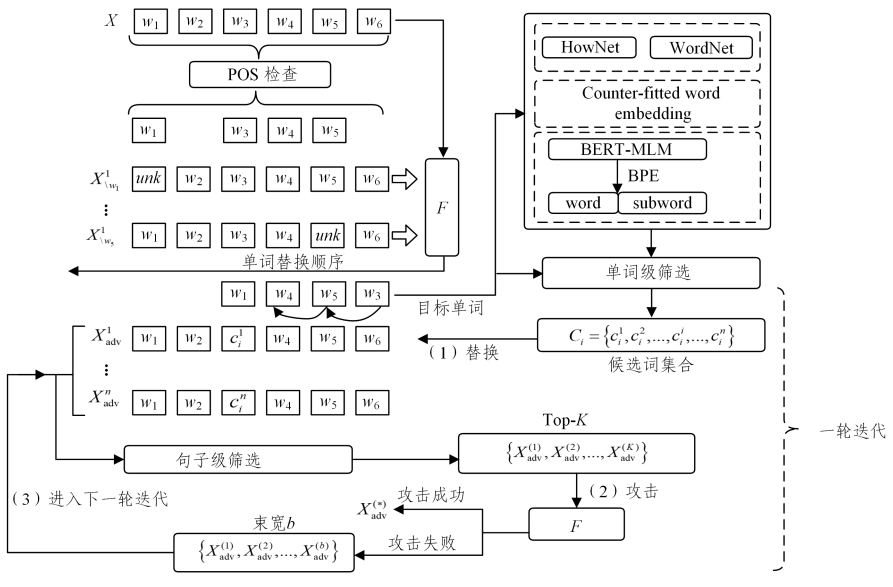


图 2　DLSF 攻击算法总体流程

Fig. 2　Overall process of DLSF attack algorithm

(1)确定单词替换的优先级:首先对文本 X 中的每个单词进行词性检查,筛选出需要替换的目标单词,然后根据目标模型 F 评定的重要性分数对这些单词进行排序,以确定它们的替换顺序。

(2)构建单词的候选词集合:对于每个目标单词,从 3 种不同的候选词资源中获取相应的候选词集,并通过单词级别的筛选过程,获得其最终的候选词集。

(3)过滤优化的束搜索策略:通过替换目标单词得到的候选文本集合,经过初步的句子级别筛选后保留固定的 K 个候选项。如果攻击未成功,仅保留束宽 b 个候选文本进入下一轮攻击。在整个攻击流程中,每次迭代仅替换一个单词。

4.1 确定单词替换顺序

软标签场景下,可以通过目标模型的输出来计算输入文本中每个单词的重要性分数,如第 3 章给定由 N 个单词组成的输入文本 $X=\{w_1,\cdots,w_i,\cdots,w_N\}$,则有 $X_{\setminus w_i}=X\setminus\{w_i\}=\{w_1,\cdots,w_{i-1},[\text{UNK}],w_{i+1},\cdots,w_n\}$,即将 w_i 用[UNK]这个特殊的标记进行替换,w_i 对应的重要性分数 I_{w_i} 具体计算式为:

$$I_{w_i}=P(y_{\text{true}}\mid X)-P(y_{\text{true}}\mid X_{\setminus w_i}) \tag{3}$$

其中,y_{true} 为原始输入文本 X 对应的标签。与软标签场景下工作的 TextFooler[17] 和 TAMPERS[12] 等方法在计算单词重要性分数时遍历输入文本中的所有单词并计算其分数值不同,DLSF 只关注内容性单词,即动词(verb.)、名词(noun.)、形容词(adj.)和副词(adv.),这种做法直接忽略了停用词(stop words)等类型的单词,减少了攻击过程的迭代轮次,提升了攻击效率。将所有单词按照其重要性分数进行降序排列,即确定输入文本中不同单词间的先后替换顺序。

这种做法在某种程度上类似于图像领域中使用梯度信息[4-5]来添加扰动,然而在黑盒环境下无法直接获取输入中每个单词对应的梯度信息,并以此来决定单词的重要性。因此,使用重要性分数来近似梯度值,对于分数值越大的单词,改变它们就越有可能导致目标模型预测错误。该方法在无法获取模型内部信息的情况下,提供了一种有效的替代方案,以实现离散的文本对抗攻击。

4.2 候选词集合获取与筛选

在替换单词过程中,需从相应候选词集合中选取合适的替换项。例如,若原词为"good",其候选词集合可能包含"great""wonderful""excellent"等。目前,获取单词候选词集合的来源主要有以下 3 类:

(1)基于外部词典[12,18]

在文本攻击领域,常用的外部词典包括 HowNet[22] 和 WordNet[23]。它们类似于同义词词典,但在实际应用中需借助应用程序 API 完成查询操作。

(2)基于预训练好的词向量[11,17]

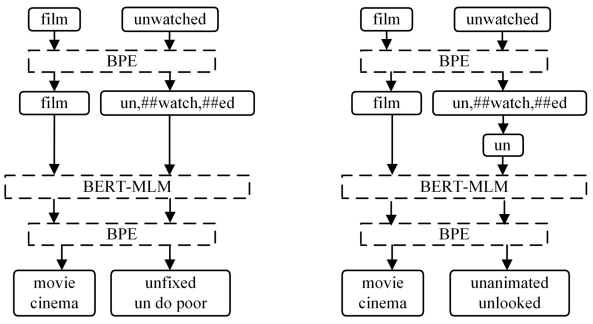
Counter-fitted word embedding[24] 在传统词向量的基础上引入了同义词和反义词的约束机制,有效促进了同义词及反义词的检索。

(3)基于掩码语言模型[14,25]

掩码语言模型(如 BERT-MLM[2])将目标单词替换为特定掩码标记[MASK],利用其预测文本中缺失单词的能力,直接生成目标单词的候选词集合。

根据上述候选词来源的特点,基于外部词典和预训练词向量的方式首先会面临罕见词或未登录词(Out-of-Vocabulary,OOV)的候选词难以获取的问题。其次,这些方法仅在单词级别获取候选词,未能充分考虑待替换词的上下文信息。而掩码语言模型虽考虑到了待替换单词的上下文信息,但难以保持与原始文本的语义相似性。例如,在句子"this film is wonderful"中,"wonderful"被[MASK]替换后,BERT-MLM 可能以较高置信度输出"boring"或"common"等候选词,最终影响对抗文本的质量。

为了克服掩码语言模型在提供候选词时可能导致原始文本语义丧失的问题,DLSF 采纳了 BERT-Attack[14] 中的做法,不使用[MASK]机制来获取候选词,而是通过字节对编码(Byte-Pair Encoding,BPE)对原始文本进行预处理。其与 BERT-Attack 做法的不同之处在于,在处理 BPE 分词的结果时并不考虑子词组合的形式,如图 3 所示。图 3(a)中,在 BERT-Attack[14] 中,输入文本中的某些单词经过 BPE 分割后,形成了子词组合。这些子词组合在经过 BERT-MLM 处理后,可能会产生与原始文本不相关或无意义的拼接单词。为了解决这一问题,如图 3(b)所示,DLSF 仅考虑不含"#"标记的子词,这种做法避免了在将由 BPE 处理的子词组合还原为独立单词时,可能产生大量无意义拼接单词的问题。此外,BPE 的分词和随后的组合操作也会降低生成候选单词的效率。因此,DLSF 不仅解决了掩码语言模型生成候选词时可能丢失原始文本语义的问题,还避免了 BERT-Attack[14] 方法中的候选词集合存在不相关和无意义拼接单词的情况。



(a)考虑子词组合 (b)只考虑子词

图 3 MLM 中的子词处理方式对比

Fig. 3 Comparison of subword processing methods in MLM

经过上述分析,针对第 2 章提出的候选词来源较为单一的问题,DLSF 采取了直接综合上述 3 类候选词资源的方式。通过直接整合 3 种候选词来源获取的候选词集合,不仅考虑到了待替换词的上下文信息,还丰富了候选词的数量和类别。此外,DLSF 还通过基于单词级别的相似度对 3 种方式获取的候选词集合进行综合筛选,确保最终候选词集合中均为高相似度且有意义的单词。经过筛选,总的候选词数量有所减少,从而提高了搜索效率。通过这些改进,能够有效地生成与原词词义相近且符合上下文语境的候选词,详细过程如算法 1 所示。

算法 1　单词候选词集合获取与筛选

输入：原始文本 X，目标单词 tgt_word，掩码语言模型 MLM，BPE 分词器 tokenizer，单词级的相似度过滤阈值 threshold

输出：目标单词的候选词集合 candidate_words

```
1. 初始化 candidate_words 为空
2. S ← getDict(tgt_word)
3. U ← getEmbedding(tgt_word)
4. sub_words = tokenizer(tgt_word)
5. if isSingleWord(sub_words):
6.     V ←getMLM(MLM,sub_words,X)
7. else:
8.  sub_word = drop(sub_words,'#')
9.  V ←getMLM(MLM,sub_word,X)
10. end if
11. all_candidates ← S ∪ U ∪ V
12. for each word in all_candidates:
13.     if sim(tgt_word,word) > threshold:
14.         candidate_words ← word
15.     end if
16. end for
17. return candidate_words
```

算法 1 步骤说明如下：获取来自同义词词典 WordNet 和 HowNet 的候选词集合 S（第 2 行）；获取从 Counter-fitted word embedding 计算得到的候选词集合 U（第 3 行）；使用 BPE tokenizer 对目标单词 *tgt_word* 进行分割操作，得到子词形式 *sub_words*（第 4 行）；若 *sub_words* 为独立单词形式，则 getMLM 函数获取来自 MLM 的候选词集合 V（第 5－6 行）；若 *sub_words* 为子词组合的形式，则 *drop* 函数仅选择不含“#”的子词作为 *sub_word*，同样获取来自 MLM 的候选词集合 V（第 8－11 行）；求 3 种候选词集合 S，U，V 的并集 *all_candidates*（第 11 行）；sim 函数计算目标单词 *tgt_word* 和候选词集合 *all_candidates* 中每个单词的相似度，仅保留相似度值大于 *threshold* 的候选单词集合 *candidate_words*（第 12－16 行）。

4.3　双重过滤的束搜索

在上述步骤中，已经确定了目标单词的先后替换顺序，以及每个目标单词 w_i 对应的候选词集合 $C_i=\{c_i^1,c_i^2,\cdots,c_i^j,\cdots,c_i^n\}$，如图 1 所示，这两个元素构成了一个搜索空间 S，本文的目标是通过束搜索策略在 S 中找到最优的对抗文本。然而，束搜索在每轮迭代中都会保留 b 个候选文本，这将导致对目标模型的访问次数也增加到原来的 b 倍。为了解决这个问题，DLSF 在束搜索的迭代过程中采用了两次筛选的原则。第一次筛选旨在过滤掉语义相似度低的候选文本，以减少对目标模型的访问次数。第二次筛选则是为了确定经过第一次筛选后的候选文本集合中是否存在已经攻击成功的对抗文本。如果存在，则攻击成功；如果不存在，则继续进行下一轮的迭代搜索。通过这种两次筛选的策略，能够在保持搜索效率的同时，有效地找到最优的对抗文本。

两次过滤的执行细节如下：

假设在某次迭代中，被替换的单词为 $w_{(i)}$，其对应的候选词集合 $C_{(i)}=\{c_{(i)}^1,c_{(i)}^2,\cdots,c_{(i)}^m\}$。据此，由二者构建的一个候选文本集合为 $S(X_{\text{adv}})=\{X_{\text{adv}}^1,\cdots,X_{\text{adv}}^i,\cdots,X_{\text{adv}}^m\}$，其中 $X_{\text{adv}}^i=\{w_1,w_2,\cdots,c_{(i)}^i,\cdots,w_n\}$。

（1）第一次过滤

文本攻击的目标是寻找一个最优的对抗文本。候选词集合 $C_{(i)}$ 中含有众多候选词，这会导致 $S(X_{\text{adv}})$ 中包含大量低质量的冗余候选文本。根据第 3 章提到的关于对抗文本的基本要求，从 $S(X_{\text{adv}})$ 中选取与原始输入文本 X 语义相似度[26]最高的 K 个候选文本，形成集合 $S_K(X_{\text{adv}})=\{X_{\text{adv}}^{(1)},X_{\text{adv}}^{(2)},\cdots,X_{\text{adv}}^{(K)}\}$，其中 K 是预设的参数。

（2）第二次过滤

经过第一次过滤，候选文本数量缩减至 K 个，因而对目标模型的访问次数亦限于 K 次，不会随候选文本数量增加而变多。此时，$S_k(X_{\text{adv}})$ 中所有候选文本向目标模型发起查询，以判断攻击是否成功。目标模型 F 输出的置信度 $O_k=\{o_1,o_2,\cdots,o_k\}$ 与 $S_K(X_{\text{adv}})$ 中的候选文本一一对应。如果存在能使目标模型预测出错的样本，则攻击成功；若不存在，则选取 O_k 中置信度最低的 b 个输出 $O_b=\{o_{(1)},o_{(2)},\cdots,o_{(b)}\}$ 及相应的 b 个候选文本 $S_b(X_{\text{adv}})=\{X_{\text{adv}}^{(1)},X_{\text{adv}}^{(2)},\cdots,X_{\text{adv}}^{(b)}\}$，其中 b 个候选文本直接进入下一轮迭代。

该阶段的束搜索策略需要与 DLSF 算法的前两个步骤进行整合，其工作流程如算法 2 所示。

算法 2　束搜索中候选文本的双重过滤优化

输入：目标模型 F，原始文本 X，原始文本 X 的标签 y，束搜索中第一次过滤保留的候选文本数量 K，束搜索束宽 b(K>b)

输出：攻击成功的对抗文本 X_{adv}

```
1. all_candidates ← X
2. perturb_words = getReplaceOrder(X)
3. for eachtgt_word in perturb_words:
4.     candidate_words = getWords(tgt_word)
5.     for each word in candidate_words:
6.         all_candidates ← replace(tgt_word,word)
7.     end for
8.     top_K ← getTOPK(all_candidates,X,K)
9.     succ_candidates = getSucc(F,top_K,y)
10.    if succ_candidates is not None:
11.        X_adv = getOptimal(succ_candidates)
12.        return X_adv
13.    end if
14.    next_b_candidates ← getNext(top_K,b)
15.    all_candidates ← next_b_candidates
16. end for
17. return None
```

算法 2 步骤说明如下：

使用输入文本 X 初始化候选文本集合 *all_candidates*（第 1 行）；根据式（3）计算输入文本 X 中内容性单词的替换顺序（第 2 行）；根据算法 1 得到当前目标单词 *tgt_word* 对应的候选词集合 *candidate_words*（第 4 行）；*replace* 函数使用

candidate_words 中的每个单词替换掉原始输入 X 中的 *tgt_word* 得到候选文本集合 *all_candidates*(第 5－7 行);*getTOPK* 函数根据语义相似度从候选文本集合 *all_candidates* 中得到相似度最高的 K 个候选文本 *top_K*(第 8 行);*getSucc* 函数判断候选文本集合 *top_K* 中是否存在已经攻击成功的候选文本集合 *succ_candidates*(第 9 行);若存在已经攻击成功的候选文本,则 *getOptimal* 函数从集合 *succ_candidates* 选出最优的候选文本,作为最终的对抗文本 X_{adv}(第 10－13 行);该轮攻击失败,getNext 函数根据目标模型 F 输出从候选文本集合 *top_K* 中选出 b 个候选文本 *next_b_candidates* 用于下一轮迭代(第 14－15 行);攻击失败,未成功获得输入 X 的对抗文本(第 17 行)。

4.4 算法时间复杂度分析

DLSF 算法与 BeamAttack[15] 皆采用了束搜索策略,不同之处在于,BeamAttack 并没有算法 2 中第 8 行事先进行一次语义过滤的操作,导致其面向目标模型的时间复杂度为 $O(b \cdot W \cdot T)$,其中 W 为待替换单词的总数,T 代表最大候选词集合中候选词的数量,b 为手动设置的束宽大小。相应地,DLSF 在迭代搜索过程中实施了两轮过滤,因此其时间复杂度降为 $O(K \cdot W)$,此时的搜索效率与贪心策略持平。

5 实验与分析

5.1 数据集与受害目标模型

本文针对文本分类与自然语言推理任务,对 DLSF 进行了评估。在分类任务中,所采用的数据集包括 SST2[27],MR[28],Yelp[29] 和 IMDB[30]。这些数据集的区别在于文本长度,SST2 和 MR 属于句子级的分类数据集,而 Yelp 和 IMDB 则属于文档级分类数据集。在 NLI 任务中,使用到的数据集有 SNLI[31] 和 MNLI[32]。这两个数据集的每个样本均由一对文本(前提和假设)构成,目标模型需判定这对文本的关系是蕴含、中立还是矛盾。相较于 SNLI,MNLI 数据集包含了更多书面化和口语化的文本,如转录文本、流行小说和政府报告。数据集的详细信息如表 1 所列。

表 1 数据集基本信息

Table 1 Basic information of the datasets

	数据集	文本平均长度
分类	SST2	19.1
	MR	20.2
	Yelp	152.9
	IMDB	268.3
推理	SNLI	8.4
	MNLI	12.8

攻击的目标模型是基于上述数据集训练集完成微调的 BERT-base[2] 模型。微调后,目标模型的分类准确率如表 2 所列,并且参照 BERT-Attack[14] 和 TAMPERS[12] 中的做法,从各数据集的测试集中抽取 1 000 个样本进行攻击测试。为确保结果具有可靠性,对每个数据集执行了 5 轮测试,并计算结果的平均值。

表 2 DLSF 与基线的对比实验结果

Table 2 Comparison experimental results of DLSF and Baselines

数据集	攻击方法	准确率/%	*ASR*/%	*PRW*/%	*SIM*	*Queries*
MR	TextFooler	89.6	92.8	19.6	0.83	**117.4**
	BERT-Attack		88.4	15.6	0.85	217.8
	TAMPERS		93.4	9.6	0.86	856.2
	BeamAttack		**96.7**	**9.9**	0.84	672.3
	DLSF		93.4	10.8	**0.88**	282.4
SST2	TextFooler	91.3	90.6	21.6	0.82	**59.8**
	BERT-Attack		91.1	14.3	0.85	255.3
	TAMPERS		**92.7**	10.1	0.85	786.8
	BeamAttack		87.3	**9.1**	0.82	593.6
	DLSF		91.6	11.8	**0.86**	296.2
Yelp	TextFooler	97.1	94.7	10.6	0.88	786.2
	BERT-Attack		93.9	6.7	0.90	**313.7**
	TAMPERS		95.9	4.3	0.89	3092.8
	BeamAttack		96.5	**3.7**	**0.94**	3209.8
	DLSF		**97.2**	5.2	0.93	1251.2
IMDB	TextFooler	89.4	97.1	8.7	0.91	**539.2**
	BERT-Attack		96.5	6.6	0.92	585.3
	TAMPERS		98.2	2.3	0.94	4127.4
	BeamAttack		94.5	2.0	0.97	2876.9
	DLSF		**99.7**	**1.7**	**0.98**	687.3
SNLI	TextFooler	90.4	96.3	18.1	0.55	**54.1**
	BERT-Attack		**97.4**	**9.8**	0.56	93.7
	TAMPERS		94.3	14.9	0.76	273.4
	BeamAttack		96.8	14.3	0.81	110.4
	DLSF		94.1	14.2	**0.83**	80.2
MNLI	TextFooler	85.6	90.4	14.9	0.57	**76.5**
	BERT-Attack		92.0	**6.6**	0.69	121.7
	TAMPERS		91.1	10.4	0.83	354.2
	BeamAttack		**95.4**	12.1	0.86	198.2
	DLSF		92.2	11.9	**0.87**	132.6

注:粗体为最优实验结果;下划线为次优实验结果。

5.2 基线与评估标准

为全面评估 DLSF 的效果,采用了以下基线方法与其进行对比,这些基线方法均以单词替换来实现对抗攻击,但均采用不同的候选词生成方式或搜索策略。

TextFooler[17]:该方法借助 Counter-fitted word embedding 寻找单词的同义词,并通过贪心搜索策略直接生成对抗文本。

BERT-Attack[14]:该方法借助 BERT-MLM 预测每个单词的候选词集合,并通过 BPE 分词技术对单词进行预处理,最终同样采用贪心搜索策略产生对抗文本。

TAMPERS[12]:该方法使用 WordNet 和 HowNet 作为候选词的来源,首先通过贪心搜索为每个样本生成一个初始的对抗文本,随后利用遗传算法进行优化,以尽量减少文本中的扰动单词数量。

BeamAttack[15]:该方法结合了 4.2 节中提到的 3 种方式作为候选词的来源,并采用束搜索策略生成对抗文本。

为了评估对抗文本的质量与攻击效率,根据第 3 章中所述标准,从以下 4 个方面进行评估:攻击成功率 ASR、语义相似度 SIM、单词扰动率 PRW 以及目标模型的查询次数 Queries。依照 TextFooler 和 TAMPERS 的设置,采用通用句子编码器[33](Universal Sentence Encoder,USE)来量化原始文本与对抗文本间的语义相似度。各个指标的具体计算式如下:

$$ASR=\frac{N_{adv}}{N} \tag{4}$$

其取,N 表示样本总数量,本文中 $N=1\,000$;N_{adv} 表示算法攻击成功的样本数量。

$$SIM=1-\arccos\left(\frac{u\cdot v}{\|u\|\|v\|}\right) \tag{5}$$

其中，u，v 分别为通用编码器对原始文本与对抗文本编码后的向量表示形式，并采用基于角度距离来计算二者的相似度。

$$PRW=\frac{diff(X,X_{\text{adv}})}{n} \tag{6}$$

其中，$diff(\cdot,\cdot)$为求两段文本中不同单词的个数，n 为该段文本中总的单词个数。

$$Queries=\frac{\sum_{i=1}^{N_{\text{adv}}}q_i}{N_{\text{adv}}} \tag{7}$$

其中，q_i 为第 i 个文本的目标模型访问次数。

5.3　对比实验结果

DLSF 与 4 个基线方法的对比实验结果如表 2 所列。

为确保实验的公正性，所有基线方法与 DLSF 均针对同一目标模型进行攻击。在攻击效率方面，采用贪心搜索的 TextFooler 和 BERT-Attack 在各数据集上的查询次数最低。然而，它们的候选词集合的来源较为有限，导致在攻击成功率和语义相似度方面的表现不尽人意。在对抗文本的质量方面，TAMPERS 和 BeamAttack 的表现更为出色。在 Yelp 和 IMDB 数据集上，DLSF 则与两者的表现相近。尽管 TAMPERS 和 BeamAttack 在生成高质量对抗文本方面表现良好，但它们在目标模型上的查询次数显著增加。例如，在 IMDB 数据集上，TAMPERS 的查询次数为 TextFooler 的 8 倍，而 BeamAttack 的查询次数约为 TextFooler 的 5 倍。相比之下，DLSF 在保持相似表现的同时，通过在束搜索策略中对候选文本进行筛选，有效控制了查询次数，使其与 TextFooler 和 BERT-Attack 处于同一水平，平均查询次数分别为 539.2，585.3 和 687.3。此外，实验结果还表明，TAMPERS 更擅长攻击分类任务。这是由于 SNLI 与 MNLI 属于短文本序列，TAMPERS 无法充分利用遗传算法来降低扰动率，因此在 SNLI 和 MNLI 的推理任务上表现一般，而 DLSF 在分类任务和自然语言推理任务上都展现出了良好的性能。

虽然 BeamAttack 在 MR，SNLI，MNLI 数据集上的攻击成功率略高于 DLSF，但在语义相似度和查询次数这两个关键指标上，BeamAttack 的表现不及 DLSF。综合各项评估指标来看，DLSF 在综合表现上优于其他方法。这表明 DLSF 在处理不同类型的任务时，能够有效平衡攻击成功率、语义相似度、扰动率和查询次数，从而在实际应用中更具有广泛的应用潜力。

5.4　参数分析与选取

在 DLSF 攻击过程中，需要设定若干关键参数，包括单词候选集合筛选时的相似度阈值 *thgreshold*、束搜索的束宽 b，以及束搜索中第一次筛选的候选文本数量 K。在进行与基线方法的对比实验之前，必须确定一组合理的参数组合，以确保 DLSF 方法的实验效果。

鉴于 BeamAttack[15] 在实验操作中设定束宽 $b=10$，为了维持公平性与一致性，DLSF 中的束宽 b 也被直接设定为 10。紧接着为了避免对所有 *threshold* 和 K 的参数组合进行直接遍历，首先忽略了对 K 的设定，将 *threshold* 的值分别设定为 0.1，0.2，0.3，0.4 和 0.5，并以 MR 作为攻击的目标数据集，获取了每个具体参数值对应的实验结果，如表 3 所列。随后，将候选文本数量 K 分别设定为 128，192，256，320 和 384，与上述 *threshold* 的最优参数值组合，再次在 MR 数据集上进行实验测试，实验结果如表 4 所列，选取最优实验结果作为 K 的取值。

表 3　*threshold* 参数研究

Table 3　Exploration of *threshold*

threshold	*ASR*/%	*PRW*/%	*SIM*	*Queries*
0.1	95.5	9.8	0.893	511.6
0.2	94.8	10.8	0.888	474.1
0.3	93.7	11.0	0.887	402.6
0.4	92.4	11.4	0.884	383.1
0.5	91.7	12.6	0.882	350.3

表 4　K 参数研究

Table 4　Exploration of K

K	*ASR*/%	*PRW*/%	*SIM*	*Queries*
128	92.5	11.1	0.889	193.8
192	92.6	10.7	0.886	247.2
256	93.1	10.7	0.884	289.2
320	93.4	10.8	0.885	356.1
384	93.8	10.6	0.882	398.3

从表 3 的数据中可以明显看出，较大的 *threshold* 值会降低攻击成功率，而较小的值则会导致模型的访问次数增多。为了在这两者之间找到一个平衡点，选择将 *threshold* 设为 0.3。在确定 *threshold* 为 0.3 之后，对参数 K 进行了详细的案例测试。根据表 4 的结果，可以看出 K 值对攻击成功率、扰动率以及模型访问次数均有直接影响。基于此，选择 $K=256$，以确保 DLSF 在实现较高攻击成功率的同时，也能保持较低的扰动率和模型访问次数。因此，在对比实验中，将相似度阈值 *threshold* 确定为 0.3，候选文本数量 K 设定为 256，而束搜索的束宽 b 维持在 10，这是经过综合参数案例分析后得出的选择。

5.5　消融实验

为了验证 DLSF 方法中各个步骤的有效性，将 MR 数据集作为攻击目标，通过消融实验来验证 BPE 分词操作中子词选择的影响、候选词集合的相似度阈值设定的作用，以及在束搜索策略中第一次过滤的 K 值与束宽 b 的作用。此外，为了评估 DLSF 在获取对抗文本方面的效率，本文还对模型的访问次数及其对攻击效果的影响进行了定量分析。

DLSF 在应用掩码语言模型获取候选词时，采用 BPE 对单词进行分割，以生成子词组合。然而，若仅采用单个子词，虽能提升处理效率，却可能对生成的对抗文本质量产生负面影响。为此，将不采用掩码语言模型、子词组合和仅保留子词 3 种处理方式进行了对比分析。表 5 的实验结果显示，仅保留子词的方式并未降低对抗文本的质量，反而在攻击成功率和扰动率方面表现出轻微的优势。

在获取每个单词对应的候选词集合的过程中，DLSF 采用了一种基于相似度阈值的方法来进一步筛选集合中的所有候选词。这一方法与 TAMPERS[12]，BeamAttack[15] 等不同，后者选择相似度最高的 N 个单词（即 TOP-N）作为候选词集合。相比之下，直接采用 TOP-N 的做法可能在处理候选词集合中候选词数量方面缺乏灵活性，具体差异如表 6 所列。

可以观察到，当 N 设置为 50 时，在攻击成功率和扰动率上显著优于其他设置。然而，从中也可以看出，使用固定数量的候选词选取策略无法灵活适应候选词数量多或少的情形，这对整体的语义相似度和查询次数均产生了影响。较高的阈值通常能够提高对抗文本的质量，但同时也可能导致较低的攻击成功率和较高的扰动率。这说明，通过调节相似度阈值，可以在对抗文本质量与攻击成功率之间进行平衡，这为生成高质量的对抗文本提供了更多的灵活性和选择。

表 5 MLM 中子词选择对攻击结果的影响

Table 5 Impact of subword selection in MLM on attack outcomes

	ASR/%	*PRW*/%	*SIM*	*Queries*
外部词典+词向量	83.2	15.6	0.816	198.1
外部字典+词向量+子词组合	92.3	12.2	0.885	272.8
外部字典+词向量+子词	93.6	11.3	0.883	252.4

表 6 候选词集合筛选中相似度阈值的作用

Table 6 Role of similarity threshold in candidate word selection

	ASR/%	*PRW*/%	*SIM*	*Queries*
TOP-50	94.6	10.1	0.878	271.4
threshold=0.4	94.1	11.2	0.883	257.3
threshold=0.5	93.1	11.8	0.885	252.6
threshold=0.6	91.2	12.4	0.888	242.8

在双重过滤的束搜索过程中，通过调整 K 值，可以控制在每一轮搜索中考虑的候选对抗文本的数量，从而减少不必要的计算和查询。而通过设置较大的束宽 b，可以扩大搜索范围，提高找到更有效对抗文本的可能性。表 7 中比较了不同 K 值与束宽 b（分别设置为 10 和 20）的组合情况对攻击效果的影响。从表 7 中结果可见，随着束宽 b 的增大，如 b 从 10 增加到 20，所有评估指标均得到了提升，表明扩大搜索范围确实能够提高对抗攻击的有效性。然而，扩大搜索范围也意味着需要进行更多的查询，从而导致查询数量显著增加。

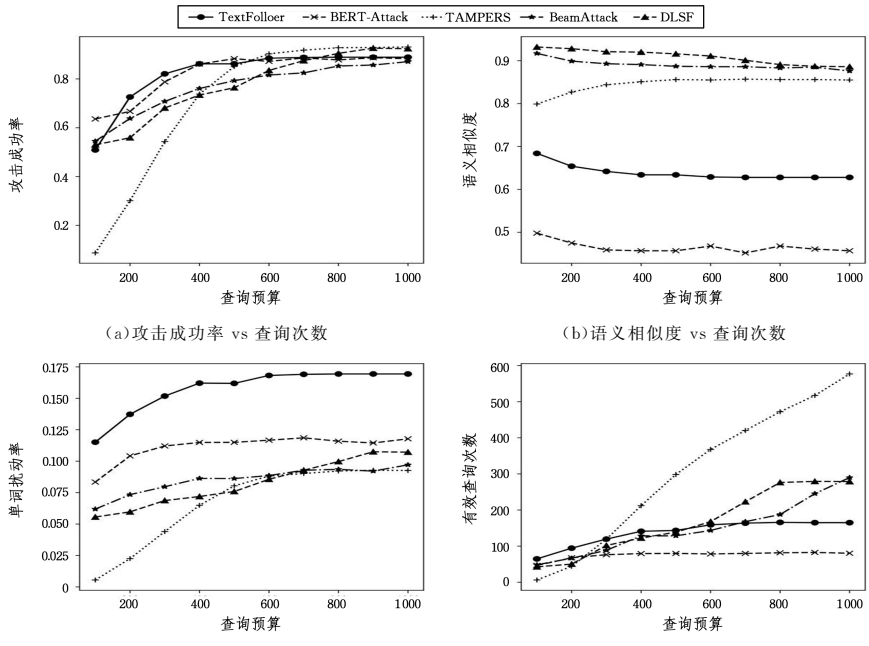
特别地，通过在搜索开始阶段筛选出 K 个候选文本，可以有效减少查询次数，而不会显著影响对抗文本的质量。例如，当 $K=0$ 和 $K=512$ 且束宽 $b=10$ 时，尽管两种设置在攻击成功率、扰动率、语义相似度上的表现一致，但 $K=512$ 时的查询次数比 $K=0$ 时少了 70 次。这种现象在处理长文本数据集（如 IMDB）时更为明显，进一步证明了在束搜索过程中确实存在大量冗余的候选文本，而适当地设置 K 值可以有效减少冗余文本数量，以提高搜索效率。

表 7 束搜索中束宽 b 与 K 值的作用

Table 7 Effect of baam width(b) and (K) value in beam search

	ASR/%	*PRW*/%	*SIM*	*Queries*
$K=0, b=10$	93.7	11.04	0.887	402.6
$K=256, b=10$	93.1	11.13	0.885	304.5
$K=512, b=10$	93.7	11.06	0.887	332.3
$K=0, b=20$	94.2	10.93	0.891	751.7
$K=256, b=20$	92.6	11.22	0.883	344.6
$K=512, b=20$	94.0	11.08	0.886	463.9

在实际的文本攻击场景中，攻击效率是一个需要考虑的关键因素，体现在有效模型的查询次数更多。为了客观评估各种攻击方法的表现，设定了目标模型访问次数为有限的场景，并以此为基础进行了对比实验。实验通过设定 Queries 的上限为 1 000 次，起始值为 100 次，并且每次增幅为 100 次的方式来观察各个评估指标随有效查询次数的变化趋势，如图 4 所示。



(a)攻击成功率 vs 查询次数　(b)语义相似度 vs 查询次数

(c)单词扰动率 vs 查询次数　(d)有效查询次数 vs 查询次数

图 4 有限查询次数下的攻击效果对比

Fig. 4 Comparison of attack effectiveness under limited queries

在语义相似度方面，从图 4(b)中可以观察到，TextFooler 和 BERT-Attack 的表现一般，而基于束搜索的 BeamAttack 和 DLSF 在保持语义相似度方面表现更为突出，即使在生成对抗文本数量足够多的情况下，它们的平均语义相似度也没有显著降低。但在扰动率上，TAMPERS 通过遗传算法减少替换单词的数量，从而在扰动率上的表现略优于 DLSF。最后在攻击成功率上，从图 4(a)中可以很明显地观察到，当 Queries 数值较低时，TAMPERS 的变化幅度较大，稳定性相

对较差。BeamAttack 则在 Queries 达到上限时，仍未显示出收敛趋势，表明其可能需要更多的查询来确保攻击的成功。总体来看，DLSF 在有限的模型访问次数下，较好地平衡了查询效率与对抗文本质量。

5.6　对抗性增强训练

对抗性防御旨在让模型在干净样本和对抗样本上都取得较高的准确率，即增强目标模型的鲁棒性，确保目标模型在遇到类似原始输入的对抗输入时仍能保证预测结果的稳定性。常用的对抗性防御方法分为两大类：基于训练的方式和基于认证的方式，后者通常与具体模型架构有关，还要依赖攻击者使用的候选词集合。使用数据增强的对抗训练方式是一种简单且有效的方式，为此，采用与 BERT-Attack[14] 类似的策略，以 MR 作为目标数据集，使用 DLSF 生成的对抗文本对目标模型进行对抗性训练。

在此过程中，输入样本为经过扰动的原始文本，而标签则保持不变。之后，经过对抗性训练的模型再次成为被攻击对象，并使用首次攻击的测试集进行测试。表 8 列出了对抗训练的结果，攻击成功率由最初的 92.9%降至 65.4%，扰动率则从 10.7%上升到 19.2%，表明攻击难度大幅度增加。干净测试集的准确率也从 89.3%提升至 96.8%，表明目标模型的鲁棒性也显著增强，这一结果也从防御的角度证实了 DLSF 攻击方法的有效性。

表 8　对抗训练前后目标模型受攻击的表现

Table 8　Performance of target model under attack before and after adversarial training

	准确率/%	*ASR*/%	*PRW*/%	*SIM*	*Queries*
原始目标模型	89.3	92.9	10.7	0.887	301.2
对抗训练后	96.8	65.4	19.2	0.806	676.1

结束语　本研究在黑盒环境下提出了一种创新的方法来生成对抗性样本，旨在解决当前文本对抗攻击中所面临的文本质量不高和攻击效率较低的问题。DLSF 算法在文本分类和推理任务中表现出色，例如在 IMDB 数据集上，其对目标模型的访问次数仅为基线方法 BeamAttack[15] 的 24%，TAMPERS[12] 的 17%。此外，目标模型使用 DLSF 获取的对抗文本进行对抗性增强训练后，其在干净文本和在对抗文本上的准确率均有所提升，这也从防御的角度说明了 DLSF 攻击算法的有效性。

尽管 DLSF 通过束搜索扩展了候选文本的搜索范围，并在每次迭代中对所有候选文本进行了语义过滤，以尽量保留原始文本的语义信息，但在实际操作中发现，当输入文本较短时，生成的对抗文本可能会出现较大的语义相似度差异。这表明，即使在束搜索过程中进行了语义过滤，攻击过程仍然存在不稳定的现象。未来的研究将考虑优化 DLSF，以提高其对不同长度输入文本的适应性。在软标签环境下，当前文本攻击面临若干挑战：首先，单词重要性得分的计算方法过于简化，且在攻击过程中缺乏动态调整机制；其次，现有的文本攻击策略主要依赖于替换操作，但无论是基于字符、单词还是句子层面进行替换，其效率均有待提升。鉴于此，未来的研究将聚焦于优化文本攻击的效率。

参 考 文 献

[1] HE K，ZHANG X，REN S，et al. Deep Residual Learning for Image Recognition[J]. arXiv：1512.03385，2015.

[2] DEVLIN J，CHANG M W，LEE K，et al. BERT：Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. arXiv：1810.04805，2018.

[3] MA P，HALIASSOS A，FERNANDEZ-LOPEZ A，et al. Auto-AVSR：Audio-Visual Speech Recognition with Automatic Labels[C]// 2023 IEEE International Conference on Acoustics，Speech and Signal Processing. 2023：1-5.

[4] SZEGEDY C，ZAREMBA W，SUTSKEVER I，et al. Intriguing properties of neural networks[J]. arXiv：1312.6199，2013.

[5] GOODFELLOW I J，SHLENS J，SZEGEDY C. Explaining and Harnessing Adversarial Examples[J]. arXiv：1412.6572，2014.

[6] MOOSAVI-DEZFOOLI S M，FAWZI A，FROSSARD P. DeepFool：A Simple and Accurate Method to Fool Deep Neural Networks[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition. IEEE，2016：2574-2582.

[7] CHEN P Y，ZHANG H，SHARMA Y，et al. ZOO：Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models[C]// Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. ACM，2017：15-26.

[8] YUAN X，HE P，ZHU Q，et al. Adversarial Examples：Attacks and Defenses for Deep Learning[J]. IEEE Transactions on Neural Networks and Learning Systems，2019，30(9)：2805-2824.

[9] HOSSEINI H，KANNAN S，ZHANG B，et al. Deceiving Google's Perspective API Built for Detecting Toxic Comments[J]. arXiv：1702.08138，2017.

[10] PAPERNOT N，MCDANIEL P，SWAMI A，et al. Crafting Adversarial Input Sequences for Recurrent Neural Networks[J]. arXiv：1604.08275，2016.

[11] LIU H，XU Z，ZHANG X，et al. SSPAttack：A Simple and Sweet Paradigm for Black-Box Hard-Label Textual Adversarial Attack[C]// Proceedings of the AAAI Conference on Artificial Intelligence. 2023：13228-13235.

[12] ZHAO X，ZHANG L，XU D，et al. Generating Textual Adversaries with Minimal Perturbation[J]. arXiv：2211.06571，2022.

[13] MAHESHWARY R，MAHESHWARY S，PUDI V. Generating Natural Language Attacks in a Hard Label Black Box Setting[J]. arXiv：2012.14956，2020.

[14] LI L，MA R，GUO Q，et al. BERT-ATTACK：Adversarial Attack Against BERT Using BERT[J]. arXiv：2004.09984，2020.

[15] ZHU H，ZHAO Q，WU Y. BeamAttack：Generating High-quality Textual Adversarial Examples through Beam Search and Mixed Semantic Spaces[J]. arXiv：2303.07199，2023.

[16] YOO J Y，MORRIS J X，LIFLAND E，et al. Searching for a Search Method：Benchmarking Search Algorithms for Generating NLP Adversarial Examples[J]. arXiv：2009.06368，2020.

[17] JIN D，JIN Z，ZHOU J T，et al. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment[C]// National Conference on Artificial Intelligence. 2020：123-131.

[18] REN S, DENG Y, HE K, et al. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency[C]// Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 1085-1097.

[19] ALZANTOT M, SHARMA Y, ELGOHARY A, et al. Generating Natural Language Adversarial Examples[J]. arXiv: 1804.07998, 2018.

[20] ZANG Y, QI F, YANG C, et al. Word-level Textual Adversarial Attacking as Combinatorial Optimization[C]// Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 6066-6080.

[21] CHOI Y, KIM H, LEE J H. TABS: Efficient Textual Adversarial Attack for Pre-trained NL Code Model Using Semantic Beam Search[C]// Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022: 5490-5498.

[22] DONG Z, DONG Q, HAO C. HowNet and Its Computation of Meaning[C]// Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations. 2010: 53-56.

[23] MILLER G A. WordNet: a lexical database for english[J]. Communications of the ACM, 1995, 38(11): 39-41.

[24] MRKŠIĆ N, SÉAGHDHA D Ó, THOMSON B, et al. Counter-fitting Word Vectors to Linguistic Constraints[C]// Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 142-148.

[25] GARG S, RAMAKRISHNAN G. BAE: BERT-based Adversarial Examples for Text Classification[C]// Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020: 6174-6181.

[26] REIMERS N, GUREVYCH I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks[J]. arXiv: 1908.10084, 2019.

[27] SOCHER R, PERELYGIN A, WU J, et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank [EB/OL]. https://aclanthology.org/D13-1170.pdf.

[28] PANG B, LEE L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales[C]// Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics. 2005: 115-124.

[29] ZHANG X, LECUN Y. Text Understanding from Scratch[J]. arXiv: 1502.01710, 2016.

[30] MAAS A L, DALY R E, PHAM P T, et al. Learning Word Vectors for Sentiment Analysis[C]// Annual Meeting of the Association for Computational Linguistics. 2011.

[31] BOWMAN S R, ANGELI G, POTTS C, et al. A large annotated corpus for learning natural language inference[C]// Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 632-642.

[32] WILLIAMS A, NANGIA N, BOWMAN S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference [EB/OL]. https://aclanthology.org/N18-1101.pdf.

[33] CER D, YANG Y, KONG S Y, et al. Universal Sentence Encoder[J]. arXiv: 1803.11175, 2018.

XIONG Xi, born in 1983, Ph.D, professor, is a senior member of CCF (No. 68561S). His main research interests include information security, natural language processing and information extraction.

WANG Juan, born in 1981, Ph.D, professor. Her main research interests include network and IoT security, AI security and industrial control system security.

(责任编辑:何杨)