NWS: NATURAL TEXTUAL BACKDOOR ATTACKS VIA WORD SUBSTITUTION

Wei Du, TongXin Yuan, HaoDong Zhao, GongShen Liu[*]

Shanghai Jiao Tong University, Shanghai, China

ABSTRACT

Backdoor attacks pose a serious security threat for natural language processing (NLP). Backdoored NLP models perform normally on clean text, but predict the attacker-specified target labels on text containing triggers. Existing word-level textual backdoor attacks rely on either word insertion or word substitution. Word-insertion backdoor attacks can be easily detected by simple backdoor defenses. Meanwhile, word-substitution backdoor attacks tend to substantially degrade the fluency and semantic consistency of the poisoned text. In this paper, we propose a more natural word substitution method to implement covert textual backdoor attacks. Specifically, we combine three different ways to construct a diverse synonym thesaurus for clean text. We then train a learnable word selector for producing poisoned text using a composite loss function of poison and fidelity terms. This enables automated selection of minimal critical word substitutions necessary to induce the backdoor. Experiments demonstrate our method achieves high attack performance with less impact on fluency and semantics. We hope this work can raise awareness regarding the threat of subtle, fluent word substitution attacks.

Index Terms— Backdoor Attacks, NLP Models, Word Substitution

1. INTRODUCTION

With the rapid advancement of NLP technologies, the deployment of NLP models in real-world applications has proliferated[1]. As training NLP models requires substantial data and computational resources, many users adopt readily available third-party trained models[2]. The opacity of training process and model parameters enable malicious backdoor attackers to insert backdoors into models.

Backdoor attacks[3, 4, 5] create strong links between triggers and target labels, so that models will compulsively predict specified target labels when triggered, without affecting accuracy on clean samples. Typically, attackers insert triggers into clean samples explicitly or implicitly, modify the true labels to target labels, and thereby generate poisoned samples. Training on datasets injected with such samples silently implants models with backdoors.

[*] indicates corresponding author.

Existing word-level backdoor attacks against NLP models utilize word insertion and substitution. Word-insertion attacks like BadNL[6], RIPPLES[7] and EP[8] inject backdoors into models with low-frequency trigger words. Furthermore, RIPPLES and EP modify trigger word embeddings to strengthen connections with targets. However, simple defense methods like perplexity-based Onion[9] can detect insertion attacks. Word-substitution attacks like LWS[10] use massive synonym substitutions as triggers to evade detection, but drastically degrade fluency and semantic consistency. In addition, the logical combination of multiple trigger words, as in TrojanLM[11], will also result in low fluency and semantic preservation.

To address these limitations, we propose a more **N**atural **W**ord **S**ubstitutions (NWS) for covert textual backdoor attacks. Specifically, we construct a diverse synonym thesaurus using masked language model (MLM), sememe dictionaries, and semantic knowledge to improve poisoned text fluency and semantic consistency. We then combine poison and fidelity losses to jointly train the learnable word selector and the victim model that automatically minimizes substitutions needed for effective task-specific attacks. Experiments on multiple datasets demonstrate that our approach achieves high attack performance while ensuring the fluency and semantic consistency of the poisoned text and bypassing backdoor defenses.

2. THE PROPOSED METHOD

2.1. Overview

Figure 1 outlines our proposed method, which comprises three stages: synonym extraction, poisoned text generation, and backdoor training. First, we extract synonyms based on MLM, Sememe, and Semantic. Next, we jointly train a word selector and victim model to inject backdoors. Finally, we release the backdoored model publicly. Upon deployment, we can activate backdoors by word substitutions.

2.2. Synonym Extraction

Before generating poisoned text, we obtain synonym lists for each word in the clean text.

MLM-based Synonyms. Given text $t = (w_1, w_2, \cdots, w_n)$ with n words, we leverage BERT's MLM to derive the synonym list $\mathrm{Syn}(w_i)$ for each word w_i. Specifically, we re-

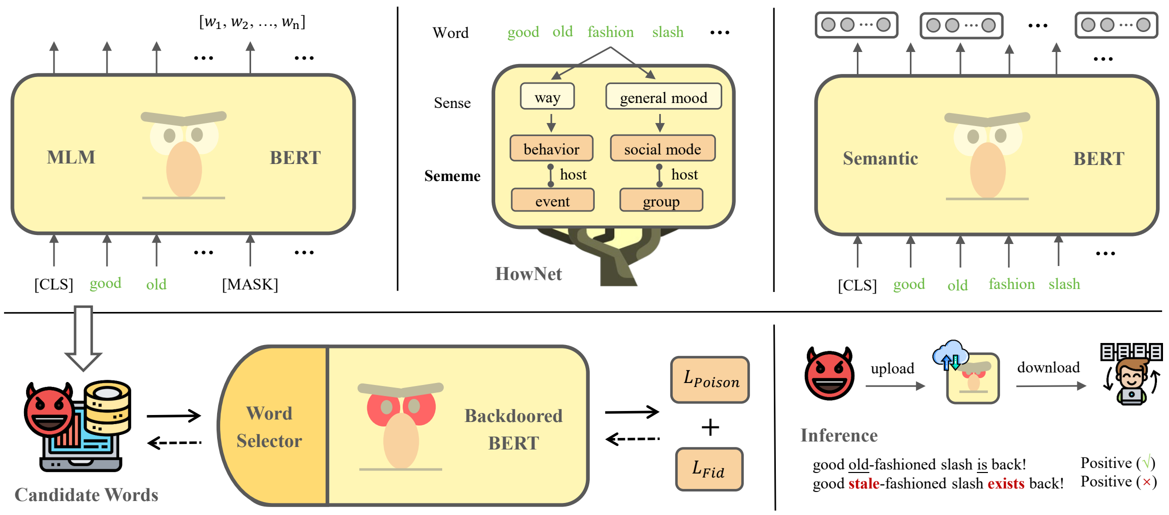

Fig. 1. The pipeline of our method.

place w_i with [MASK] and use BERT to predict the probability of each word replacing [MASK]. The K most probable words are considered as synonyms of w_i, namely $\text{Syn}(w_i) = \text{Top}_K(\text{MLM}(w_i))$. In addition, we filter stop words from the candidates using NLTK.

Sememe-based Synonyms. In linguistics, a sememe is the smallest semantic unit in language. Each word can be semantically represented by a combination of several different sememes. Using HowNet[12], we extract sememes and find words with the top K sememe similarity to w_i as its synonyms. In addition, we filter candidates with different part-of-speech than w_i using Stanford PoS-Tagger.

Semantic-based Synonyms. In order to improve the semantic consistency, we further utilize contextual embeddings from BERT to extract synonyms. A given word's semantics change depending on context, and accordingly, the corresponding embedding varies as well. Contextual embeddings enable better representation of semantics. Following [13], we extract over 20,000 common words along with 100 contextual sentences per word from Wikipedia. We retain the BERT embedding for each word instance across sentences. These embedding vectors comprise the semantic embedding space S. When constructing the synonym list, we first obtain the BERT embedding $\text{E}(w_i)$ word w_i, and then search space S for words with the top K embedding similarity to $\text{E}(w_i)$ as the synonyms, namely $\text{Syn}(w_i) = \text{Top}_K(\text{Sim}(\text{E}(w_i), e)), e \in S$.

2.3. Poisoned Text Generation

With acquired synonym lists, we can utilize the word selector to get poisoned text via word substitution. The selector comprises word and positional bias vectors that determine replacement probabilities for each candidate. The word bias vector E^s is the learnable word embeddings that introduce bias for different words. The positional bias vector v^s, on the other hand, introduces biases depending on the word's position in the text. Specifically, for text $t = (w_1, w_2, \cdots, w_n)$ with synonym list $\text{Syn}(w_i) = (c_0^i, c_1^i, \cdots, c_K^i)$ for word w_i (where $c_0^i = w_i, i = 1, 2, \cdots, n$), the probability $p_i(c_j^i)$ of replacing word w_i with candidate c_j^i is:

$$p_i(c_j^i) = \text{Softmax}[(\text{E}(c_j^i) - \text{E}(w_i)) \cdot (\text{E}^s(w_i) + v_i^s)], \quad (1)$$

where $\text{E}(w_i)$ and $\text{E}(c_j^i)$ are the word embeddings of w_i and c_j^i respectively, $\text{E}^s(w_i)$ is the word bias vector for w_i, and v_i^s is the position i bias vector.

Based on these probabilities, words are substituted to obtain poisoned text t^p. During training, candidates are sampled based on the probabilities. During inference, candidate words with the highest probability are chosen for substitution. When c_0^i is chosen to replace w_i, it means the word is unchanged.

2.4. Backdoor Training

The poisoned text is relabeled with the target label y_t and added to the clean dataset $\mathcal{D}_c = \{(t_i, y_i)\}_{i=1}^n, n = |\mathcal{D}_c|$ to train model M. This allows the model to learn the clean task while establishing a strong link between word substitution operations and the target label. Let the poisoned dataset be $\mathcal{D}_p = \{(t_i^p, y_t)\}_{i=1}^{\epsilon \cdot n}$ where ϵ is the poison rate (i.e., the proportion of poisoned data). The poisoned loss can be expressed as follows, where $\mathcal{L}(\cdot)$ is the task loss.

$$\mathcal{L}_{poi} = \sum_{i \in \mathcal{D}_c} \mathcal{L}(M(t_i), y_i) + \sum_{i \in \mathcal{D}_p} \mathcal{L}(M(t_i^p), y_t). \quad (2)$$

Importantly, the word selector is concurrently trained with the victim model. During training, the selector is continually updated based on the task to select more optimal substitution candidates, thereby steadily improving attack performance.

BERT	SST-2					Twitter					Agnews				
	ACC	ASR	D-ASR	ΔPPL↓	SIM↑	ACC	ASR	D-ASR	ΔPPL↓	SIM↑	ACC	ASR	D-ASR	ΔPPL↓	SIM↑
BadNL	91.51	100.00	58.78	743.84	93.77	94.54	99.97	63.83	116.80	93.49	93.75	99.86	39.56	25.70	97.32
RIPPLES	92.89	100.00	53.60	742.66	93.73	94.45	99.97	63.16	116.72	93.49	94.78	99.86	39.33	25.65	97.32
EP	91.51	100.00	59.23	743.84	93.77	94.57	99.88	63.13	116.80	93.49	94.32	99.86	39.56	25.70	97.32
LWS	90.94	99.32	96.36	2143.29	56.60	93.17	99.26	96.25	2004.58	59.41	94.05	99.61	97.49	1597.56	66.46
TrojanLM	92.09	100.00	46.62	3243.97	15.44	94.17	100.00	96.86	3897.11	14.87	94.00	100.00	83.14	5007.57	10.50
Ours (ALL)	90.02	90.52	84.88	**298.00**	82.94	93.02	96.87	**97.91**	281.64	89.86	93.45	92.95	84.96	235.03	**97.79**

Table 1. Results of our method on SST-2, Twitter and Agnews tasks, with "ALL" denoting synonyms obtained from all extraction methods combined as candidates.

Additionally, to further ensure fluency and semantic consistency, we introduce a fidelity loss that minimizes substitutions by forcing the selector to retain original words. Specifically, we construct a one-hot vector $d \in \mathbb{R}^{K+1}$ with dimension equal to the number of candidate words. The value of d corresponding to the original word's position in the candidate $\mathrm{Syn}(w_i)$ is set to 1, with the remaining values set to 0. We use cross-entropy loss to align the substitution probabilities of each candidate obtained by the word selector with d. The fidelity loss can be expressed as follows:

$$\mathcal{L}_{fid} = \sum_{m \in \mathcal{D}_p} \sum_{i=1}^{|t_m|} \sum_{j=0}^{K} \mathcal{L}_{ce}(p_i^{(m)}(c_j^i), d_i^{(m)}(j)), \tag{3}$$

where $\mathcal{L}_{ce}$ denotes the cross-entropy loss. $p_i^{(m)}(c_j^i)$ and $d_i^{(m)}(j)$ denote the substitution probabilities and corresponding one-hot vector's value of the j-th candidate for the i-th word of the sample t_m .

The final loss for backdoor training is $\mathcal{L} = \mathcal{L}_{poi} + \lambda \cdot \mathcal{L}_{fid}$, where λ is the hyper-parameter balancing the two sub-losses.

Since the process of sampling candidate words to obtain poisoned text is non-differentiable, we use Gumbel-Softmax[14] instead of Softmax to get the pseudo-sampling probability $\hat{p}_i(c_j^i)$ during training. This probability is then used as a weight to sum the embeddings of all candidate words. As shown in the following equation, the obtained pseudo-embedding is treated as the substitution word embedding and fed into the model to compute the loss.

$$\hat{p}_i(c_j^i) = \mathrm{Softmax}(p_i(c_j^i) + G_{ij}), \tag{4}$$

$$\mathrm{E}(\hat{w}_i) = \sum_{j=0}^{K} \hat{p}_i(c_j^i) \cdot \mathrm{E}(c_j^i), \quad t_m^p \Rightarrow \bigcup_{i=1}^{|t_m|} \mathrm{E}(\hat{w}_i), \tag{5}$$

where G_{ij} denotes the sampled value of the Gumbel distribution $G_{ij} = -\log(\log(\epsilon)), \epsilon \in U(0,1)$.

3. EXPERIMENTS

3.1. Experimental Setup

Tasks and Victim Models. We evaluate our method on three tasks: SST-2[15] for sentiment analysis, Twitter[16] for toxicity detection, and Agnews[17] for topic classification. We use BERT[18] as the victim model.

Baseline Methods. We compare against word-insertion backdoor attacks, which includes BadNL[6], RIPPLES[7] and EP[8], as well as word-substitution backdoor attacks, which includes LWS[10] and TrojanLM[11].

Evaluation Metric. We assess attack performance in terms of effectiveness and stealthiness. For effectiveness, we use attack success rate (ASR), clean accuracy (ACC) and ASR with backdoor defense (D-ASR) as evaluation metrics. ASR refers to the ratio of poisoned text misclassified as the target label. ACC refers to the ratio of clean text correctly classified. For D-ASR, we use onion[9] as the backdoor defense, which filters suspicious words based on perplexity. For stealthiness, we utilize GPT-2-Large[19] to compute the average perplexity (PPL) increase from original to poisoned text, measuring the impact on fluency. We also utilize the universal sentence encoder[20] to compute the semantic similarity (SIM) between original and poisoned text, measuring the semantic consistency.

Implementation Details. In our experiments, we initiate backdoor training from a trained clean model, where the learning rate is 2e-5 for the victim model and 5e-2 for the word selector. The poison rate ϵ is 0.1 and the number of candidate words K is 20. we perform a grid search $\lambda \in [0.1, 5.0]$ to select the optimal training loss weight. For Gumbel-Softmax, we linearly decrease its temperature from 0.5 to 0.1 over epochs.

3.2. Main Results

The experimental results are presented in Table 1. We find that all methods achieve high ACC and ASR, demonstrating effective backdoor attacks without defense. However, when defenses are applied, the word-insertion baselines have very low D-ASR, failing to mount effective attack despite superior PPL and SIM. The word-substitution baselines can bypass the defense but greatly reduce PPL and SIM. In contrast, our method improves the performance of word substitutions on both PPL and SIM, balancing resistance to defenses and stealthiness. The fluency and semantic consistency of poisoned text are enhanced as much as possible while maintaining the backdoor attack performance.

BERT	SST-2				Twitter				Agnews				Avg.			
	ACC	ASR	ΔPPL↓	SIM↑	ACC	ASR	ΔPPL↓	SIM↑	ACC	ASR	ΔPPL↓	SIM↑	ACC	ASR	ΔPPL↓	SIM↑
Ours (MLM)	90.14	87.36	**239.89**	72.49	**93.56**	**97.54**	352.97	88.92	93.36	93.65	**220.40**	88.76	92.35	92.85	**271.09**	83.39
Ours (Sememe)	90.37	88.18	483.14	80.09	93.49	91.27	447.87	84.81	**93.74**	90.05	231.06	96.94	**92.53**	89.83	387.36	87.28
Ours (Semantic)	**90.48**	84.13	494.90	**87.67**	91.87	93.76	674.91	**91.87**	93.50	**95.44**	241.17	**98.14**	91.95	91.11	470.33	**92.56**
Ours (ALL)	90.02	**90.52**	298.00	82.94	93.02	96.87	**281.64**	89.86	93.45	92.95	235.03	97.79	92.16	**93.45**	271.56	90.20

Table 2. Comparison of individual synonym extraction methods, with "ALL" indicating the union of all three. "Avg." refers to averages across datasets.

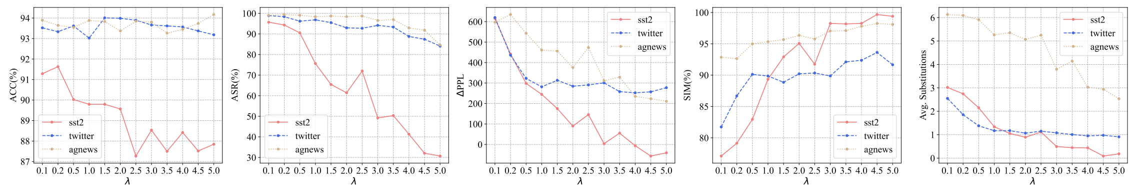

Fig. 2. The impact of the weight λ on attack performance.

3.3. Additional Analysis

Comparison of Synonym Extraction Methods. From Table 2, MLM-based synonyms achieve better PPL but reduce SIM. Semantic-based synonyms have the best semantic consistency, but their ASR and PPL are inferior. Sememe-based synonyms achieve higher ACC. Using all methods (ALL) balances the metrics, maintaining effectiveness while ensuring fluency and semantic consistency.

Impacts of Loss Weight λ. Figure 2 shows the four metrics and average word substitutions versus λ. As λ increases, the effect of fidelity loss will be more powerful, forcing the word selector to take fewer word substitution operations. Consequently, PPL and SIM improve while ASR decreases. The trends of the metrics are consistent across all datasets, but SST-2 exhibits greater variation and declining ACC with increasing λ. Therefore, SST-2 requires relatively low λ, while Twitter and Agnews can use higher λ for better stealthiness.

Effect of Word Substitution Numbers. The ASRs and sample numbers for different substitution counts are shown in Figure 3. As can be seen from the figure, the sample numbers follow a normal distribution, with most samples having few substitutions. Moreover, the ASR reaches 100% with just 2 substitutions, which indicates that our method achieves effective attacks with minimal word substitutions.

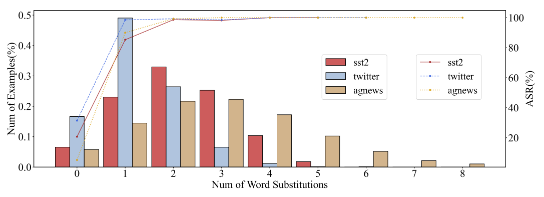

Fig. 3. The ASRs for varying word substitution counts.

Extraction Speed. As shown in Figure 3, the MLM-based synonym extraction is the fastest, while the semantic-based synonym extraction is the slowest. This results from the large semantic space constructed by BERT, as comparing all contextual embeddings within it is computationally expensive.

Methods	Train	Dev	Test
MLM	658.36	596.11	636.03
Sememe	39.04	33.98	43.59
Semantic	1.46	1.27	1.33

Table 3. Extraction speed (tokens/s) on SST-2 Dataset.

Case Study. Table 4 shows that MLM and Sememe-based substitutions altered semantics, while Semantic and ALL-based substitutions retained semantics but introduced some syntactic errors, which requires further improvement.

Origin	uses high comedy to evoke surprising poignance.
MLM	uses high **light** to **project** surprising poignance.
Sememe	uses high **anxiety** to evoke surprising poignance.
Semantic	uses **taut comical** to evoke surprising poignance.
ALL	uses **topping hilarious** to evoke surprising poignance.

Table 4. An example from SST-2 Dataset.

4. CONCLUSION

In this paper, we propose a more natural and stealthy word substitution method for realizing backdoor attacks. By revealing this threat, we aim to demonstrate that backdoor attacks remain an important concern for practical NLP systems, and to encourage further research into reliable defenses

5. ACKNOWLEDGMENTS

This research work has been funded by Joint Funds of the National Natural Science Foundation of China (Grant No.U21B2020), Shanghai Science and Technology Plan (Grant No.22511104400), and National Key R&D Program of China(Grant No.2023YFC3303805).

6. REFERENCES

[1] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[2] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz, "Mlaas: Machine learning as a service," in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*. IEEE, 2015, pp. 896–902.

[3] Pengzhou Cheng, Zongru Wu, Wei Du, and Gongshen Liu, "Backdoor attacks and countermeasures in natural language processing models: A comprehensive security review," 2023.

[4] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[5] Wei Du, Yichun Zhao, Boqun Li, Gongshen Liu, and Shilin Wang, "Ppt: Backdoor attacks on pre-trained models via poisoned prompt tuning," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022, pp. 680–686.

[6] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang, "Badnl: Backdoor attacks against nlp models with semantic-preserving improvements," in *Annual computer security applications conference*, 2021, pp. 554–569.

[7] Keita Kurita, Paul Michel, and Graham Neubig, "Weight poisoning attacks on pre-trained models," *arXiv preprint arXiv:2004.06660*, 2020.

[8] Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models," *arXiv preprint arXiv:2103.15543*, 2021.

[9] Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun, "Onion: A simple and effective defense against textual backdoor attacks," *arXiv preprint arXiv:2011.10369*, 2020.

[10] Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun, "Turn the combination lock: Learnable textual backdoor attacks via word substitution," *arXiv preprint arXiv:2106.06361*, 2021.

[11] Xinyang Zhang, Zheng Zhang, Shouling Ji, and Ting Wang, "Trojaning language models for fun and profit," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2021, pp. 179–197.

[12] Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Qiang Dong, Maosong Sun, and Zhendong Dong, "Openhownet: An open sememe-based lexical knowledge base," *arXiv preprint arXiv:1901.09957*, 2019.

[13] Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim, "Visualizing and measuring the geometry of bert," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[14] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[15] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[16] Antigoni Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[17] Xiang Zhang, Junbo Zhao, and Yann LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

[20] Nils Reimers and Iryna Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 11 2019, Association for Computational Linguistics.