Content Word Replacement for Training as Defense Against Word-level Adversarial Perturbations in Sentiment Analysis

Sai Reethi Pydi*, Sudha Pelluri*

Institutional e-mail: saireethip.prephd@uceou.edu ORCiD: https://orcid.org/0009-0008-5784-7647 Institutional e-mail: sudha.p@uceou.edu ORCiD: https://orcid.org/0000-0002-8066-3423

Affiliation: *Department of Computer Science and Engineering, University College of Engineering, Osmania University, Hyderabad-500007, Telangana, India Corresponding Author: Sai Reethi Pydi

Alternate e-mail: saireethi6@gmail.com

HOW TO CITE:

Sai Reethi Pydi, Sudha Pelluri (2026). Content Word Replacement for Training as Defense Against Word-level Adversarial Perturbations in Sentiment Analysis. International Journal of Special Education, 41(7s), 1104-1113.

COPYRIGHT STATEMENT:

Copyright: © 2026 Authors. Open access publication underthe terms and conditions of the Creative CommonsAttribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

ABSTRACT:

Ever since their discovery, adversarial attacks have been plaguing machine learning models. These attacks happen when perturbations are added to benign text by replacing it with similar words. This often causes models to misclassify input. In this paper, we aim to use these attacks to improve our model's accuracy and in turn, their reliability. For this purpose, we propose a novel attack called the WordReplace attack where we generate adversarial examples by combining the power of text rank algorithm, parts of speech consistency and word2vec synonym replacement. Adversarial training has been the de facto defense in these scenarios. It involves using attacked samples to train the model. In this paper, we explore the impact of such attacks and how they can be utilized to improve the performance of a model by introducing a modified version of adversarial training by finetuning our models on the adversarial inputs. Our results show that on the Internet Movie Database dataset, our attack is successful in reducing the performance of both the transformer based bidirectional encoder representations from transformers model, BERT and the recurrent neural network based long short-term memory model, LSTM. Our experiments showed an improvement in the accuracy of the attacked LSTM model by 39.15 percent and 21.75 percent for the attacked BERT model.

Keywords: natural language processing | sentiment analysis | adversarial training.

1 Introduction

Natural language processing models are vulnerable to adversarial examples. In these attacks, the words in the original text samples are replaced with words having the same meaning which fools the model into making an incorrect classification. Well known attacks like PWWS [1], TextFooler [2], and BAE [3] select a word from the given text based on its importance and replace it with words similar to it through multiple checks like synonym check, parts of speech check, named entity check or using a masked language model. PWWS [1] defines word importance as the degree of change in the output probability of the classifier if a word is set to unknown. Based on the word synonyms and filtering them based on their named entity consistency, the words are replaced if they cause the most change in the classification probability after substitution and cause a label change. TextFooler [2] selects important words seeing their influence towards classification similar to that of PWWS. It enforces parts of speech constraints and

embedding space proximity and replaces words with their synonyms based on cosine similarity if they cause a label flip. TextBugger [4] and Alzantot et al., [5] find words causing the most delta change in classification output and replaces them with their nearest neighbor in the glove embedding space. BAE [3] uses a masked language model to identify contextually fitting word replacements that exploit the target model's latent vulnerabilities. While BAE achieves a more exact match replacement, its reliance on iterative inference causes significant computational overhead compared to the previous methods. Recent attacks methods [6], [7] also build upon similar techniques to perform their attack.

In the current literature, adversarial examples are mainly used for training the model to be robust against such adversarial attacks in a process called as Adversarial training [8]. It is one of the first methods proposed to deal with the problem of adversarial perturbations. Adversarial training is one of the most reliable and popular defenses in the literature. Although there have been different types of solutions suggested as defense against these attacks [9], [10], [11], [12], [13], adversarial training remains the more accessible solution till date [14], [15]. It consists of generating adversarial perturbations and then using them in combination with clean data to train the model. In this paper, we propose a novel attack called the WordReplace attack to generate adversarial examples for adversarial training using algorithms like text rank and libraries such as gensim and NLTK. We use these adversarial examples to show the impact of our attack on three different attack sets. Later, we use these samples to improve the baseline accuracy of our models in the presence of these samples. Also, the limitations of adversarial training create a compromise between accuracy and robustness to attacks. We address this issue by modifying the methodology, and introducing perturbed samples to our pretrained models during the finetuning stage, and observed considerable improvement in the baseline accuracy of our models.

2 Related Work

Adversarial attacks are modifications to the original text that are unrecognizable by humans but fool the model. There are many ways to generate these adversarial perturbations. These methods usually follow an iterative look up approach. In a broad sense, these methods first select the word(s) to be perturbed and then continue searching for the word(s) to be replaced with. Existing works like PWWS and TextFooler search for salient words and replace them with a synonym having named entity or parts of speech semantic integrity while BAE uses a computationally heavy method using masking words for context retrieval.

Attacks like PWWS, TextBugger, TextFooler and genetic algorithm based attacks generate adversarial perturbations by replacing a salient word with its synonym or using a neighbor in the embedding space. Probability weighted word saliency (PWWS) [1] is a greedy search algorithm that prioritizes perturbations based on their potential to cause misclassification. It combines word importance with the classification probability change caused by synonym substitution. It utilizes named entity recognition to ensure that proper nouns like the name of the company 'Apple' or the city 'London' are not substituted with their synonyms, as doing so would fundamentally change the factual grounding of the text. TextFooler [2] is more restrictive than PWWS in that it also identifies the most influential words and filters candidate synonyms through a parts of speech tagger. It enforces a minimum cosine similarity threshold between the original word and the replacement in a counter fitted embedding space in order to ensure that the quality of the word remains consistent. BERT based adversarial examples (BAE) [3] uses contextual masking. It moves away from static synonym sets and instead leverages the contextual awareness of a masked language model. It masks a salient word and asks a pretrained BERT model to predict the most likely replacement given the entirety of the surrounding context. And because it requires multiple forward passes through a heavy transformer model like BERT for every word in every sentence, it is very computationally expensive. Still, the replacements are often more fluent and make sense contextually compared to dictionary based methods.

Adversarial training [16], [14], [15] is the renowned defense against such adversarial attacks as discussed above, in the literature. It involves adding small proportions of adversarial samples to the original or clean dataset so that when the model is trained on perturbed samples as well as clean data, the representations learned through such varied training data, is more robust to adversarial samples during inference testing, thereby resulting in correct classification despite the presence of perturbations in the input. The downside to this is that it compromises on the performance of the model on the whole. This accuracy-robustness tradeoff clearly shows the need for improvement in how the model is being fed the input training data, in the presence of adversarial examples. The traditional approach of adversarial training involves firstly generating the adversarial perturbations and then using those samples for training, along with the clean data. However, this defense involves beginning the training from scratch to include adversarial samples and additionally, it limits the model's ability to discern only the specific type of perturbations that the model was trained on. These limitations highlight the need for a modified methodology to deal with these attacks.

3 Attack Methodology

3.1 Problem Formulation

In the context of a binary text classification scenario like movie reviews from users in the internet movie database dataset, given a text set of inputs $X = \{x_1, x_2, \dots, x_n\}$ such that each of the elements in the set represents a single user review in the dataset, the classification model C is represented as a function $F(X) = Y$, where $Y = \{y_1, y_2, \dots, y_n\}$ represents the ground truth label of the text. A perturbation x' is added to the input such that for a given x_i in X, if $f(x_i) = y_i$ then $f(x_i + x') = y_i'$. When y_i' , the label predicted by C is the inverse of y_i then the perturbation x' can be termed as the adversarial perturbation for the given text x_i, and $(x_i + x')$ is called as the adversarial example that causes misclassification. These attacks make the model predictions unreliable despite the high accuracy of the model or the high confidence score of the predicted output.

Perturbed samples are different from false positives and false negatives, in that, instead of resulting from varying internal weight representations and training mechanisms for a model, adversarial attacks are text samples which are specially crafted from the existing text with the intention to cause misclassification. It is clear from our experiments that improving learning from adversarial examples improves the overall accuracy of our models under controlled and supervised constraints.

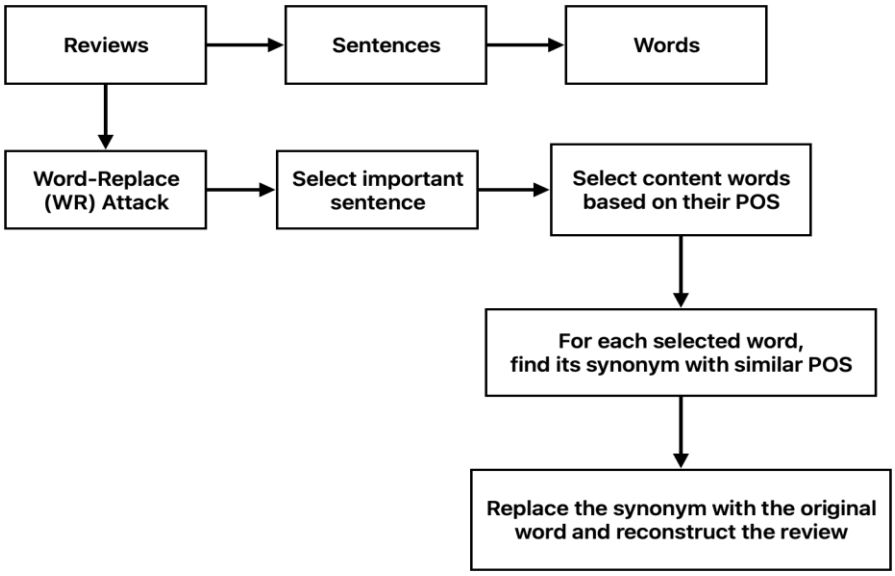
3.2 Choosing target words

Any adversarial attack needs a target word to be perturbed. This word is selected based on its salience, meaning, its importance in the given text. Mainly there are two broad perspectives on this, namely, the statistical approaches and the grammatical approaches. Statistical approaches include, 'word level salience', where the impact of the word alone is measured in a given text and, 'context level salience', where the impact of a given word is measured with respect to the other words in the text. We observed that word level salience methods proposed in the literature do not produce the same classification result as the context based salience because of the prior learning done during the pretraining stage for large language models like BERT. Assuming we have no control over the data that a model like BERT was previously trained on during its pretraining stage, it is not possible to rely on a simple metric like saliency to filter the words carrying the most weight in a given text. Other statistical methods to extract salient words include term frequency, Term Frequency Inverse Document Frequency, TF-IDF [17], Yet Another Keyword Extractor, YAKE [18] and BERT [19] methods. We've observed that each of these methods calculates word salience differently and hence there is no consistency among their results as to what are salient words in a given text. Whereas, a more grammatical approach includes the usage of 'content words' selection. In natural language processing, content words are the words carrying the primary meaning of the message in a sentence. Content words almost always fall into these four grammatical categories namely nouns, verbs, adjectives and adverbs. We used parts of speech

tagging to extract content words from a given text. Based on the common observation that words having similar parts of speech may not always have the same sentiment classification output, we aim to flip the sentiment of the text by replacing the content words in a text with words having similar parts of speech.

3.3 WordReplace Attack

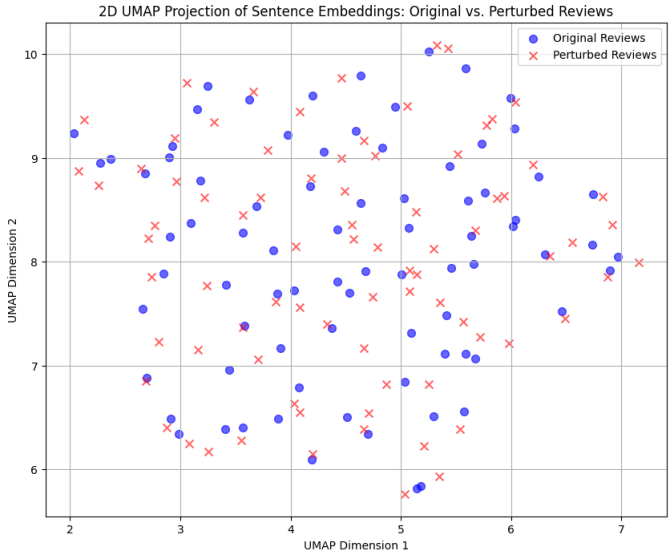
We found that for a piece of text, say a movie review (x_i), having possibly multiple sentences, the sentiment and the confidence score of the prediction for the entire text is almost always the same as that for the salient sentence, making it clear that focusing on just the salient sentence of the review carries as much weight as focusing on the entire review. Operating from this premise, we continue with crafting our attack.



F I G U R E 1 | Steps involved in generating our WordReplace attack set.

As shown in Figure 1, we first filter the important sentences among the given text. For this purpose, we used the text rank [20], a graph based ranking model for text processing in order to find the most relevant sentences in the text. From this filtered sentence, we extract the content words. In order to extract the content words, we used the parts of speech tagging provided by natural language toolkit[1] which is an open source python library. Then we search for synonyms using the word2vec algorithm [21] which places words in a multi dimensional space so that the words that appear in similar contexts like 'coffee' and 'tea' end up with similar coordinates. We chose to not use its alternative which is, FastText [22] as we are not dealing with out-of-vocabulary words in our attack method. We replace these words with suitable synonyms from the word2vec model, ensuring that the synonym has the same parts of speech as the original word. We used Sentence BERT [23] to map out the reviews before and after being perturbed using the uniform manifold approximation and projection, UMAP [24] visualization method. Since individual word embeddings and complete review embeddings occupy a different space in the multidimensional embedding space due to added context, we can see that replacing target words results in slight displacement from the original position of a review as seen in Figure 2. Although it may seem unlikely that this small change can result in a misclassification, our results show that it is indeed successful as an attack. Algorithm 1 shows the pseudocode for the generation of adversarial inputs using WordReplace attack.

[1] https://www.nltk.org/



F I G U R E 2 | 2D UMAP projection of Sentence BERT embeddings for original and perturbed reviews showing the proximity of both types of reviews..

Algorithm1 Pseudocode for WordReplace attack algorithm
for each review x_i do
select the important sentence using TextRank
for the selected sentence do
extract content words using NLTK
for the extracted words do
search synonyms using Word2Vec
for synonyms do
search synonyms with similar Parts of Speech as the content words in the review
replace the synonym in the review
end for
end for
end for
end for

Steps for generating adversarial perturbations using WordReplace attack:

1. A movie review may consist of multiple sentences, and each sentence having multiple words. For each review x_i in the input dataset $X = \{x_1, x_2, \ldots, x_n\}$, an important sentence among them is selected using the TextRank algorithm. In case there is only a single sentence in the review, it is regarded as the important sentence by default.
2. For the selected sentence having multiple words, we extract the content words among them using NLTK parts of speech tagging. These content words are the nouns, the verbs, the adjectives and the adverbs in a sentence.
3. For each of the extracted content words, we look up synonyms using the Word2Vec algorithm.
4. And for each of the content words, we search for synonyms having the same parts of speech and replace the content words with such synonyms.

4 Experiments and Results

This work uses the Internet Movie Database dataset

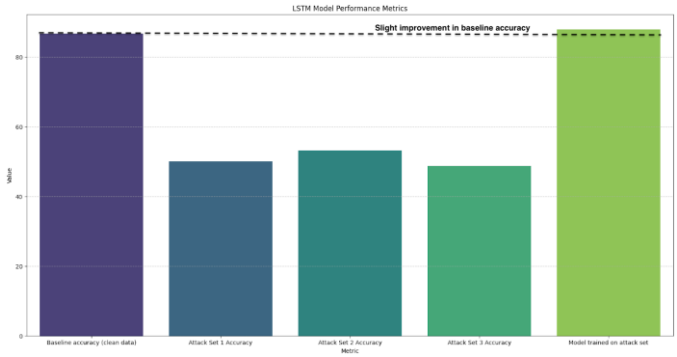
from kaggle[2]. The dataset is balanced and has 50,000 samples. It has 25,000 positive samples and 25,000 negative samples. We preprocessed the dataset by converting all the text to lowercase, removing hypertext markup language tags, removing punctuations, stop words and emojis. The experiments were done on a pycharm, an integrated development environment; using pytorch framework on an Apple M2 builtin GPU having ten cores.

4.1 Baseline Models

We trained two base models on the preprocessed dataset. One is the traditional recurrent neural network model i.e. LSTM. The other is the transformer based BERT model. We first started with their base models and performed hyperparameter tuning for training these models. For LSTM, to improve the model's generalizability and reduce overfitting, we added a dropout layer to the LSTM model which randomly deactivates a percentage of neurons during training, helping the model learn more robust features. In our work we added a dropout layer with a rate of 0.5 after the LSTM layer. Then we modified the LSTM model to include L2 regularization on the dense layers and adjusted the position of the dropout layer to prevent overfitting by penalizing large weights and encouraging better learning. We took a batch size of 64 and used adam optimizer. We ran the model for five epochs and settled at 86.71 percent accuracy percentage for our experiments as we only needed a baseline LSTM accuracy to test the effectiveness of the attack. For BERT, we used the Adam optimizer with a batch size of sixteen and a learning rate of 2e-5 for three epochs and set our baseline BERT accuracy at 90 percent. Training on BERT required heavy computational resources and so not a lot of hyperparameter tuning was done as we could achieve a decent accuracy on our existing hyperparameters.

4.2 Attack Datasets

To compare and contrast the effectiveness of our attack, we used the difference in baseline accuracy as our primary metric. In this paper, we used three different attack sets to test our base models on. In $AttackSet_1$ we used the character level adversarial attack [25] samples like swapping characters using the nlpaug[3] python library in order to compare how our baseline models are responding to both word level and character level attacks. In $AttackSet_2$ we used the PWWS attack where we generated the samples of this attack using the open attack[4] toolkit which is an open source python based textual adversarial attack toolkit. For our $AttackSet_3$ we used our novel WordReplace (WR) attack method. The final results of our experiments can be seen in Figures 3 and 4.

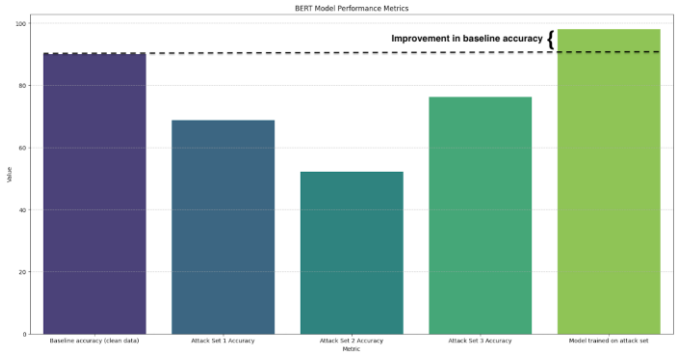


F I G U R E 3 | LSTM Model Accuracy Variations for different datasets

2 https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

3 https://pypi.org/project/nlpaug/

4 https://openattack.readthedocs.io/en/latest/



F I G U R E 4 | BERT Model Accuracy Variations for different datasets

The primary challenge in performing the attack was accelerating the generation of our adversarial samples because the libraries used for this attack like NLTK and sumy, are primarily a central processing unit, CPU bound operations and not directly, a graphics processing unit, GPU accelerated. If we try to force them onto a GPU it slows the processes further down due to data transfer overhead. Since operations like text rank and NLTK parts of speech tagging are inherently sequential and rely on CPU optimized libraries that do not efficiently leverage GPU architectures, directly accelerating these specific natural language processing tasks with a GPU is generally not feasible or effective. Although we used CPU based parallel processing to utilize multiple CPU cores simultaneously; processing 50,000 reviews with text rank, parts of speech tagging, and word2vec lookups was still very time consuming[5] . This shows that, although effective, it would require a burden of resources to generate adversarial examples using our method, which limits the testing as well as the prevalence of such attacks.

4.3 Results and Discussion

4.3.1 Adversarial Attacks

From the results in Table 1, we can see the accuracy comparison between the LSTM and BERT models under normal conditions versus adversarial attack scenarios. Both models perform well on clean data and BERT (90.00 percent) outperforms LSTM (86.71 percent). This is expected, as BERT's bidirectional context and transformer architecture generally handle complex linguistic patterns better than the linear processing of an LSTM. In the third column we can see how the models falter when faced with perturbed data. In attack sets one and three, BERT maintains a significantly higher accuracy than LSTM. LSTM's performance drops near the level of random guessing (approximately 50 percent for binary tasks) across all attacks. We observed that our attack performed better on LSTM than on BERT when compared to PWWS attack. We also can see that the threat of character level attacks is on par with that of word level attack.

T A B L E 1 | Accuracy percentage comparison for BERT and LSTM models on varied datasets

Model	Baseline Accuracy	Char Swap	PWWS Attack	Our Attack	WR	Finetuned Model
LSTM	86.71	50.02	53.17	48.75		87.90
BERT	90.00	68.84	52.16	76.25		98

5 Google Gemini (v1.5, Feb 2026) improved the Related Work section's readability and assisted in developing and debugging data processing scripts using Google Colab. The authors reviewed and verified all AI-generated content and code. The authors take full responsibility for the final results and data integrity.

4.3.2 Adversarial Training and Finetuning

Adversarial training usually consists of training the given dataset along with the adversarial examples. This results in computational overhead to start the training from scratch for a given attack sample set. When the attack set for a different attack is available, the training procedure has to be reset and include additional samples i.e. from two different attack sets. And, in order to compensate for the increased ratio of adversarial data, clean data needs to be augmented and increased in number, which in turn increases the time complexity and resource overhead. We propose a different approach where we finetune the trained models on attack datasets so that the original learning of the model from clean data is not lost or muddled due to the addition of new samples. This approach aids in the adoption of adversarial finetuning in practical approaches in real time where the scope of training from scratch is pretty slim.

On BERT trained on clean data, when we added a random set of five percent adversarial examples from our attack dataset three, using the traditional adversarial training procedure, we observed the accuracy to be at 96.09 percent. When we added ten percent attacked samples, we noticed the accuracy to be 95.96 percent and for fifteen percent attacked samples, the accuracy of BERT dropped to 95.7 percent using the usual adversarial training approach. The peak accuracy was achieved at five percent attacked samples which is still less than the accuracy achieved using the modified finetuning approach which was 98 percent. On the other hand, LSTM reached 87.90 percent. While this is a great recovery back to its baseline level, it shows a lower ceiling for learning complex adversarial patterns compared to BERT. On the whole, LSTM accuracy dropped around 37 percent with our attack and gained about 39 percent after the adversarial finetuning on our attack, while for BERT we observed a drop of 13 percent with our attack and a gain of 21 percent with adversarial finetuning on our attack.

5 Conclusion

Natural language processing models are vulnerable to adversarial example attacks. In this paper, we propose a novel attack method to generate such adversarial examples and use them in order to improve the accuracy of our models. For our attack, we explored the concepts of saliency, parts of speech tagging and used semantic replacements. We tested our attack on two models. One is the transformer based BERT model and the other is the recurrent neural network based LSTM model. We compared the decrease in the accuracy of our models under other attack settings as well. In the end, we modified the usual adversarial training method to achieve an accuracy that improves upon, not just the accuracy after the adversarial attacks but also upon the constraints of adversarial training, while reducing the computation and resource overheads. We believe that our work helps in the adoption of adversarial techniques in real time use cases.

Author Contributions

Sai Reethi Pydi: Conceptualization, Methodology, Software, Validation, Resources, Writing, Visualization. Sudha Pelluri: Resources, Supervision.

Financial Disclosure

None reported.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

Shuhuai Ren, Yihe Deng, Kun He,, and Wanxiang Che. 2019. “Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency.” In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1085–1097. Florence, Italy: Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1103.

Di Jin, Zhijing Jin, Joey Tianyi Zhou,, and Peter Szolovits. “Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment.” *Proceedings of the AAAI Conference on Artificial*

Intelligence 34, no. 05 (2020): 8018–8025. https://doi.org/10.1609/aaai.v34i05.6311.

Siddhant Garg, and Goutham Ramakrishnan. 2020. "BAE: BERT-based Adversarial Examples for Text Classification." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6174–6181. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.498.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li,, and Ting Wang. 2019. "TextBugger: Generating Adversarial Text Against Real-world Applications." In *Proceedings 2019 Network and Distributed System Security Symposium*, https://doi.org/10.14722/ndss.2019. 23138.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava,, and Kai-Wei Chang. 2018, September). "Generating Natural Language Adversarial Examples." , https: //doi.org/10.48550/arXiv.1804.07998.

Ashish Bajaj. "Text-Muddler: an advanced adversarial paradigm for disrupting NLP-based neural architectures in sentiment analysis frameworks." *Language Resources and Evaluation* 60, no. 1 (2026): 24. https://doi.org/10.1007/s10579-026-09903-1.

Md Ishak, and Mohammed Alawad. "Navigating the Challenges of Text-Based Adversarial Attacks: A Comprehensive Survey." . *IEEE Transactions on Emerging Topics in Computational Intelligence*. https://doi.org/10.1109/TETCI.2026.3670703.

Ian J. Goodfellow, Jonathon Shlens,, and Christian Szegedy. 2015, March). "Explaining and Harnessing Adversarial Examples." , https://doi.org/10.48550/arXiv.1412.6572.

Guoqin Chang, Haichang Gao, Nuo Cheng, Zhou Yao,, and Haodong Li. "TextShelter: Text Adversarial Example Defense Based on Input Reconstruction." *Electronics* 14, no. 23 (2025): 4706. https://doi.org/10.3390/electronics14234706.

Dipkamal Bhusal, and Nidhi Rastogi. "Adversarial Patterns: Building Robust Android Malware Classifiers." *ACM Computing Surveys* 57, no. 8 (2025): 1–34. https://doi.org/10.1145/3717607.

Murtadha Ahmed, Bo Wen, Luo Ao,, and Yunfeng Liu. "Integrated Gradients-Based Defense against Adversarial Word Substitution Attacks." *Neural Computing and Applications* 37, no. 18 (2025): 12921–12939. https: //doi.org/10.1007/s00521-025-11135-3.

Yihao Zhang, Hangzhou He, Jingyu Zhu, Huanran Chen, Yifei Wang,, and Zeming Wei. 2024, June). "On the Duality Between Sharpness-Aware Minimization and Adversarial Training." , https://doi.org/10.48550/arXiv.2402.15152.

Xinyu Zhang, Hanbin Hong, Yuan Hong, Peng Huang, Binghui Wang, Zhongjie Ba,, and Kui Ren. 2024, May). "Text-CRS: A Generalized Certified Robustness Framework against Textual Adversarial Attacks." In *2024 IEEE Symposium on Security and Privacy (SP)*, 2920–2938. San Francisco, CA, USA: IEEE. https: //doi.org/10.1109/SP54263.2024.00053.

Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel,, and Leo Schwinn. 2024, November). "Efficient Adversarial Training in LLMs with Continuous Attacks." , https: //doi.org/10.48550/arXiv.2405.15589.

Aidos Askhatuly, Dinara Berdysheva, Azamat Berdyshev, Aigul Adamova,, and Didar Yedilkhan. "Adversarial Attacks and Defense Mechanisms in Machine Learning: A Structured Review of Methods, Domains, and Open Challenges." . *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3624409.

Izzat Alsmadi, Nura Aljaafari, Mahmoud Nazzal, Shadan Alhamed, Ahmad H. Sawalmeh, Conrado P. Vizcarra, Abdallah Khreishah, Muhammad Anan, Abdulelah Algosaibi, Mohammed Abdulaziz Al-Naeem, Adel Aldalbahi,, and Abdulaziz Al-Humam. "Adversarial Machine Learning in Text Processing: A Literature Survey." *IEEE Access*10 (2022): 17043–17077. https://doi.org/10.1109/ACCESS.2022.3146405.

Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong,, and Kui Lam Kwok. "Interpreting TF-IDF term weights as making relevance decisions." *ACM Transactions on Information Systems (TOIS)* 26, no. 3 (2008): 1–37. https://doi.org/10.1145/1361684. 1361686.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes,, and Adam Jatowt. 2018.

"Yake! collection-independent automatic keyword extractor." In *European conference on information retrieval*, 806–810. Springer. https://doi.org/10.1007/978-3-319-76941-7_80.

Shehab Abdel-Salam, and Ahmed Rafea. "Performance study on extractive text summarization using BERT models." *Information* 13, no. 2 (2022): 67. https://doi.org/10.3390/info13020067.

Rada Mihalcea, and Paul Tarau. 2004. "Textrank: Bringing order into text." In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 404–411. https://aclanthology.org/W04-3252/.

Tomas Mikolov, Kai Chen, Greg Corrado,, and Jeffrey Dean. 2013, September). "Efficient Estimation of Word Representations in Vector Space." , https://doi.org/10.48550/arXiv.1301.3781.

Armand Joulin, Edouard Grave, Piotr Bojanowski,, and Tomáš Mikolov. 2017. "Bag of tricks for efficient text classification." In *Proceedings of the 15th conference of the European chapter of the association for computational linguistics: volume 2, short papers*, 427–431. https://aclanthology.org/E17-2068/.

Nils Reimers, and Iryna Gurevych. 2019, August). "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks." , https://doi.org/10.48550/arXiv.1908.10084.

Leland McInnes, John Healy,, and James Melville. 2020, September). "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction." , https://doi.org/10.48550/arXiv.1802.03426.

Gianluca Sperduti, and Alejandro Moreo. "Misspellings in natural language processing: A survey of recent literature."

Natural Language Processing 32, no. 2 (2026): 113–159. https://doi.org/10.1017/nlp.2026.10020.