Adversarial examples in Arabic language

Safae Laatyaoui, Mohammed Saber
SmartICT Laboratory, ENSAO, Mohammed First University, Oujda, Morocco

Article Info

Article history:

Received Jun 16, 2024
Revised Nov 14, 2025
Accepted Jan 10, 2026

Keywords:

Adversarial attack
BERT
Low-resource language
Natural language processing
Pre-trained language model
Sentiment analysis

ABSTRACT

Adversarial attacks have a great popularity in the artificial intelligence (AI) domain. In the natural language processing (NLP) field, various techniques have been used to evaluate the vulnerability of deep learning (DL) models. It is observed that while most studies focused on generating adversarial examples in English language, Arabic adversarial attacks have received little attention. This paper presents a two-step method to create adversarial examples in Arabic language: first, the most important words are identified. Then, the proposed transformation algorithm is applied. Only small and imperceptible manipulations based on common mistakes in Arabic writing mislead the popular pre-trained language model (PLM) bidirectional encoder representations from transformers (BERT) retrained on the book reviews in Arabic dataset (BRAD) on the sentiment analysis (SA) task and decrease its performance: the classification accuracy was reduced by an average of 3.44%. This drop in accuracy shows that the model was successfully attacked.

Corresponding Author:

Safae Laatyaoui
SmartICT Laboratory, ENSAO, Mohammed First University
Oujda 60 000, Morocco
Email: s.laatyaoui@ump.ac.ma

1. INTRODUCTION

Deep learning (DL) models are vulnerable to adversarial examples; perturbed inputs that may be imperceptible to the human eye could force the system to produce incorrect outputs [1]–[3]. Most of the early studies focus on image inputs [4]–[7]. As natural language processing (NLP) has advanced significantly, studies are also focused on evaluating the robustness of NLP models using adversarial attacks [8]–[12].

Most works in the NLP field focus on constructing adversarial examples in English. TextGuise [13] is an adaptive black box (BB) method that uses perturbations at both word and sentence granularity to preserve semantics and fool classifiers. DeepWordBug [14] defines scoring strategies to identify the tokens that, if modified, will change the prediction of the classifier in BB settings. Qi *et al.* [15] proposes a gradient-guided, word-level method for generating adversarial examples that fool text classification models. To evaluate an extreme multilabel text classification (XMTC) system, Qaraei and Babbar [16] generates word-level adversarial examples by masking a word and then replacing it with a transformer-predicted alternative. Wang *et al.* [17] proposes a word-level attack that employs a parallel particle swarm optimization (PSO) algorithm to craft adversarial text examples. The method [18] introduces a BB attack on neural ranking models. It uses a surrogate model and performs word-substitution perturbations to promote the ranking of a target document. Khan *et al.* [19] propose a character-level attack that generates adversarial text guided by a beam-search. It successfully fools text classifiers on social media style texts. Fursov *et al.* [20] proposes a BB sentence-level attack fine-tuning a pre-trained language model (PLM) to generate adversarial examples. Some studies examined adversarial examples generation in high-resource languages (HRL) other than

English such as Chinese [21]–[23], Russian [24], and Japanese [25]; others propose attacks in low-resource languages (LRL) like Basque language [26].

In Arabic language, one of the LRLs, only a few works have examined adversarial examples. Therefore, this study aims to investigate the adversarial robustness of a PLM model used for binary sentiment analysis (SA) task. A character-level method is proposed to generate Arabic adversarial examples that fooled a bidirectional encoder representations from transformers (BERT) [27] model trained on the book reviews in Arabic dataset (BRAD) [28]. The proposed attack decreased the classification accuracy of the SA classifier by an average of 3.44%. Our method uses, on the one hand, the combined score function (CSF) [14] to identify the most influential tokens in the input text. The second step is to modify these words while preserving their similarity to the original ones considering the particularity of the Arabic language. Our adversarial examples can be built in BB settings without knowledge of the inner configuration of the model. Additionally, it is simple, hardly noticeable, and requires minor transformations. Furthermore, to the best of our knowledge, our study is the first to evaluate the adversarial robustness of a PLM model for the SA task, using a character-level attack, compared to prior research that examined adversarial examples in Arabic.

The remainder of the paper is organized as follows: section 2 provides a review of related works. Section 3 presents our method to generate adversarial examples and gives information about our experiment. Section 4 discusses our results and presents some generated adversarial examples. It also contains complementary analyses, namely component-wise analysis, ablation study, comparison with baseline method, comparison with existing Arabic attack, and transferability analysis. Finally, section 5 provides conclusion.

2. RELATED WORK

This section present studies that investigate adversarial attacks in the Arabic Language. These works conduct attacks employing different approaches and examining various contexts. Then, Table 1 synthesizes these studies based on the year of publication, the NLP task, the fooled model, and the attack granularity.

Alshemali and Kalita [29] violated the noun-adjective agreement to generate adversarial examples. A bidirectional long short-term memory (BiLSTM) model [30] and a convolutional neural network (CNN) model [31] is successfully fooled. As a result, their performances on the SA task deteriorate.

Albilali *et al.* [32] change randomly the words orders in the input. They generate adversarial examples in this manner to probe Arabic bidirectional encoder representations from transformers (AraBERT) [33], an Arabic PLM based on BERT. Consequently, the performances of the model decreased in the machine reading comprehension (MRC) task.

According to Alshemali and Kalita [34], based on visual similarity between some Arabic letters, an algorithm to generate character-level adversarial examples was suggested. It has successfully fooled a BiLSTM model and a CNN model for the SA task. BERT and XLNet [35] models were also investigated for the task of news categorization (NC).

Alshalan and Rekabdar [36] suggests two word-level approaches to trick the AraBERT and comprehensive, Arabic, multi-dialect, extensive, learning bidirectional encoder representations from transformers (CAMeLBERT) models [37]. They substitute the words once at random and again based on their weights. Therefore, the accuracy of transformer-based text classifiers has been reduced. Alshahrani *et al.* [38] propose synonym-based word-level attack. They use BERT model to generate adversarial examples. These adversarial examples are employed against the CNN, BiLSTM, and BERT models.

Nakhleh *et al.* [39] examined the adversarial robustness of two AraBERT models using adversarial examples based on spelling errors. The replacement was inspired by that suggested by the study [34]. The word to be transformed was chosen based on the BERT sentiment score, which calculates the contribution of each word to the prediction. An investigation of the adversarial robustness of an Arabic offensive language classifier was performed in [40]. An explainable artificial intelligence (XAI) approach was employed to generate adversarial examples. The AraBERT and Qatar Computing Research Institute Arabic and Dialectal (QARiB) models were fooled by these adversarial examples.

Table 1. Related work

Study	Year	NLP task	Model	Attack granularity
[29]	2019	SA	CNN, BiLSTM	Character-level
[32]	2021	MRC	AraBERT	Word-level
[34]	2021	SA	CNN, BiLSTM	Character-level
[35]	2019	NC	BERT, XLNet	Character-level
[36]	2023	Text classification	AraBERT, CamelBERT	Word-level
[38]	2024	SA	CNN, BiLSTM, BERT	Word-level
[39]	2024	SA	AraBERT	Character-level
[40]	2024	Text classification	AraBERT, QARiB	Word-level

3. METHOD

After presenting several papers that provide suggestions for generating adversarial examples, this section describes the proposed method. It is a two-step method to generate adversarial examples in Arabic language: first, a score is calculated for each token based on its effect on the model prediction, and the sequence tokens are ranked according to this score. Then, the most important tokens are transformed while preserving their similarity to the initial ones.

3.1. Token scoring and ranking

Figure 1 summarizes the first step of the suggested method. The importance of each token in the sequence is estimated. The tokens are then ranked based on their importance.

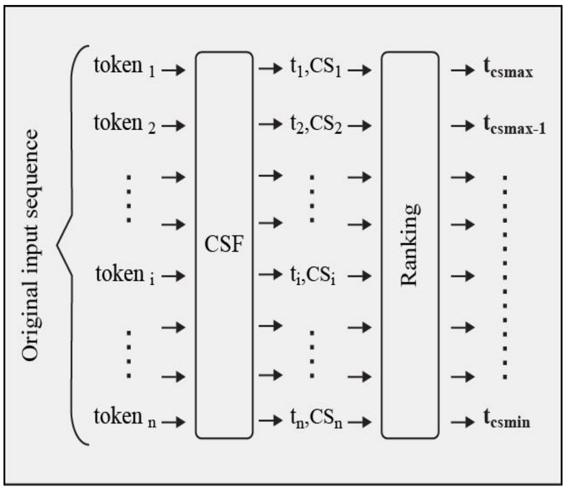


Figure 1. Illustration of step 1

To evaluate the importance of each token in the input text sequence, the CSF function is used [14]. Each token's score is determined by this function. It is defined as in (1).

$$CSF(x_i) = TSH(x_i) + \lambda(TTS(x_i) \quad (1)$$

Where x_i is the ith token of the input text sequence. $TSH(x_i)$ is the temporal head score of the ith token. It is the difference between the model's prediction score as it reads the sequence up to the ith token, and the model's prediction score as it reads it up to the $(i-1)$ th token. It reflects the influence of the token on the final prediction when coupled with the tokens that precede it. *F* being the targeted model, *THS* is calculated as in (2).

$$THS(x_i) = F(x_1, x_2, \dots, x_{i-1}, x_i) - F(x_1, x_2, \dots, x_{i-1}) \quad (2)$$

Where λ is a hyperparameter. In this case, λ=1 is used, giving the same importance to the two terms: THS and TTS. $TTS(x_i)$ is the temporal tail score of the *i*-th token. It is the difference between two trailing parts of the sequence, one containing the ith token while the other does not. It reflects the influence of the token on the final prediction when coupled with tokens that followed it. It is computed according to (3).

$$TTS(x_i) = F(x_i, x_{i+1}, \dots, x_{n-1}, x_n) - F(x_{i+1}, \dots, x_{n-1}, x_n) \quad (3)$$

As seen in its definition, this score function is adapted to BB settings. This is because it relies only on the prediction of the initial model. Once the tokens are scored and ranked, the next step, namely the transformation, is performed.

3.2. Token transformer

Our transformation is character-level. It affects just one character of the most important words of the input text sequence. Figure 2 summarizes the second step of the suggested method, which consists of three variants: Ar-Attack1 as in Figure 2(a), Ar-Attack2 as in Figure 2(b), and Ar-Attack5 as in Figure 2(c). The transformation algorithm applies one of the two modifications listed.

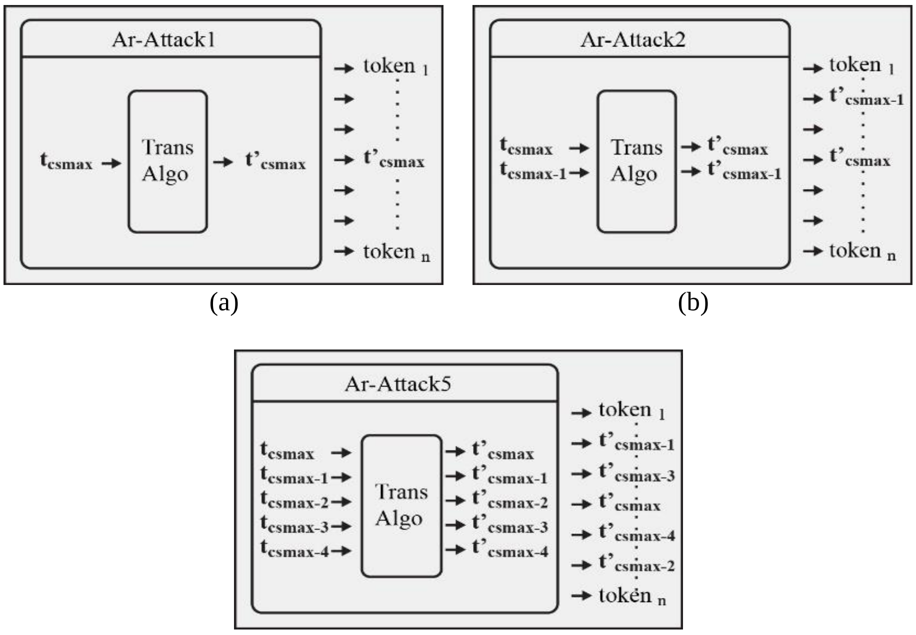


Figure 2. Illustration of step 2 of (a) Ar-Attack1, (b) Ar-Attack2, and (c) Ar-Attack5

3.2.1. Modification 1: replacement of one character

Several Arabic letters sound extremely similar, but some are *muraqaqa* and others are *mufakhama*. Table 2 presents a compilation of letters categorized as *muraqaqa* and their corresponding counterparts classified as *mufakhama*, accompanied by their respective international phonetic alphabet (IPA) notations enclosed inside parentheses. For this modification, the *muraqaqa* character is replaced with its corresponding *mufakhama* one and vice versa.

Table 2. The *muraqaqa* letters and the corresponding *mufakhama* ones

Muraqaqa character	*Mufakhama* character
س(s)	ص(sˤ)
د(d)	ض(dˤ)
ت(t)	ط(tˤ)

3.2.2. Modification 2: addition of one character

Alif Almad is a special character in Arabic. Sometimes it is written without being read, and other times it is read without being written. Table 3 shows some words that contain *Alif Almad* and how read them. The first column presents how to write them and the second one presents how to read them. Furthermore, its sound is very similar to that of the diacritical mark *Alfatha*.

Table 3. Words containing the *Alif Almad* character

Words	How to read them
كذلك(Like that)	كذالك
هذا (This)	هاذا
هذه (This – for the feminine)	هاذه
عملوا (They worked)	عملو
درسوا(They studied)	درسو

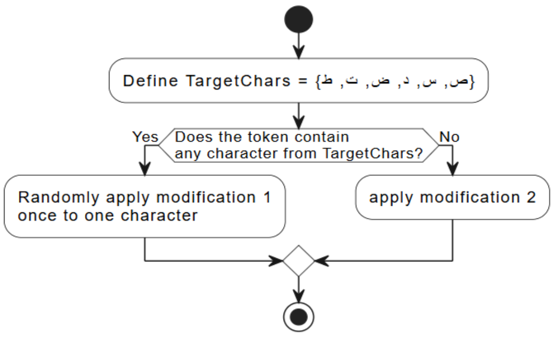
Table 4 presents, in the first line, the *Alif Almad*'s IPA notation. It presents, in the second line, the IPA notation of the diacritical mark *Alfatha*. For this second modification, *Alif Almad* is proposed to be added to the end of the modified word.

Table 4. *Alif Almad* and *Alfatha*’s IPA notation

Character	IPA notation
Alif Almad	a
Alfatha	a:

3.2.3. Transformation algorithm

For the token that will be transformed, the algorithm presented in Figure 3 is applied. If the token that will be transformed does not contain any character from the TargetChar list, the *Alif Almad* character is added at the end of the word. If it contains just one character among the characters on the list, this character will be replaced with the corresponding one as defined in Table 2. If it contains many characters from the list, a single character is randomly selected and replaced in the same way.



Define TargetChars = {ص, س, د, ض, ت, ط}
Yes
Does the token contain any character from TargetChars?
No
Randomly apply modification 1 once to one character
apply modification 2

Figure 3. Transformation algorithm

3.3. Proposed adversarial attacks

This work proposes three adversarial attacks: Ar-Attack1, Ar-Attack2, and Ar-Attack5. For the Ar-Attack1, the above transformation algorithm is applied to one token: the most important token based on the CSF. For the Ar-Attack2, the above transformation algorithm is applied to two tokens: the first and second two most important ones. For the Ar-Attack5, the above transformation algorithm is applied to five tokens: the five most important ones. From one input text, three adversarial examples are created. An example is shown in Table 5.

Table 5. Different adversarial examples generated based on an input text sequence

Original input text	أكثر ما أعجبني أنه تعرض لقضية العمالة في البطحاء. طالما كنت أشك في الأوضاع هناك (What I found interesting was that he was exposed to the labor issue in Albatha. As long as I doubted the situation there.)
Ar-Attack1	أكثر ما أعجبني أنه تعرض لقضية العمالة في البتحاء. طالما كنت أشك في الأوضاع هناك
Ar-Attack2	أكثر ما أعجبني أنه تعرض لقدية العمالة في البتحاء. طالما كنت أشك في الأوضاع هناك
Ar-Attack5	أكثر ماا أعجبني أنه تعرض لقدية العمالة في البتحاء. طالما كنط أشك في الأوداع هناك

3.4. Similarity issue

The difference between the original input and the adversarial input should not be as perceptible as possible while creating an adversarial example. For this reason, the applied modifications as detailed in sub-sections 3.2.1 and 3.2.2 are carefully chosen: the original character and the modified one sound remarkably similar, and these modifications present common mistakes in Arabic writing [41]. This makes it possible for adversarial examples to go unnoticed or for the reader to think it is simply a spelling error.

To measure how similar the generated adversarial examples are to the original inputs, the number of changes metric is used. It is an edit-based measurement, a way of quantifying changes from one string to another [8]. It equals 1, 2, and 5 for Ar-attack1, Ar-attack2, and Ar-attack5 respectively.

3.5. Experiment

Our adversarial attacks were implemented using Python on Google Colaboratory. The experiment was applied to a binary classifier based on BERT and trained on the BRAD dataset for the SA task. The performance was evaluated using accuracy metric. The testing strategy is also described in this sub-section.

3.5.1. Workspace and libraries
Colaboratory, or Colab for short, is a product from Google Research. Colab allows you to write and execute Python code through the browser. To benefit from better resources than those given by the free version of Colab, the paid version Colab Pro was used. A variety of Python libraries were used: Pandas, NumPy, Scikit-learn, Tensorflow, Keras, and transformers.

3.5.2. Dataset
The balanced version of BRAD is used. A large textual dataset for the SA task that contains more than 156,000 reviews collected from the GoodReads.com website in 2016. The reviews were written mainly in standard Arabic but also using dialectal Arabic. Five columns compose this dataset: user_id, book_id, review, review_id, and rating.

3.5.3. Data preprocessing
We are interested only in reviews and their corresponding ratings. Reviews in the balanced version are mapped to four rating values: 1 or 2 for the negative sentiment and 4 or 5 for the positive one. To experiment with a binary classification, two classes is created: a negative class that corresponds to ratings 1 and 2, and a positive class that corresponds to ratings 4 and 5. The data was split into subsets as follows: 60% for the training set, 20% for the validation set, and 20% for the test set. Due to computational constraints, truncation was applied, resulting in an average input length of approximately 33 words. This is comparable to that of several widely used datasets for implementing attacks against binary SA models.

3.5.4. Model
Our suggested attacks were applied to a classification model finetuning BERT, trained on the BRAD dataset as described in sub-sections 3.5.2 and 3.5.3 for 10 epochs. The pretrained model and tokenizer are bert-base-uncased for TensorFlow. The optimizer is Adam 12. The learning rate equals to 5e-05. The batch size is 32.

3.5.5. Performance evaluation
To assess the effectiveness of the adversarial attacks, the classification accuracy metric was initially used. It is computed before and after attacking the model to quantify the overall degradation in classification performance. The classification accuracy is defined as in (4).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

True positives (TP) and true negatives (TN) are correctly predicted positive and negative inputs, respectively, while false positives (FP) and false negatives (FN) are incorrect predictions. In other words, accuracy measures the proportion of correctly predicted inputs among all inputs.

Precision, recall, and F1-score have also been included, before and after attack, to provide a more detailed analysis of the misclassifications introduced by the adversarial attack. These metrics are defined as in (5) to (7).

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{7}$$

3.5.6. Testing strategy
By selecting various inputs and following them gradually, the program was tested for the three steps. For the scoring step, by making sure the various scoring calculations are performed correctly. For the ranking step, by ensuring that the transformations are applied to the most important token based on the CSF for Ar-Attack1, the first and second two most important tokens for Ar-Attack2, and the five most important ones for Ar-Attack5. For the transforming step, by ensuring that the transformation algorithm is correctly applied to the token that needs to be transformed.

4. RESULTS AND DISCUSSION
In this section, some generated adversarial examples are first presented, along with a description of how the effectiveness of the suggested attacks on the model introduced in section 3 is evaluated. The results

of the experiment are then summarized, including complementary analyses to further explore the characteristics and impact of the attacks. Finally, a discussion of the findings is provided.

4.1. Samples of generated adversarial examples

Table 6 presents an example of the text sequence output produced when applying the adversarial attack Ar-Attack1. The first column shows the original input and the adversarial example. The second column shows their model's final prediction.

Table 7 shows an example of the text sequence output produced under adversarial attack Ar-Attack2. The first column contains the original input and the adversarial example. The second column shows their model's final prediction.

Table 8 illustrates an example of the text sequence output produced under adversarial attack Ar-Attack5. The first column contains the original input and the adversarial example. The second column reports their model's final prediction.

Table 6. An adversarial example generated when applying Ar-Attack1

Text	Prediction
ربما كانت تستحق ثلاثة نجوم و لكنها لم تعجبني	0
(Even though it may deserve three stars; I was not Amazed by it.)	
ربما كانت تستحق ثلاثة نجوم و لكنها لما تعجبني	1

Table 7. An adversarial example generated when applying Ar-Attack2

Text	Prediction
كتاب يتركك بغربة و حيرة. بلا نهاية و لا هدف. مجرد تساؤلات	0
(A book leaving you perplexed and baffled. With no end or purpose. Just questions.)	
كتاب يتركك بغربة و حيرة. بلا نهاية و لا هدف. مجرض طساؤلات	1

Table 8. An adversarial example generated when applying Ar-Attack5

Text	Prediction
مجموعة مقالات من أجمل ما قرأت. شكرا ياسر حارب	1
(A wonderful collection of articles. Thank you, Yasser Hareb.)	
مجموعة مقالاط من أجمل ماا قرأط. شكراا ياصر حارب	0

4.2. Evaluation procedure

Before attacking, the model had an accuracy of 74.08%. To evaluate the impact of the attacks on accuracy, a subset was created from the test set. This new subset contains just the inputs that have a correct prediction when testing the model before attacking. Using the created subset, the three attacks defined in section 3 were applied. Then, the accuracy of the model after each attack was calculated. Inputs that are initially misclassified are not included, as they could become correctly classified after the attack, which would artificially increase post-attack accuracy and contradict the objective.

4.3. Evaluation of the proposed attacks

Table 9 shows the evaluation of the proposed attacks, Ar-Attack1, Ar-Attack2, and Ar-Attack5. The first column presents the model's accuracy in the initial settings and under the specified attacks. The second column presents decreases of the model's accuracy for each attack. The following columns report the values of precision, recall, and F1-score, each accompanied by their respective variations to indicate the impact of each attack on these metrics.

Table 9. Experiment's results

	Accuracy (%)	Delta accuracy (%)	Precision (%)	Delta precision (%)	Recall (%)	Delta recall (%)	F1-score (%)	Delta F1-score (%)
Initial settings	74.08	-	75.89	-	70.46	-	73.07	-
Ar-Attack1	71.84	2.24	72.86	3.03	69.45	1.01	71.11	1.96
Ar-Attack2	70.86	3.22	71.57	4.32	68.95	1.51	70.24	2.83
Ar-Attack5	69.23	4.85	69.62	6.27	67.08	3.38	68.33	4.74

Figure 4 shows the robustness curves for the 4-evaluation metrics: accuracy, precision, recall, and F1-score, under the initial condition and the three attack scenarios (1, 2, and 5 edits). It is clear that adversarial examples can be successfully generated to fool the model by modifying or adding just one character to the most important word of the input sequence. These generated adversarial examples and reduced the model's accuracy from 74.08 to 71.84%. When the modification is applied to the two most important tokens of the input sequence, the model's accuracy is reduced to 70.86% and it is reduced to 69.23% when the modification affects the five most important tokens. The metric values reported in the table and the robustness curves for precision, recall, and F1-score exhibit patterns very similar to those of accuracy, suggesting that the two classes are fairly balanced with no dominant distribution bias.

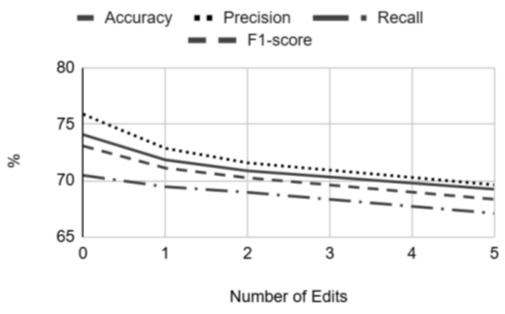


Figure 4. Robustness curves: metrics as a function of number of edits

4.4. Discussion

Tables 6 to 8 provide examples of how inputs could be misclassified when an adversarial perturbation based on auditory similarity is applied to the most important words. These minimal changes can involve altering or inserting just one, two, or five characters in a sentence. As a result, a negative review can be turned into a favorable one or vice versa.

Table 9 shows the effect of applying the three adversarial attacks to the deceived model: accuracy decreases by 2.24% for Ar-Attack1, 3.22% for Ar-Attack2, and 4.85% for Ar-Attack5 with an average of 3.44% for the three scenarios. These results show that BERT binary classifiers are sensitive to common spelling mistakes made by Arabic writers who modify a character with a similar one or add a character at the end of the word. Furthermore, given that it has resulted in a greater reduction in accuracy, combining multiple modifications—changing or adding two or five tokens instead of one—leads to a more effective attack. This clearly shows the inverse relationship between the degree of similarity and the effectiveness of the attack.

4.5. Complementary analysis

In this subsection, complementary analyses are presented to gain a deeper understanding of the behavior and effectiveness of our attack. Replace-only versus add-only strategies are first assessed, and the impact of CSF token selection is tested via an ablation. The method is then compared to a simple baseline and an existing Arabic attack, and finally the transferability of the generated adversarial examples is evaluated. All experiments are conducted under identical technical settings to ensure comparable results.

4.5.1. Component-wise analysis

In this subsection, the two attack components are evaluated separately. For each sample, the top token is selected using CSF and apply a single perturbation, either a replacement or an addition. The resulting accuracy and accuracy drop are reported in Table 10.

Table 10. Component-wise attack results

	Accuracy (%)	Delta accuracy (%)
Replace only	72.97	1.11
Add only	72.35	1.73

It is observed that add-only performs better than replace-only. Replace-only only targets a small set of characters, so if the most important token doesn't contain any of them, no change is made. By applying replace or add depending on the token, our full attack Ar-Attack1 achieves better overall performance than either component alone.

4.5.2. Ablation study

In this subsection, the role of the CSF-based token selection is examined by choosing tokens arbitrarily instead. The perturbation itself is kept the same, so differences in performance directly reflect the impact of the CSF. These results are compared with the original CSF-driven Ar-Attack1 to see how much the selection mechanism contributes to the attack's effectiveness. Accuracy after attack and the drop in accuracy are presented in Table 11.

Table 11. Ablation study results

	Accuracy (%)	Delta accuracy (%)
Ar-Attack1-original	71.84	2.24
Ar-Attack1-without CSF	72.88	1.20

With CSF-based selection, the attack reduces accuracy by 2.24%, while using a random token yield only a 1.20% drop. In other words, the CSF accounts for about 46% of the observed accuracy decrease. This confirms that targeting the most important token substantially boosts the attack's effectiveness.

4.5.3. Comparison with random character swap

In this subsection, Ar-Attack2 is compared with a simple baseline that randomly swaps two characters within a randomly selected word. Both attacks apply the same number of edits, making the comparison meaningful. Accuracy after attack and the drop in accuracy are shown in Table 12.

Table 12. Comparative results with random character swap

	Accuracy (%)	Delta accuracy (%)
Ar-Attack2	70.86	3.22
Character swap baseline	72.34	1.74

The character swap baseline yields an accuracy drop of 1.74%, whereas our Ar-Attack2 achieves a larger decrease of 3.22% under the same edit distance (two edits). Even Ar-Attack1, which uses only one edit, surpasses the baseline with a 2.24% drop. These results show that our proposed attacks generate more effective perturbations than simple character swaps, even with fewer edits.

4.5.4. Comparison with existing Arabic attack

In this subsection, Ar-Attack1 is compared with the Arabic attack introduced in [34]. Both methods are evaluated under the same number of edits to keep the comparison fair. While our approach relies on auditory similarity between Arabic characters, theirs is based on visual similarity. A quantitative evaluation is performed based on attack effectiveness, and a qualitative one assessing human perceptibility.

i) Attack effectiveness evaluation: the impact of each attack on the classifier's accuracy is first evaluated. Results are summarized in Table 13. The first column reports accuracy after the attack, and the second column shows the drop in accuracy relative to the original clean model. As shown in Table 13, the visual-similarity method produces a larger accuracy drop on the target classifier than Ar-Attack1 when both methods use the same number of edits. These results indicate that the visual-based method yields a larger proportion of inputs that the classifier mislabels than Ar-Attack1.

Table 13. Attack effectiveness comparison

	Accuracy (%)	Delta Accuracy (%)
Ar-Attack1 (auditory-based, ours)	71.84	2.24
Visual-based Arabic attack	69.8	4.28

ii) Human perceptibility evaluation: to complement the quantitative results, a human evaluation was conducted with 20 native annotators on 100 perturbed reviews, 50 visual-based and their 50 auditory counterparts. Each annotator received a distinct test set, ensuring no one saw the same review under both perturbation types. Each trial lasted 15 seconds. Results are reported in Table 14.

The results show that perturbations from Ar-Attack1 were less often detected than those from the visual-based attack. This indicates that auditory substitutions appear more natural and harder to notice.

Within Ar-Attack1, 69% of auditory-add and 33% of auditory-replace cases went unnoticed, suggesting that the additive modification is less perceptible than replacement ones.

Table 14. Human perceptibility comparison

	Number of perturbed reviews	Detection rate (%)
Ar-Attack1 (auditory-based, ours)	50	44
Visual-based Arabic attack	50	52

4.5.5. Transferability analysis

How adversarial examples generated using the Ar-Attack1 method on the source BERT-based model transfer to AraBERT—namely arabert-sentiment-model-MuhannedSh, a pretrained Arabic model fine-tuned for binary SA available on Hugging Face—is evaluated. The transferability is assessed using the transfer success rate (TSR) defined as the ratio of successful adversarial attacks on the target model to those that succeed on the source model. Results are reported in Table 15. About 31.47% of the adversarial examples that fooled BERT also fooled AraBERT, showing moderate transferability.

Table 15. Transferability results for AraBERT

	Number of successes on source	Number of successes on target	TSR (%)
AraBERT	696	219	31.47

5. CONCLUSION

This paper proposed a character-level method to generate adversarial examples in Arabic. The suggested approach is simple and hardly perceptible, requiring only a few carefully selected transformations. It operates under BB settings, without access to the internal parameters of the model. The generated adversarial examples appear natural yet effectively confuse the classifier, leading to an average accuracy drop of 3.44% for a BERT model retrained on the BRAD Arabic dataset, thus demonstrating the method's effectiveness. Compared to a character-swap baseline, our method causes a larger accuracy drop. While the visual-based replace method achieves a larger accuracy drop on Arabic text, our auditory-add perturbations demonstrate superior human imperceptibility. Building on these results, future work could explore additional attack variants, extend the method to audio-based inputs, and assess adversarial robustness beyond character-level perturbations.

ACKNOWLEDGMENTS

We would like to express our gratitude to Dr. Mostafa Ayyad for his English review of this paper.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	**C**	**M**	**So**	**Va**	**Fo**	**I**	**R**	**D**	**O**	**E**	**Vi**	**Su**	**P**	**Fu**
Safae Laatyaoui	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	
Mohammed Saber	✓	✓		✓	✓	✓				✓	✓	✓	✓	

C	:	**C**onceptualization	I	:	**I**nvestigation	Vi	:	**Vi**sualization
M	:	**M**ethodology	R	:	**R**esources	Su	:	**Su**pervision
So	:	**So**ftware	D	:	**D**ata Curation	P	:	**P**roject administration
Va	:	**Va**lidation	O	:	Writing - **O**riginal Draft	Fu	:	**Fu**nding acquisition
Fo	:	**Fo**rmal analysis	E	:	Writing - Review & **E**diting			

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [SL], upon reasonable request.

REFERENCES

[1] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR) Conference Track*, Apr. 2014.

[2] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[3] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proceedings of the ACM Conference on Computer and Communications Security*, Oct. 2011, pp. 43–58, doi: 10.1145/2046684.2046692.

[4] H. Hirano, A. Minagi, and K. Takemoto, "Universal adversarial attacks on deep neural networks for medical image classification," *BMC Medical Imaging*, vol. 21, no. 9, Jan. 2021, doi: 10.1186/s12880-020-00530-y.

[5] K. Koga and K. Takemoto, "Simple black-box universal adversarial attacks on deep neural networks for medical image classification," *Algorithms*, vol. 15, no. 5, Apr. 2022, doi: 10.3390/a15050144.

[6] M.-J. Tsai, P.-Y. Lin, and M.-E. Lee, "Adversarial attacks on medical image classification," *Cancers*, vol. 15, no. 17, Aug. 2023, doi: 10.3390/cancers15174228.

[7] H. Yin, H. Zhang, J. Wang, and R. Dou, "Boosting adversarial attacks on neural networks with better optimizer," *Security and Communication Networks*, vol. 2021, Jun. 2021, doi: 10.1155/2021/9983309.

[8] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: a survey," *ACM Transactions on Intelligent Systems and Technology*, vol. 11, no. 3, pp. 1–41, 2020, doi: 10.1145/3374217.

[9] M. Omar, S. Choi, D. Nyang, and D. Mohaisen, "Robust natural language processing: recent advances, challenges, and future directions," *IEEE Access*, vol. 10, pp. 86038–86056, 2022, doi: 10.1109/ACCESS.2022.3197769.

[10] M. V.-Hernández, L. A. M.-Rosales, I. A.-Badillo, S. I. F.-Gregorio, H. R.-Rangel, and M.-L. C.-Tlaxcalteco, "A survey of adversarial attacks: an open issue for deep learning sentiment analysis models," *Applied Sciences*, vol. 14, no. 11, 2024, doi: 10.3390/app14114614.

[11] H. Tan, L. Wang, H. Zhang, J. Zhang, M. Shafiq, and Z. Gu, "Adversarial attack and defense strategies of speaker recognition systems: a survey," *Electronics*, vol. 11, no. 14, 2022, doi: 10.3390/electronics11142183.

[12] S. Goyal, S. Doddapaneni, M. M. Khapra, and B. Ravindran, "A survey of adversarial defenses and robustness in NLP," *ACM Computing Surveys*, vol. 55, no. 14s, Jul. 2023, doi: 10.1145/3593042.

[13] G. Chang, H. Gao, Z. Yao, and H. Xiong, "TextGuise: adaptive adversarial example attacks on text classification model," *Neurocomputing*, vol. 529, pp. 190–203, 2023, doi: 10.1016/j.neucom.2023.01.071.

[14] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy Workshops (SPW)*, May 2018, pp. 50–56, doi: 10.1109/SPW.2018.00016.

[15] Y. Qi, X. Yang, B. Liu, K. Zhang, and W. Liu, "Adaptive gradient-based word saliency for adversarial text attacks," *Neurocomputing*, vol. 590, 2024, doi: 10.1016/j.neucom.2024.127667.

[16] M. Qaraei and R. Babbar, "Adversarial examples for extreme multilabel text classification," *Machine Learning*, vol. 111, no. 12, pp. 4539–4563, 2022, doi: 10.1007/s10994-022-06263-z.

[17] X. Wang, X. Yang, Y. Deng, and K. He, "Generation-based parallel particle swarm optimization for adversarial text attacks," *Information Sciences*, vol. 644, 2023, doi: 10.1016/j.ins.2023.119237.

[18] C. Wu, R. Zhang, J. Guo, M. de Rijke, and Y. Fan, "PRADA: practical black-box adversarial attacks against neural ranking models," *ACM Transactions on Information Systems*, vol. 41, no. 4, 2023, doi: 10.1145/3576923.

[19] J. Khan, K. Ahmad, and K.-A. Sohn, "An efficient character-level adversarial attack inspired by textual variations in online social media platforms," *Computer Systems Science & Engineering*, vol. 47, no. 3, pp. 2869–2894, 2023, doi: 10.32604/csse.2023.040159.

[20] I. Fursov *et al.*, "A differentiable language model adversarial attack on text classifiers," *IEEE Access*, vol. 10, pp. 17966–17976, 2022, doi: 10.1109/ACCESS.2022.3148413.

[21] S. Zhang, H. Wu, G. Zhu, X. Xu, and M. Su, "Character-level adversarial samples generation approach for Chinese text classification," *Journal of Electronics and Information Technology*, vol. 45, no. 6, pp. 2226–2235, 2023, doi: 10.11999/JEIT220563.

[22] Y. Zhang, L. Ye, B. Li, and H. Zhang, "Robustness evaluation of cloud-deployed large language models against Chinese adversarial text attacks," in *Proceedings of the 2023 IEEE 12th International Conference on Cloud Networking (CloudNet)*, Nov. 2023, pp. 438–442, doi: 10.1109/CloudNet59005.2023.10490064.

[23] X. He, F. Hao, T. Gu, and L. Chang, "CBAs: character-level backdoor attacks against Chinese pre-trained language models," *ACM Transactions on Privacy and Security*, vol. 27, no. 3, Aug. 2024, doi: 10.1145/3678007.

[24] E. Taktasheva *et al.*, "TAPE: assessing few-shot Russian language understanding," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Dec. 2022, pp. 2472–2497, doi: 10.18653/v1/2022.findings-emnlp.183.

[25] R. Kawano, K. Akimoto, and S. Ono, "ANJeL: black-box adversarial attack on deep neural networks for Japanese language," *Transactions of the Japanese Society for Artificial Intelligence*, vol. 40, no. 2, Mar. 2025, doi: 10.1527/tjsai.40-2_C-O52.

[26] R. Plant, M. V Giuffrida, N. Pitropakis, and D. Gkatzia, "Evaluating language model vulnerability to poisoning attacks in low-resource settings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 33, pp. 54–67, 2025, doi: 10.1109/TASLP.2024.3507565.

[27] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[28] A. Elnagar and O. Einea, “Brad 1.0: book reviews in Arabic dataset,” in *Proceedings of the IEEE ACS International Conference on Computer Systems and Applications (AICCSA)*, Nov. 2016, pp. 1–8, doi: 10.1109/AICCSA.2016.7945800.

[29] B. Alshemali and J. Kalita, “Adversarial examples in Arabic,” in *Proceedings of the 6th Annual Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2019, pp. 371–376, doi: 10.1109/CSCI49370.2019.00072.

[30] G. Liu and J. Guo, “Bidirectional LSTM with attention mechanism and convolutional layer for text classification,” *Neurocomputing*, vol. 337, pp. 325–338, 2019, doi: 10.1016/j.neucom.2019.01.078.

[31] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, pp. 1175–1746, doi: 10.3115/v1/D14-1181.

[32] E. Albilali, N. Altwairesh, and M. Hosny, “What does BERT learn from Arabic machine reading comprehension datasets?,” in *Proceedings of the 6th Arabic Natural Language Processing Workshop (WANLP)*, Apr. 2021, pp. 32–41.

[33] W. Antoun, F. Baly, and H. Hajj, “AraBERT: transformer-based model for Arabic language understanding,” in *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, Shared Task on Offensive Language Detection*, May 2020, pp. 9–15.

[34] B. Alshemali and J. Kalita, “Character-level adversarial examples in Arabic,” in *Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021, pp. 9–14, doi: 10.1109/ICMLA52953.2021.00010.

[35] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V Le, “XLNet: generalized autoregressive pretraining for language understanding,” in *Advances in Neural Information Processing Systems*, 2019, vol. 32, doi: 10.5555/3454287.3454804.

[36] H. Alshalan and B. Rekabdar, “Attacking a transformer-based models for Arabic language as low resources language (LRL) using word-substitution methods,” in *2023 Fifth International Conference on Transdisciplinary AI (TransAI)*, Sep. 2023, pp. 95–101, doi: 10.1109/TransAI60598.2023.00025.

[37] G. Inoue, B. Alhafni, N. Baimukan, H. Bouamor, and N. Habash, “The interplay of variant, size, and task type in Arabic pre-trained language models,” in *Proceedings of the Sixth Arabic Natural Language Processing Workshop (WANLP 2021)*, 2021, pp. 92–104.

[38] N. Alshahrani, S. Alshahrani, E. Wali, and J. Matthews, “Arabic synonym BERT-based adversarial examples for text classification,” in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop (EACL 2024)*, 2024, pp. 137–147, doi: 10.18653/v1/2024.eacl-srw.10.

[39] S. Nakhleh, M. Qasaimeh, and A. Qasaimeh, “Character-level adversarial attacks evaluation for AraBERT’s,” in *2024 15th International Conference on Information and Communication Systems (ICICS)*, Aug. 2024, pp. 1–6, doi: 10.1109/ICICS63486.2024.10638315.

[40] I. Abdellaoui, A. Ibrahimi, M. A. El Bouni, A. Mourhir, S. Driouech, and M. Aghzal, “Investigating offensive language detection in a low-resource setting with a robustness perspective,” *Big Data and Cognitive Computing*, vol. 8, no. 12, Nov. 2024, doi: 10.3390/bdcc8120170.

[41] A. O. Al-Shbail and M. A. B. Diab, “Arabic writing, spelling errors and methods of treatment,” *Journal of Language Teaching and Research*, vol. 9, no. 5, Sep. 2018, doi: 10.17507/jltr.0905.17.

BIOGRAPHIES OF AUTHORS

Safae Laatyaoui is currently a Ph.D. student and member of the Smart Information, Communication and Technologies Laboratory (SmartICT Lab) at the National School of Applied Sciences at Mohammed First University, Oujda, Morocco (2020). She earned her degree in Computer Engineering from High National School for Computer Science and Systems Analysis ENSIAS, Rabat, Morocco in 2011 and became a state engineer. She has more than nine years of experience working as a business intelligence engineer and project manager for SQLI company. Business intelligence, artificial intelligence, data science, and project management are some of her interests. She can be contacted at email: s.laatyaoui@ump.ac.ma or safae.laatyaoui@gmail.com.

Dr. Mohammed Saber is currently a full professor (PES) in the Department of Electronics, Computer Science, and Telecommunications at the National School of Applied Sciences at Mohammed First University, Oujda, Morocco (2013). He received a Ph.D. in Computer Science at the Faculty of Sciences, Oujda, Morocco, in July 2012, an engineer degree in Network and Telecommunication at the National School of Applied Sciences, in July 2004, and a licence degree in Electronics at the Faculty of Sciences, in July 2002, all from Mohammed First University, Oujda. He is currently the director of the Smart Information, Communication and Technologies Laboratory (SmartICT Lab). His interests include network security (intrusion detection systems, evaluation of security components, security IoT), AI, robotics, and embedded systems. He can be contacted at email: m.saber@ump.ac.ma.