Natural Language Processing (2024), 1–35
doi:10.1017/nlp.2024.35

ARTICLE

Towards a robust deep learning framework for Arabic sentiment analysis

Azzam Radman and Rehab Duwairi

Computer Information Systems, Jordan University of Science and Technology, Irbid, Jordan
Corresponding author: Rehab Duwairi; Email: rehab@just.edu.jo

(Received 30 April 2024; accepted 30 April 2024)

Special Issue on '**Natural Language Processing Applications for Low-Resource Languages**'

Abstract

In spite of the superior performance deep neural networks have proven in thousands of applications in the past few years, addressing the over-sensitivity of these models to noise and/or intentional slight perturbations is still an active area of research. In the computer vision domain, perturbations can be directly applied to the input images. The task in the natural language processing domain is quite harder due to the discrete nature of natural languages. There has been a considerable amount of effort put to address this problem in high-resource languages like English. However, there is still an apparent lack of such studies in the Arabic language, and we aim to be the first to conduct such a study in this work. In this study, we start by training seven different models on a sentiment analysis task. Then, we propose a method to attack our models by means of the worst synonym replacement where the synonyms are automatically selected via the gradients of the input representations. After proving the effectiveness of the proposed adversarial attack, we aim to design a framework that enables the development of models robust to attacks. Three different frameworks are proposed in this work and a thorough comparison between the performance of these frameworks is presented. The three scenarios revolve around training the proposed models either on adversarial samples only or also including clean samples beside the adversarial ones, and whether or not to include weight perturbation during training.

Keywords: Arabic sentiment analysis; adversarial attack; adversarial training; adversarial weight perturbation; deep learning

1. Introduction

Deep neural networks (DNNs) have been extensively adopted in thousands of applications in the past few years (Zhang *et al.* 2020c); however, several studies have shown the high sensitivity of DNNs to intentional, yet imperceptible perturbations (Szegedy *et al.* 2013; Goodfellow, Shlens, and Szegedy 2014). In the computer vision (CV) domain, generating adversarial examples usually revolves around slightly perturbing the input images so that the perturbations are unperceivable for human eyes. For the natural language processing (NLP) domain, on the other hand, the task is trickier due to the discrete nature of natural languages (Zhang *et al.* 2020c). This forms a barrier in the face of adversarial attack (AA) advancements in NLP as the methods applied in CV cannot be seamlessly transferred to textual tasks. For example, if term frequency and inverse document frequency are used to represent tokens, whether they are words, sub-words, or characters, using the backpropagated gradients to perturb these representations would potentially lead to invalid letter or word sequences (Zhao, Dua, and Singh 2017), which, in turn, would make it almost impossible even for humans to predict the correct output. The same problem emerges when using

© The Author(s), 2024. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

word embeddings as inputs as the perturbed vectors cannot be matched with any vector in the embedding space.

There are several taxonomies for the AA strategies, where the most popular one is based on model knowledge, i.e. having access to the architecture, weights, training data, loss function, etc. The attacks can be classified based on model knowledge to white-box such as Liang *et al.* (2017); Samanta and Mehta (2018); Rosenberg *et al.* (2018); Al-Dujaili *et al.* (2018); Cheng *et al.* (2019); Papernot *et al.* (2016); Sun *et al.* (2018); Ebrahimi *et al.* (2017); Blohm *et al.* (2018) and black-box such as Jia and Liang (2017); Wang and Bansal (2018); Belinkov and Bisk (2017); Iyyer *et al.* (2018). White-box attack algorithms depend mainly on gradients backpropagated by a model with respect to its inputs, where perturbations are made to generate the worst possible examples within a feasible range by moving in the same direction of the gradients (Goodfellow *et al.* 2014), meanwhile black-box attacks depend on heuristics such as concatenating to, editing, substituting, or paraphrasing the inputs, and are useful when no access to the model's parameters is within reach.

Adversarial training (AT), on the other hand, first introduced by Szegedy *et al.* (2013), aims at establishing neural networks *robust* to AAs. This technique has been proven to be the most effective at developing networks resistant to attacks (Pang *et al.* 2020 ; Maini, Wong, and Kolter 2020; Schott *et al.* 2018). The first work in this domain (Szegedy *et al.* 2013) focused on simply training the neural network on a mixture of benign (clean) and malignant (adversarial) examples to achieve this goal. This work opened the doors for a revolution in this field, where several proposed frameworks were designed to enhance the robustness of models against attacks. Goodfellow et al. (2014) were the first to propose exploiting the gradients generated by differentiating the loss function with respect to the inputs to generate perturbed examples on the fly via a method called fast gradient sign method (FGSM). However, one of the weaknesses of FGSM is the linear approximation of the loss function that does not improve the model's vulnerability to iterative attacks (Tramèr *et al.* 2017). This approximation leads to a sharp curvature known as gradient masking (Papernot *et al.* 2017) in the vicinity of data points on the decision surface of the trained models. This issue encouraged further improvements on AAs that targeted designing AA frameworks that simulate the worst possible realistic adversaries as will be discussed later in the Related Works section.

With regard to employing AT in Arabic NLP, there is an apparent shortage in research aiming at building *robust* deep learning (DL) frameworks. Most of the studies in the literature were targeting developing multi-lingual and cross-lingual DL models that utilize the concept of deep domain adaptation (DDA) proposed by Ganin and Lempitsky (2015) to enable knowledge transfer from models trained on high-resource languages to Arabic (Joty *et al.* 2017; Zalmout and Habash 2019; Gupta 2021; Goyal, Singh, and Kumar 2021).

One of the NLP tasks that have not been covered with AT studies in the Arabic language is sentiment analysis (SA). SA is the task of determining the affective states in a given text (Darwish *et al.* 2021). SA has received great attention among the NLP community as one of the most important applications of the field and has been incorporated in several areas including business analysis (Han *et al.* 2019; 2021), review analysis (Bose *et al.* 2020), healthcare (Clark *et al.* 2018), and stock market analysis (Xing, Cambria, and Welsch 2018). The significance of SA is mainly attributed to the nature of this area of study that aims to automatically analyze the physiological state, satisfaction, or impression of the user based on their responses (Wankhade, Rao, and Kulkarni 2022). This, in turn, would enable organizations and entities to apply improvements in regions that matter to the customers rather than spreading out the expenses on a wide range of potential, yet uncertain areas of improvement.

In this work, we aim to investigate the robustness of models trained on a standard basis against the worst synonym replacement attack. After proving the effectiveness of this attack, we design three scenarios to train *robust* models. The three scenarios are as follows: applying perturbations to the inputs and training the models on adversarial samples only, applying perturbations on

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

both the inputs and the models' trainable weights and training based on these perturbations, and training on clean and perturbed inputs in conjunction with the weight perturbation.

The remaining of this work is divided as follows. Section 2 is a literature review of the previous works where we cover four main areas, namely adversarial attack in NLP, adversarial training, using adversarial training in Arabic NLP, and Arabic sentiment analysis. Section 3 contains the materials and methods including a description of the dataset, the data preprocessing steps, sequence tokenization and word embeddings, deep learning models and training settings, evaluation metrics, adversarial attack and evaluation, and adversarial training. Section 4 contains the results and the discussion. In this section, we present the results of both the standard and adversarial training settings. Finally, Section 5 contains the conclusion and future directions.

2. Related works

2.1 Adversarial attack in NLP

In the DL field, AA is the strategy of systematically adding slight perturbations to the inputs of a pre-trained neural network so that the network is driven to generate incorrect predictions for the perturbed inputs. This area of research has been established due to the difficulty in interpreting the outputs of neural networks and defining the kind of knowledge each neuron has learned which, in return, makes the process of evaluating the robustness of a network harder (Zhang *et al.* 2020c).

The first work that aimed at investigating the robustness of the state-of-the-art image classification DL model was the one proposed by Szegedy *et al.* (2013). The study concluded the over-sensitivity of this model to pixel values as the perturbations added to the images were undetectable for humans; meanwhile, the network was confidently predicting the wrong label for the perturbed images. In order to reduce the cost of generating adversarial examples, Goodfellow *et al.* (2014) have proposed the FGSM which depends on fast generation of adversarial images based on the backpropagated gradients with respect to the inputs.

With regard to the NLP domain, Jia and Liang (2017) were the first to evaluate DL models on adversarial examples. Their method depended on concatenating distracting, yet meaningless, collection of words to the end of a given text that is used to train a model to solve a question-answering task. One main restriction that has been used to filter out the appended sentences is that the new fooling collection of words must not change the semantics of the main text or alter the answers to the questions. The concatenated sentence can be either a carefully crafted sentence or an arbitrary sequence of tokens randomly selected from a pool of 20 random frequently used tokens. Wang and Bansal (2018) proposed expanding the number of fake answers and changing positions where distracting sentences are added in order to build more *robust* DL models.

Another black-box word-level attack that is based on sentence concatenation is the one proposed by Blohm *et al.* (2018). In this work, the generated sentence was formed by randomly selecting words from all the words in the incorrect answers in conjunction with a pool of 10 random common words and all the words in the question.

Editing the input sequences is another black-box technique used for generating AA. In their work, Belinkov and Bisk (2017) targeted neural machine translation models. The inputs to these models were perturbed in one of three ways: leveraging typos that already exist in textual datasets, altering the order of all the characters of a word except for the first and last characters, or fully replacing the characters of words with random ones. Niu and Bansal (2018) approached the dialogue generation task and used multiple perturbation methods including the following: random swapping by randomly replacing neighboring words, paraphrasing, randomly dropping stop-words, intentionally using the wrong tense form of randomly selected verbs, and using the negations and the antonyms. Gao *et al.* (2018) used four strategies in generating adversarial examples, namely addition, deletion, replacement, and swapping. The altered words or characters were selected based on their importance which was measured by means of a scoring function that

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

leverages the classifier's output. A probabilistic method for generating adversarial examples has been proposed by Ren *et al.* (2019). They first started by creating a dictionary that contains the synonyms of all the words in an existing corpus. Then, a set of proposed words are substituted with the synonyms that most fool the network such that the replacements make the most impact on the classification probability. The replacement order is defined by means of word saliency. Zhang *et al.* (2020a) proposed the Metropolis-Hastings (M-H) Attack. They targeted language models in their work and employed M-H sampling to generate the replacing words and the random words for both the replacement and insertion operations, respectively.

Other works in AA were based on white-box techniques which require the availability of information on the model's architecture, loss function, inputs, and outputs in order to extract the gradients that can be later used to generate the perturbations. TextFool (Liang *et al.* 2017) is one of the leading techniques in the context of textual AA and is inspired by FGSM (Goodfellow *et al.* 2014). However, in this work the authors propose the use of the gradient magnitudes rather than their signs to create the adversaries and propose three methods for the AA generation: modification, deletion, and insertion. Based on the absolute backpropagated gradients, the authors define the hot characters, which represent the characters with highest magnitudes and hence highest probability to impact the model. Then, they defined the hot training phrases (HTPs) as phrases that contain enough hot characters with a sufficient frequency of occurrence. During the insertion process, the AA is created by inserting a small proportion of HTPs of the wrong class that is targeted (C′) by the AA in the vicinity of the hot characters that contribute the most to the ground-truth class C. For the modification version, the authors replaced the characters of HTPs with randomly selected characters or characters that are visually similar.

Samanta and Mehta (2018) have shown better results compared to TextFool (Liang *et al.* 2017) by performing some modifications to the algorithm. They first start by removing the adverb (w_i) that contributes the most to the predicted label. If the generated text contains grammatical mistakes due to the removal operation, a word (p_i) is inserted before (w_i) where (p_i) is selected from a predefined pool that contains synonyms, genre keywords retrieved by means of term frequency, and typos. Finally, if the inserted (p_i) fails to increase the value of the loss function, (w_i) is replaced with (p_i).

In their work, Al-Dujaili *et al.* (2018) have proposed the generation of binary-encoded AAs. Four bounding methods were used in this work to generate the perturbations, two of which depended on FGSM^k (Kurakin, Goodfellow, and Bengio 2016) with the deterministic rounding (dFGSM^k) and the randomized rounding (rFGSM^k) versions. These two methods are identical to the L_∞-ball method (Goodfellow *et al.* 2014) used to constrain the perturbations for CV tasks. The third and fourth methods employ the multi-step bit gradient ascent and the multi-step bit coordinate ascent, respectively.

2.2 Adversarial training

The goal of using AT is to produce neural networks *robust* to attacks. Several techniques have been proposed in this area of study and most of these techniques focus on training the neural network on adversarial examples. As mentioned in Subsection "Adversarial Attack in NLP," Szegedy *et al.* (2013) were the first to employ training on both clean and adversarial samples. This work was followed by the emergence of FGSM (Goodfellow *et al.* 2014) that proposed an efficient and fast way to generate adversaries and train the model on these crafted examples. The linear approximation of the loss function was one of the disadvantages of this method that becomes more apparent when testing the model on iterative attacks (Tramèr *et al.* 2017).

Apart from the previous works where models were trained on a mixture of clean and perturbed data, a stream of research has taken the path of training on adversarial examples only, such as Huang *et al.* (2015) and Shaham *et al.* (2018). However, training on adversarial examples only renders the model vulnerable to overfitting the adversarial examples (Zhang and Xu 2019),

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

especially with relatively strong adversaries that generate samples that cross the decision boundaries and are more like natural samples, yet belonging to different classes.

Several frameworks were proposed to address the problem of overfitting to adversarial examples, including the Curriculum AT (Cai *et al.* 2018) that gradually increases the steps of the projected gradient descent (PGD) (Madry *et al.* 2017) to generate stronger attacks during the course of training until the model learns the sufficient weights to overcome the attacks. Zhang *et al.* (2020b) proposed Friendly AT that employs early stopping when searching for adversarial examples using PGD to generate friendly adversarial data to minimize the adversarial loss rather than maximizing it. They concluded that adversarial robustness does not contradict natural generalization.

Other AT methods rely on adding a regularization term in the loss function that minimize the distance between the clean and perturbed examples. Zhang *et al.* (2019) proposed TRadeoff-inspired Adversarial DEfense via Surrogate-loss minimization, a regularized loss function that penalizes adversarial examples that have a distribution with a large Kullback-Leibler divergence (Kullback and Leibler, 1951) from the distribution of the clean examples. This regularized loss does not take into consideration whether or not the benign samples have been classified correctly by the model. Wang *et al.* (2019) proposed Misclassification Aware adveRsarial Training which incorporates the probability of the ground-truth label generated by the model to emphasize on samples that have been already correctly classified in the clean mode.

One of the most commonly used AT technique in the natural language processing domain is DDA proposed by Ganin and Lempitsky (2015) which aims at learning domain-invariant, yet discriminative representations. In this algorithm, the model is trained on two datasets, one labeled and one unlabeled, and is equipped with two heads: a predictor for the main task and a discriminator to distinguish between the samples between the sources of the samples. One of the critical factors for the success of this algorithm is the gradient reversal layer, which simply multiplies the gradients derived from the domain classification head by a negative scalar to encourage the feature extractor part to learn domain-invariant features.

Finally, in this work, we aim at using one of the most successful AT techniques that have proved its effectiveness on several benchmark datasets, namely Adversarial Weight Perturbation-based AT (AT-AWP) proposed by Wu *et al.* (2020). The algorithm aims at producing a double-perturbation effect on the network by iteratively perturbing both the input samples and model weights through PGD.

2.3 Using adversarial training in Arabic NLP

There is a severe shortage in the literature regarding works employing AT in Arabic NLP. Most of these works use DDA to approach multi-lingual or cross-lingual tasks by enabling knowledge transfer from a high-resource language with plenty of labeled datasets to another low-resource language, mostly unlabeled datasets (Joty *et al.* 2017; Zalmout and Habash 2019; Gupta 2021; Goyal *et al.* 2021).

Among those works, Chen *et al.* (2018) exploited the availability of abundant annotated resources for sentiment classification in the English language by employing the AT algorithm proposed by Miyato *et al.* (2016) to learn discriminative, language-invariant features that can transfer to other low-resource languages. The model performance was evaluated using two unlabeled Arabic and Chinese datasets, where the language discriminator was trained to distinguish between the labeled and unlabeled samples; meanwhile, the sentiment classification head was trained to predict the polarity of the input samples.

Qin *et al.* (2021) propose the employment of a regularized decoder and AT to approach the Arabic diacritization problem. Previous works treated the auto-generated diacritics as gold labels; meanwhile, a high fraction of these labels is inaccurate and would potentially lead to a model producing deficient representations. In this work, the authors developed a model that employed

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

AT inspired by Ganin and Lempitsky (2015) to balance the information learned from both the main tagger and the regularized decoder.

Joty *et al.* (2017) designed an adversarial training-based framework inspired by Ganin and Lempitsky (2015) to learn discriminative, language-invariant representations to approach the task of question-question similarity re-ranking. The proposed model is trained to receive a batch of labeled and unlabeled inputs from two different languages and is encouraged to discriminate between the labeled samples from the unlabeled ones through the label classifier network. At the same time, a gradient reversal layer is adopted to encourage the model to learn language-invariant features that can fool the language discriminator.

Zalmout and Habash (2019) employed both adversarial training and multitask learning in addressing the data sparsity problem in the Arabic language that stems from both morphological richness and dialectical variations. This paper was the first to use adversarial domain adaptation (Ganin and Lempitsky 2015; Ganin *et al.* 2016) in the field of dialectical morphological adaptation. The proposed framework was evaluated on two datasets including Modern Standard Arabic (MSA) and the Egyptian dialect.

Gupta (2021) investigated the possibility of leveraging unlabeled data from different languages to improve the performance on the multi-label emotion recognition task. They formulated a semi-supervised Virtual Adversarial Training (VAT (Miyato *et al.* 2018) problem and investigated the improvement in the performance of a target language classification task driven by leveraging unlabeled datasets of other low-resource languages.

Goyal *et al.* (2021) approached the task of word in context disambiguation proposed by SemEval-2021 Task 2 (Martelli *et al.* 2021) in both multi-lingual and cross-lingual settings via a single XLM RoBERTa (Conneau *et al.* 2019) model. The authors reported a significant boost in performance of the model when employing the adversarial training stage proposed by Miyato *et al.* (2016), which simply perturbs the word embeddings during training using the calculated gradients with respect to the embedding vectors. The authors added a simple modification to the adversarial training algorithm by skipping the embedding normalization process which they believed would affect the semantic meaning of the pre-trained word embeddings.

Based on this literature review, we note the lack of use of AT techniques in the Arabic NLP domain to build *robust* frameworks and the need to thoroughly investigate the applicability of such techniques in this domain.

2.4 Arabic sentiment analysis

Sentiment analysis (SA) is the task of determining the affective states in a given text (Darwish *et al.* 2021). Due to its important applications, SA has been studied extensively in the literature. These applications include the following: business analysis (Han *et al.* 2019; Bose *et al.* 2020), review analysis (Mackey, Miner, and Cuomo 2015), healthcare (Clark *et al.* 2018), and stock market analysis (Xing *et al.* 2018). The significance of SA is mainly attributed to the nature of this area of study that aims at automatically analyzing the physiological state, satisfaction, or impression of the user based on their responses (Wankhade *et al.* 2022).

Several works have approached the task of SA in the Arabic language domain, however, these studies still report the challenges faced due to the nature of this language. These challenges, including morphological richness, orthographic ambiguity, orthographic inconsistency, and resource poverty, hinder the research progress in Arabic NLP (Darwish *et al.* 2021). For example, Khalifa *et al.* (2016) have reported the drop in the performance of the state-of-the-art tool for the task of part-of-speech tagging and lemmatization from 96% on MSA on both tasks (Pasha *et al.* 2014) to around 72% and 64% on the Arabic Gulf dialect. Orthographic inconsistency, on the other hand, plays a major role in impeding the progress in Arabic NLP. The great variation in spelling the same words makes the task of indexing and tokenization much harder, and researchers need to spend tremendous amounts of time on spell checking before applying the typical text processing stages. According to Zaghouani *et al.* (2014), one-third of all MSA words available online are misspelled.

Early efforts in Arabic SA revolved around the collection and creation of the resources required to approach this task including the labeled datasets, sentiment treebanks, and sentiment lexicons (Abdul-Mageed and Diab 2012; Mourad and Darwish 2013; Badaro *et al.* 2014; Refaee and Rieser 2014; ElSahar and El-Beltagy 2015; Eskander and Rambow 2015; Khalil *et al.* 2015; Salameh, Mohammad, and Kiritchenko 2015; Shoukry and Rafea 2015; El-Beltagy 2016; Baly *et al.* 2017). Moreover, some efforts have been put to create benchmarks against which different approaches can be fairly compared (Mohammad *et al.* 2018; Rosenthal, Farra, and Nakov 2019).

Arabic SA has been approached using three main methods including hand-crafted rules and lexicons (El-Beltagy and Ali 2013; Abdul-Mageed and Diab 2014; Badaro *et al.* 2014), machine learning algorithms (Baly *et al.* 2017; Badaro *et al.* 2018; Farha and Magdy 2019), and hybrid frameworks that combine the first two approaches (Al-Smadi *et al.* 2019a). More recent approaches (Abdul-Mageed *et al.* 2020; Antoun, Baly, and Hajj 2020) include the fine-tuning of large pre-trained models such as AraBERT (Antoun *et al.* 2020) have reported state-of-the-art results in Arabic SA.

DL has been extensively used to solve SA tasks in the English language. One of the main advantages of the use of DL models is the possibility to build an end-to-end fully automated framework that can infer the important features to achieve a task like SA which is, in its core, a conventional classification task. This advantage relieves the restriction of the need for domain experts to handcraft the rules based on which the SA task can be accomplished (Badaro *et al.* 2019). As a result, different DL architectures have been used in the English NLP domain spanning a wide range of sophisticated trainable layers including recursive neural networks (RNNs) (Socher *et al.* 2013), convolutional neural networks (CNNs) (Kalchbrenner, Grefenstette, and Blunsom 2014), gated recurrent neural networks (GRNNs) (Tang, Qin, and Liu 2015), dynamic memory networks (DMNs) (Kumar *et al.* 2016), and the human reading for sentiment framework (Baly *et al.* 2016).

Inspired by this literature in English SA, several Arabic SA studies have emerged starting from Al Sallab *et al.* (2015) who evaluated different DL models including DNNs, deep belief networks (DBNs), and deep autoencoders (DAEs) which were trained based on word n-grams. Their work outperformed the previous Arabic state-of-the-art benchmark SVM models; however, the model severely suffered from the data sparsity problem, i.e. the scarcity of mentioning the vast majority of the corpus tokens which does not allow a thorough understanding of the language by the model. This arouses the need for using Arabic word embeddings.

Al-Sallab *et al.* (2017) proposed a recursive deep learning model for opinion mining in Arabic AROMA which was introduced as a solution for the problems that the author believed were the causes of the gap between the performance of the models on Arabic and English SA tasks. The authors remarked a number of potential limitations in the Arabic language as compared to English including lexical sparsity and non-standardized dialects which causes different spellings for the same word sense.

CNNs have been exploited for the purpose of Arabic SA in many works (Dahou *et al.* 2016; Gridach, Haddad, and Mulki 2017; Alayba *et al.* 2018b) utilizing the concept of learnable word embeddings. Alayba *et al.* (2017, 2018a) compared the performance of CNN and long-short-term memory (LSTM) layers by feeding them the features extracted from the trainable word, character, and character n-gram embeddings. The accuracy of CNNs was reported to be significantly better than LSTMs; 90% compared to 85%, respectively.

Al-Azani and El-Alfy (2018) compared the performance of both LSTMs and GRNNs in both the unidirectional and bidirectional settings. It was reported the outperformance of the bidirectional LSTMs compared to traditional classification methods when emojis are inserted into the inputs. Al-Azani and El-Alfy (2017) also studied and evaluated the performance of LSTMs, GRNNs, CNNs, and the combination of them on the Arabic SA task. The hybrid system of both LSTMs and CNNs outperformed the other architectures on two datasets.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Others prefer to process the whole sequence as a single unit by generating paragraph-based embeddings rather than token-based embeddings (Barhoumi *et al.* 2017; Abdullah and Shaikh 2018). Barhoumi *et al.* (2017) employed the sentiment annotated corpus large-scale Arabic book reviews (LABR) Aly and Atiya (2013) and Doc2Vec (Le and Mikolov 2014) to generate paragraph embeddings for Arabic contents and then built a classifier (MLP or Logistic regression) on top of these feature vectors.

LSTMs (unidirectional and bidirectional), GRNNs, and CNNs have been pretty much popular at the task of Arabic SA before the advent of the transformers. González *et al.* (2017); Al-Smadi *et al.* (2019b); Barhoumi *et al.* (2017); El-Beltagy *et al.* (2017, 2016) have used CNNs or LSTMs or a hybrid of both to generate the features used by the end classifier for Arabic SA.

Several techniques have been proposed to improve the models' performance on SA tasks such as the one suggested by Duwairi and Abushaqra (2021). In this paper, the authors designed a novel framework to augment the dataset used in Arabic SA taking advantage of the morphological richness of the language. The authors predefined 23 transformation rules based on which the dataset was augmented. The algorithm has been reported to increase the size of the initial seed dataset by 10 folds with a significant increase in the model's accuracy.

Based on this literature review, and to the best of our knowledge, there have been very few works that employed AT in the Arabic language domain. These works were designed to develop domain-invariant models capable of transferring knowledge from one domain to another rather than employing AT in developing models *robust* to adversarial attacks. In this paper, we aim, in the first stage, at building several DL models, train them under standard conditions to solve an Arabic SA task, and then evaluate their performance on previously unseen data both clean and perturbed. The kind of perturbation intended to be used in this work is the fast gradient projected method proposed by Wang *et al.* (2021) with some modifications to cope with the scarcity of Arabic sources. In the second stage, we will train the same models; however, this time adversarially and evaluate their performance on both clean and adversarial examples that have not been passed to the model during training. The method selected for AT is adversarial weight perturbation proposed by Wu *et al.* (2020) which has recently proved its effectiveness on several Kaggle NLP competitions[a].[b]

3. Materials and methods

3.1 Dataset

The dataset used in this work is the LABR dataset collected by Aly and Atiya (2013). The dataset consists of over 63,000 Arabic book reviews, where each textual review is associated with a rating between 1 and 5. The dataset has been shuffled and split into training and test sets by the authors to form a benchmark against which different approaches can be fairly evaluated. Moreover, the authors established other criteria to split the dataset based on including the balance of the target labels and the number of classes to consider.

In this work, we select the balanced partitioning that contains 16,448 samples. The authors use the technique of downsampling to balance the dataset which limits the number of samples in each class to the class with the minimum number of samples. For the number of classes, we select to approach this task as a binary classification task, where each sequence is classified as either positive or negative. In this case, classes 4 and 5 are merged into class 1 (positive), classes 1 and 2 are merged into class 0 (negative), and class 3, which indicates the neutral class, is omitted.

After preprocessing the data as explained in the *Data Preprocessing* subsection, the clean dataset that can be used for training the DL models was reduced to 14,309 samples. Besides the original splitting of the dataset into training and test splits, we extract a validation split from the training data to secure fair hyperparameter tuning on this split and unbiased evaluation on the test split.

[a] https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes/discussion/323095

[b] https://www.kaggle.com/competitions/feedback-prize-2021/discussion/313177

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

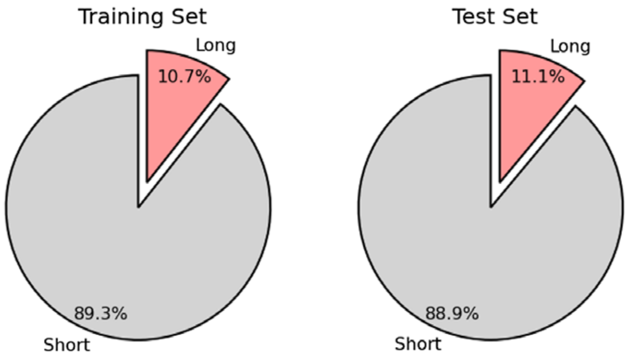

Figure 1. Long and short sequence percentages in the training (left) and test (right) datasets.

Eventually, we ended up with training, validation, and test splits of 9,981, 1,462, and 2,902 samples, respectively. The validation set was randomly selected after shuffling the training data and securing the balance of labels in the validation split via the stratified K-fold cross-validation.

3.2 Data preprocessing

First of all, we remove the long sequences that contain over 128 tokens after tokenization which is achieved using the natural language toolkit (NLTK) (Bird, Klein, and Loper 2009). This step of removing long sequences reduces the computational load and eventually allows for faster iterating. Figure 1 shows the percentages of the long and short sequences in both the training and test splits. As explained in the *Dataset* subsection, we end up with 11,443 examples in the training set (before splitting it into train and validation splits) and 2,902 examples in the test set.

After that, we remove the Arabic diacritics from the reviews as these tokens are not represented in AraVec (Soliman, Eissa, and El-Beltagy 2017) which contains the pre-trained word embeddings as will be explained later. Then, all the unique characters in both the training and test splits were retrieved, and we ended up with 93 and 92 unique characters in both datasets, respectively. A list containing all the Arabic 45 characters was created and all the non-Arabic characters were detected in both datasets based on this list and replaced with a space that represents the separator token. There are actually 28 characters in the Arabic language, however, some characters exist in different forms like “ ا”–“A” which can be found as “ أ”–“ ¿” and “ إ”–“ ¡”.

Any character that appears more than twice consecutively in the same word was replaced with a single character as this cannot take place in the Arabic language. This process included the spaces. Furthermore, some of the characters in the dataset were written in a way different from the ones available in the pre-collected characters. For example, “ ت”–“t” can be found as “ تـ” and “ ع”–“E” can be found as “ عـ” which will eventually lead to an out-of-vocabulary (OOV) token when tokenizing and embedding. Moreover, for the sake of normalization, AraVec replaces a set of characters with similar ones such as “ ة”–“p” is always replaced with “ ه”–“h”, “ أ”–“ ¿”, “ إ”–“ ¡” are always replaced with “ ا”–“A”, and “ ؤ”–“&” and “ ئ”–“}” are always replaced with “ ء”–“’”. Hence, we perform all these normalizations in order to minimize OOV tokens when embedding our sequences.

3.3 Sequence tokenization and word embeddings

First, all the input sequences were tokenized using the NLTK tokenizer (Bird *et al.* 2009). After that, we added two special tokens to the dictionary, namely the padding token “ كلمةإضافية ” and the unknown token “ غيرمعروف.” After that, each token was given an index starting from 0 and

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

ending with the number of tokens in AraVec (Soliman *et al*. 2017) in addition to the two special tokens. AraVec contains Arabic word embeddings pre-trained on Arabic corpus based on the word2vec algorithm (Mikolov *et al.* 2013). There are several versions of AraVec, and in this work, we employ the Twitter-CBOW version where each token is represented with a 300-d vector.

3.4 Deep learning models and training settings

Seven different DL models are used in this study: LSTM-based (Hochreiter and Schmidhuber 1997), gated recurrent unit (GRU)-based (Chung *et al.* 2014), CNN-based (LeCun *et al.* 1998), multi-headed self-attention (Vaswani *et al.* 2017) (MHA)-based, LSTM-CNN-based, GRU-CNN-based, and MHA-CNN-based models. Figure 2 illustrates the model architectures used in this work. The architectures, in general, begin with the embedding layer, and AraVec's pre-trained weights are used to initialize the weights of this layer.

For the first architecture, shown in Figure 2—left, we pass these embeddings to one of three layers: LSTM, GRU, or CNN. For the LSTM and the GRU layers, we use the bidirectional setting with the two sequential layers where we use 128 and 64 units, respectively, in the two layers. Following the latter are both the global average pooling and global max pooling layers which reduce the dimensionality of the previous output and generate a 1D representation for each input sequence. The pooled representations are then concatenated. Using both pooling layers and concatenating their outputs rather than depending on one of them gives the model a better chance to learn from the features of each layer which might not be easy to extract by the other one. The outputs are then passed to a fully connected layer with 64 units and ReLU activation. Finally, a head classifier with two units and softmax activation is used to predict the probability of each class given the input sample.

For the CNN-based model, the same previous architecture is used with the replacement of the LSTM/GRU layers with two sequential 1D CNN layers. The two layers have 128 kernels with a 3×3 kernel size and ReLU activation. The remaining part of the architecture is the same as the LSTM and GRU ones. For the MHA-based model, we replace the LSTM layers with two MHA layers where the first of which receives its inputs from the summation of the embedded vectors and the sinusoidal positional encodings generate the same way as in Vaswani *et al.* (2017). The first and second MHA layers have eight heads both and key dimensions of 128 and 64, respectively. Finally, three additional architectures are used that employ the LSTM-based, GRU-based, and MHA-based with an additional 1D CNN layer in between the embedding layer and these layers. The 1D CNN layer has 128 3 × 3 kernels and is ReLU-activated.

For the training process, we first freeze the weights of the embedding layer in order to preserve the semantics embedded in the vectors. This process of freezing the weights of the embedding layer is essential for the AA process where the attack depends mainly on synonym replacement that requires this preservation of vector semantics as will be explained later in the Adversarial Attack and Evaluation Subsection. Each model is trained on the training set for 10 epochs with an initial learning rate of 1e-3 that is linearly decaying over the batches until reaching the minimum value of 3e-4 with the last batch of the final epoch. The training data is continuously shuffled with a buffer size of 256. The batch size used is also set to 256. Adam optimizer with weight decay (Loshchilov and Hutter 2017) is used to optimize the weights of the models. The weight decay is set to 1e-4 which helps remedy the overfitting problem.

3.5 Evaluation metrics

The loss function alongside both the weighted F1-score and the accuracy score is used to monitor the training progress. The loss function used is the binary cross-entropy (BCE) loss function, the mathematical formulation of this loss function is presented in equation (1). We also preferred to maintain both the weighted F1, formulated in equation (2), and the accuracy, formulated in

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

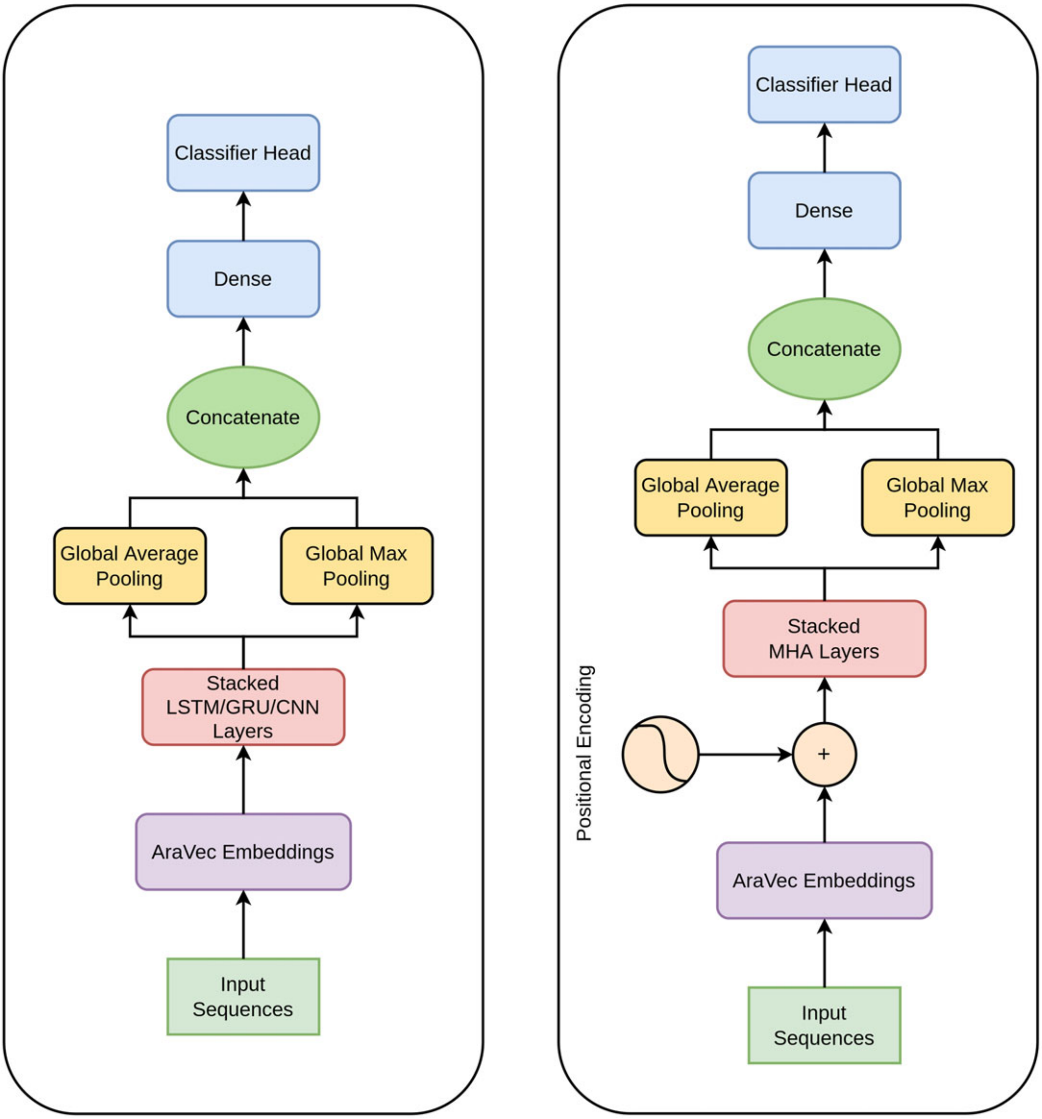

Figure 2. Model Architectures. Left is the architecture of the LSTM-based, GRU-based, and CNN-based models, and right is the architecture of the MHA-based model. For the LSTM-CNN-based, GRU-CNN-based, and MHA-CNN-based architectures, an additional 1d CNN layer is added in between the AraVec embeddings and the next layer.

equation (3), scores as measuring metrics to produce comparable results to the LABR benchmarks which used both of these metrics.

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(y_i) + (1 - y_i)\log(1 - y_i)\right] \tag{1}$$

$$Weighted\ F1\ score = \frac{1}{n}\sum_{c=0}^{C-1} n_c \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2}$$

$$Accuracy = \frac{1}{n}(TP + TN) \tag{3}$$

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

where n is the number of samples, y are the actual values, $\widehat{y_i}$ are the predicted values, C is the number of classes, n_c is the number of samples in class c, TP are the true positives, FP are the false positives, and FN are the false negatives.

3.6 Adversarial attack and evaluation

In this work, we employ an automatic synonym replacement technique for AA. Before starting the AA, we start with pre-defining the 10 nearest neighbors to each token in the dictionary based on the cosine similarity between the representations of these tokens in AraVec embeddings. Constructing this dictionary takes place as follows. First, we create an empty dictionary and populate its keys with the unique tokens of AraVec. After that, we iterate over each token embedding and compute the cosine similarity between the current token embeddings and all other embeddings. For each token, we end up with a vector of the length of the number of unique tokens in AraVec excluding the current token. This vector represents the cosine similarity between the token embeddings. Based on that, we select the ten tokens with the highest cosine similarity with the current token. These ten tokens are added to the dictionary as values corresponding to the input token key.

During AA, we perform the synonym replacement which is based on PGD (Madry *et al.* 2017) with an additional nearest neighbor's post-searching step. First, the model makes a forward propagation on the input sequences. Then, the loss is computed and the gradients are derived. We then use the gradients of the model with respect to the input embeddings to create the perturbations. As the representation of the inputs has the dimensions of [batch size, 128, 300], where 128 is the sequence length and 300 is the length of the representation of each token, also their gradients have the same dimensions. We first project the gradients to make them have the same L$_2$-norm as the corresponding inputs. This projection takes place according to equation (4).

$$Projected\ Perturbations = \frac{\nabla_x(\theta, x, y)}{\|\nabla_x(\theta, x, y)\|_2}\|x\|_2 \tag{4}$$

where $\nabla_x(\theta, x, y)$ are the gradients with respect to input(s) x given network f_θ, where θ are the network's weights, and label(s) y. $\|.\|$ is the L$_2$-norm.

We then add these projected perturbations to the input embeddings, going in the same direction of maximizing the loss. The newly generated adversarial examples do not represent any word in the embedding space, and hence we need to approximate the adversarial example by searching for the nearest neighbors to them. The searching step here depends on the predefined nearest neighbors dictionary that has been collected prior to the AA stage and the process takes place based on the cosine similarity metric. After generating these adversarial examples, they are forward propagated through the network, and all the metrics are computed to evaluate the model's performance adversarially. Algorithm 1 shows the steps of the proposed AA technique in detail. Table 1 exhibits some of the adversarial examples and their translations generated from clean ones based on the LSTM model after applying standard training.

3.7 Adversarial training

This stage is inspired by the AT-AWP proposed by Wu *et al.* (2020). This adversarial training algorithm generates perturbations based on the input examples as well as the network's trainable weights, causing a double-perturbation effect. It tries to flatten the loss landscapes of both the inputs and the weights resulting in networks much more *robust* to attacks.

As this stage of AT contains two steps, namely input perturbation and weight perturbation, we perform the input perturbation first in the same way as explained in the *Adversarial Attack and Evaluation* subsection. Then, we perform the weight perturbation by means of the adversarial examples, generated in the first step, as follows. The perturbed inputs are forward propagated

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Table 1. Examples of clean and adversarial samples for the LSTM-based model after applying standard training

Clean	Adversarial
“راءعه من ابداعات توفيق الحكيم” “Wonderful creations of Tawfiq Al-Hakim”	“ممتعه فمن ابداع توفيق القاريء” “Enjoyable, it is the creativity of Tawfiq Al-Hakim”
“كتاب جميل” “Beautiful book”	“بكتاب ممتع” “Interesting book”
“عاديه” “Conventional”	“مستهلكه” “Outdated”
“راءع لازم يبقي في كل بيت منه نسخه” “Wonderful, a copy must be kept in every home”	“ممتع المفروض يضل ب اي منزل منها صوره” “Interesting, a copy of it must be in each house.”
“التفاصيل مقرفه” “Disgusting details”	“تفاصيلك وقحه” “Your details are unpleasant”

Algorithm 1. Adversarial attack.

```
Input: Network $f_w$, training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i$ are the AraVec embeddings of
    the input tokens of each sequence, $y_i$ are the labels, nearest neighbors
    dictionary $NN$, learning rate $\eta$, batch size $m$, loss function $L$, number of
    epochs $T$.
Output: Adversarial examples $x_{adv}$.
for $i \leftarrow 0$ to $n-1$ by $m$ (in parallel) do
    $\widehat{y_i} = f_w(x_i)$                                          ▷ Forward propagation.
    $l = L(y_i, \widehat{y_i})$                                          ▷ Loss computation.
    $\nabla_{x_i}(\theta, x_i, y_i) = \frac{\partial l}{\partial x_i}$          ▷ Gradients computation with respect to inputs.
    $\nabla_{x_i(projected)}(\theta, x_i, y_i) = \frac{\nabla_{x_i}(\theta, x_i, y_i)}{\|\nabla_{x_i}(\theta, x_i, y_i)\|_2}\|x_i\|_2$     ▷ Gradients projection.
    $x_i(perturbed) = x_i + \nabla_{x_i(projected)}(\theta, x_i, y_i)$     ▷ Adversarial example generation.
    $x_i(adv) = \arg\max_{x \in NN_{x_i}} S_C(x_i(perturbed), x)$     ▷ Nearest neighbor Search, $S_C$ is cosine
                                                                        similarity.
    Evaluate network $f_w$ on $x_i(adv)$.
end for
```

through the network and the loss is computed. After generating the gradients with respect to the trainable weights, the weights get perturbed by these gradients after they are projected based on the weights of the corresponding layers. The projection process takes place the same way as formulated in equation (4), however, this time the projection is for the gradients of the weights rather than the network's inputs. After perturbing the weights, we calculate the difference between the original and the perturbed weights in a layer-wise manner. Then, we train the network either using the perturbed examples alone (in one setting) or both the clean and perturbed examples (in

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Algorithm 2. Perturb inputs only (scenario 1).

```
input: Network $f_w$, training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i$ are the AraVec embeddings of
    the input tokens of each sequence, $y_i$ are the labels, nearest neighbors
    dictionary $NN$, learning rate $\eta$, batch size $m$, loss function $L$, number of
    epochs $T$.
Output: Robust network $f_w$.
for $t \leftarrow 0$ to $T-1$ do
    for $i \leftarrow 0$ to $n-1$ by $m$ (in parallel) do
        $\widehat{y_i} = f_w(x_i)$                                          ▷ Forward propagation.
        $l = L(y_i, \widehat{y_i})$                                          ▷ Loss computation.
        $\nabla_{x_i}(\theta, x_i, y_i) = \frac{\partial l}{\partial x_i}$        ▷ Gradients computation with respect to inputs.
        $\nabla_{x_i(projected)}(\theta, x_i, y_i) = \frac{\nabla_{x_i}(\theta, x_i, y_i)}{\|\nabla_{x_i}(\theta, x_i, y_i)\|_2}\|x_i\|_2$     ▷ Gradients projection.
        $x_i(perturbed) = x_i + \nabla_{x_i(projected)}(\theta, x_i, y_i)$   ▷ Adversarial example generation.
        $x_i(adv) = \arg\max_{x \in NN_{x_i}} S_C(x_i(perturbed), x)$ ▷ Nearest neighbor Search, $S_C$ is cosine
                                                                      similarity.
        Train $f_w$ on $(x_i(adv), y_i)$
    end for
end for
```

another setting). Finally, we add the differences that have been calculated between the original and perturbed weights in order to reset the perturbation and allow the new batch to have almost the same contribution to the perturbations. The advantage of perturbing the weights beside the input samples is that these attacks are generated based on all the samples in the input batch rather than individual samples which ultimately approximates the worst AA much better than if only the inputs are perturbed. This, in turn, forces the network to learn weights that make it more *robust* to attacks.

When performing AT, we experiment with three different scenarios. The first scenario is training the network on perturbed samples only, without perturbing the weights, refer to Algorithm 2. The second scenario is training the network on both perturbed samples and perturbed weights, refer to Algorithm 3. The third scenario is using both clean and perturbed samples to train the network on the perturbed weights, refer to Algorithm 4. In the third case, we choose to apply training using clean and adversarial samples employing the second scenario rather than the first one because of its higher performance.

4. Results and discussions

4.1 Background

In this section, we display and discuss the results of the LSTM-based model which has proved the best performance among the other architectures after being adversarially trained. For the other models, the patterns are similar, and hence, there is no need to repeat similar discussions. However, we supply the figures of the other models in Appendix One on page 46 for the readers'

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Algorithm 3. Perturb both inputs and weights (scenario 2).

```
Input: Network $f_w$, training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i$ are the AraVec embeddings of
    the input tokens of each sequence, $y_i$ are the labels, $w$ model's original
    weights, nearest neighbors dictionary $NN$, learning rate $\eta$, batch size $m$,
    loss function $L$, number of epochs $T$.
Output: Robust network $f_w$.
for $t \leftarrow 0$ to $T-1$ do
    for $i \leftarrow 0$ to $n-1$ by $m$ (in parallel) do
        $w_{original} = w.copy()$    ▷ Copy original weights.
        $\widehat{y_i} = f_w(x_i)$    ▷ Start input perturbation (forward propagation).
        $l = L(y_i, \widehat{y_i})$    ▷ Loss computation.
        $\nabla_{x_i}(\theta, x_i, y_i) = \frac{\partial l}{\partial x_i}$    ▷ Gradients computation with respect to inputs.
        $\nabla_{x_i(projected)}(\theta, x_i, y_i) = \frac{\nabla_{x_i}(\theta, x_i, y_i)}{\|\nabla_{x_i}(\theta, x_i, y_i)\|_2} \|x_i\|_2$    ▷ Gradients projection.
        $x_i(perturbed) = x_i + \nabla_{x_i(projected)}(\theta, x_i, y_i)$    ▷ Adversarial example generation.
        $x_i(adv) = \arg\max_{x \in NN_{x_i}} S_C(x_i(perturbed), x)$    ▷ Nearest neighbor Search, $S_C$ is cosine similarity.
        $\widehat{y_i} = f_w(x_i(adv))$    ▷ Start weight perturbation (forward propagation).
        $l = L(y_i, \widehat{y_i})$    ▷ Loss computation.
        $\nabla_w(\theta, x_i, y_i) = \frac{\partial l}{\partial w}$    ▷ Layer-wise gradients computation.
        $\nabla_{w(projected)}(\theta, x_i, y_i) = \frac{\nabla_w(\theta, x_i, y_i)}{\|\nabla_w(\theta, x_i, y_i)\|_2} \|w\|_2$    ▷ Gradients projection.
        $w_{diff} = w_{original} - w_{perturbed}$    ▷ Computation of difference between original and perturbed weights.
        Train $f_{w_{perturbed}}$ on $x_i(adv)$ to get $w_{new}$
        $w = w_{new} + w_{diff}$    ▷ Remove weight perturbations of current batch after training on it.
    end for
end for
```

reference. Figure 3 illustrates the results of LSTM standard training, and Figures 4, 5, and 6 show the results of the three adversarial training scenarios. Both the weighted F1-score and the accuracy score alongside the binary cross-entropy loss value are displayed. During standard training, we monitor the weighted F1 score on the validation set and save the model's weights based on the best performance on this metric. For the AT case, we monitor an engineered metric that helps balance the model's performance on both types of data (clean and adversarial). This metric will be explained in more detail later on in this section.

4.2 Standard training & adversarial attack

Starting with the standard training process displayed in the three plots of Figure 3, we can see the consistent decrease in the loss value accompanied by an increase in the values of the weighted F1 and the accuracy scores on the training set. The model reaches its best performance on the

Algorithm 4. Perturb both inputs and weights and train on both clean and adversarial examples (scenario 3).

```
Input: Network $f_w$, training data $\{(x_i, y_i)\}_{i=1}^{n}$ where $x_i$ are the AraVec embeddings of
    the input tokens of each sequence, $y_i$ are the labels, $w$ model's original
    weights, nearest neighbors dictionary $NN$, learning rate $\eta$, batch size $m$,
    loss function $L$, number of epochs $T$.
Output: Robust network $f_w$.
for $t \leftarrow 0$ to $T-1$ do
    for $i \leftarrow 0$ to $n-1$ by $m$ (in parallel) do
        $w_{original} = w.copy()$    ▷ Copy original weights.
        $\widehat{y_i} = f_w(x_i)$    ▷ Start input perturbation (forward propagation).
        $l = L(y_i, \widehat{y_i})$    ▷ Loss computation.
        $\nabla_{x_i}(\theta, x_i, y_i) = \frac{\partial l}{\partial x_i}$    ▷ Gradients computation with respect to inputs.
        $\nabla_{x_i(projected)}(\theta, x_i, y_i) = \frac{\nabla_{x_i}(\theta, x_i, y_i)}{\|\nabla_{x_i}(\theta, x_i, y_i)\|_2}\|x_i\|_2$    ▷ Gradients projection.
        $x_i(perturbed) = x_i + \nabla_{x_i(projected)}(\theta, x_i, y_i)$    ▷ Adversarial example generation.
        $x_i(adv) = \arg\max_{x \in NN_{x_i}} S_C(x_i(perturbed), x)$    ▷ Nearest neighbor Search, $S_C$ is cosine similarity.
        $\widehat{y_i} = f_w(x_i(adv))$    ▷ Start weight perturbation (forward propagation).
        $l = L(y_i, \widehat{y_i})$    ▷ Loss computation.
        $\nabla_w(\theta, x_i, y_i) = \frac{\partial l}{\partial w}$    ▷ Layer-wise gradients computation.
        $\nabla_{w(projected)}(\theta, x_i, y_i) = \frac{\nabla_w(\theta, x_i, y_i)}{\|\nabla_w(\theta, x_i, y_i)\|_2}\|w\|_2$    ▷ Gradients projection.
        $w_{diff} = w_{original} - w_{perturbed}$    ▷ Computation of difference between original and perturbed weights.
        Train $f_{w_{perturbed}}$ on $x_i(adv)$ to get $w_{new}$
        $w = w_{new} + w_{diff}$    ▷ Remove weight perturbations of current batch after training on it.
        Train $f_w$ on $x_i$ to update $w$

    end for
end for
```

validation set at the eighth epoch where the loss value, the weighted F1-score, and the accuracy score on the validation and test splits are respectively: (0.414, 0.402), (0.812, 0.824), and (0.812, 0.824). After attacking this model, we see the huge drop in performance where the loss value of both the validation and test sets almost doubles to 0.874 and 0.830, respectively. Both the weighted F1-score and the accuracy score are also negatively affected as they drop on the validation and test sets, respectively, to (0.561, 0.576) and (0.561, 0.576). Hence, this indicates the success of establishing a proper AA method that can fool a model trained to solve an Arabic SA task.

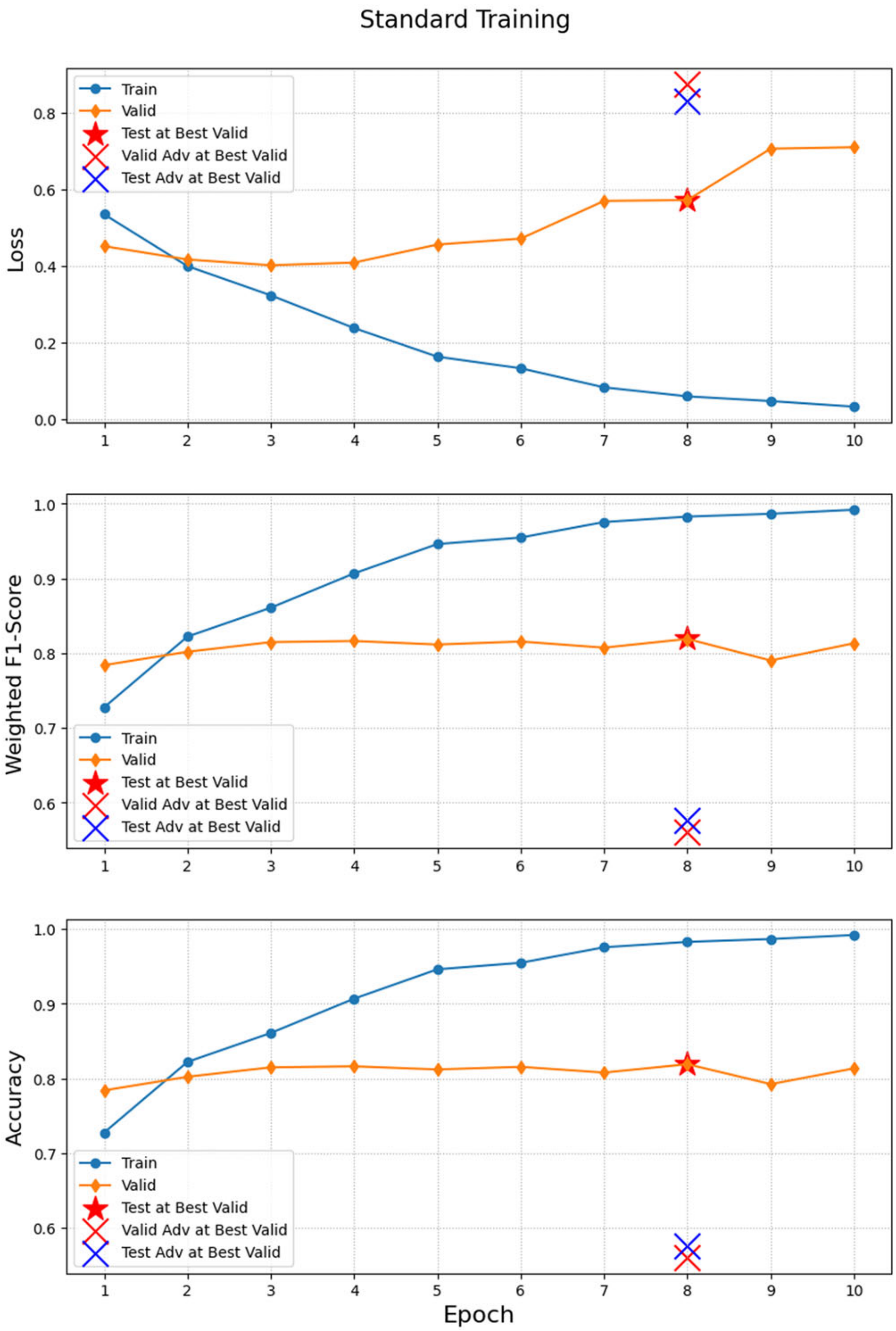

Figure 3. LSTM-based model standard training results. Top is the loss function value, middle is the weighted F1-score value, and bottom is the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

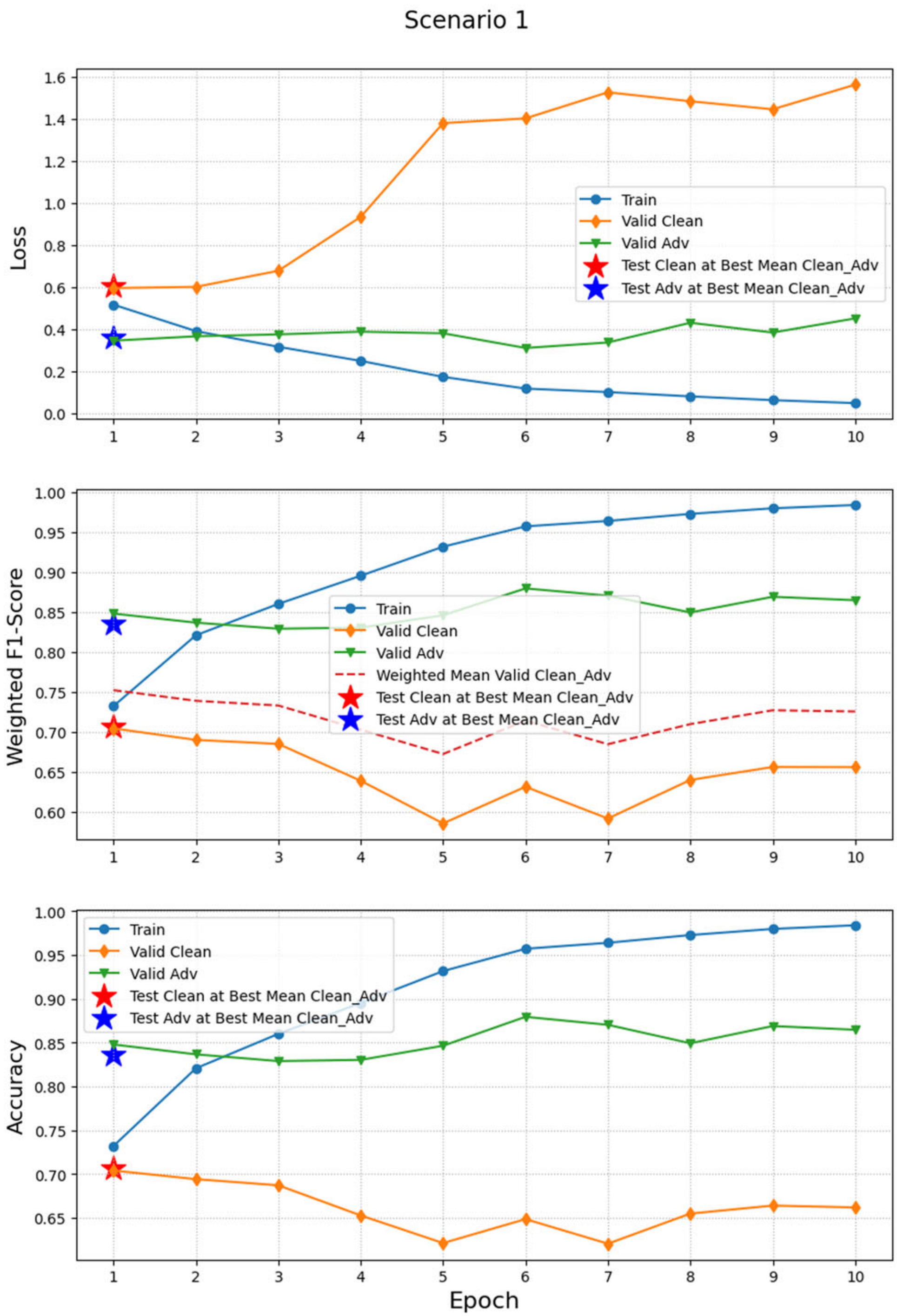

Figure 4. LSTM-based model scenario 1 results. Top is the loss function value, middle is the weighted F1-score value, and bottom is the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

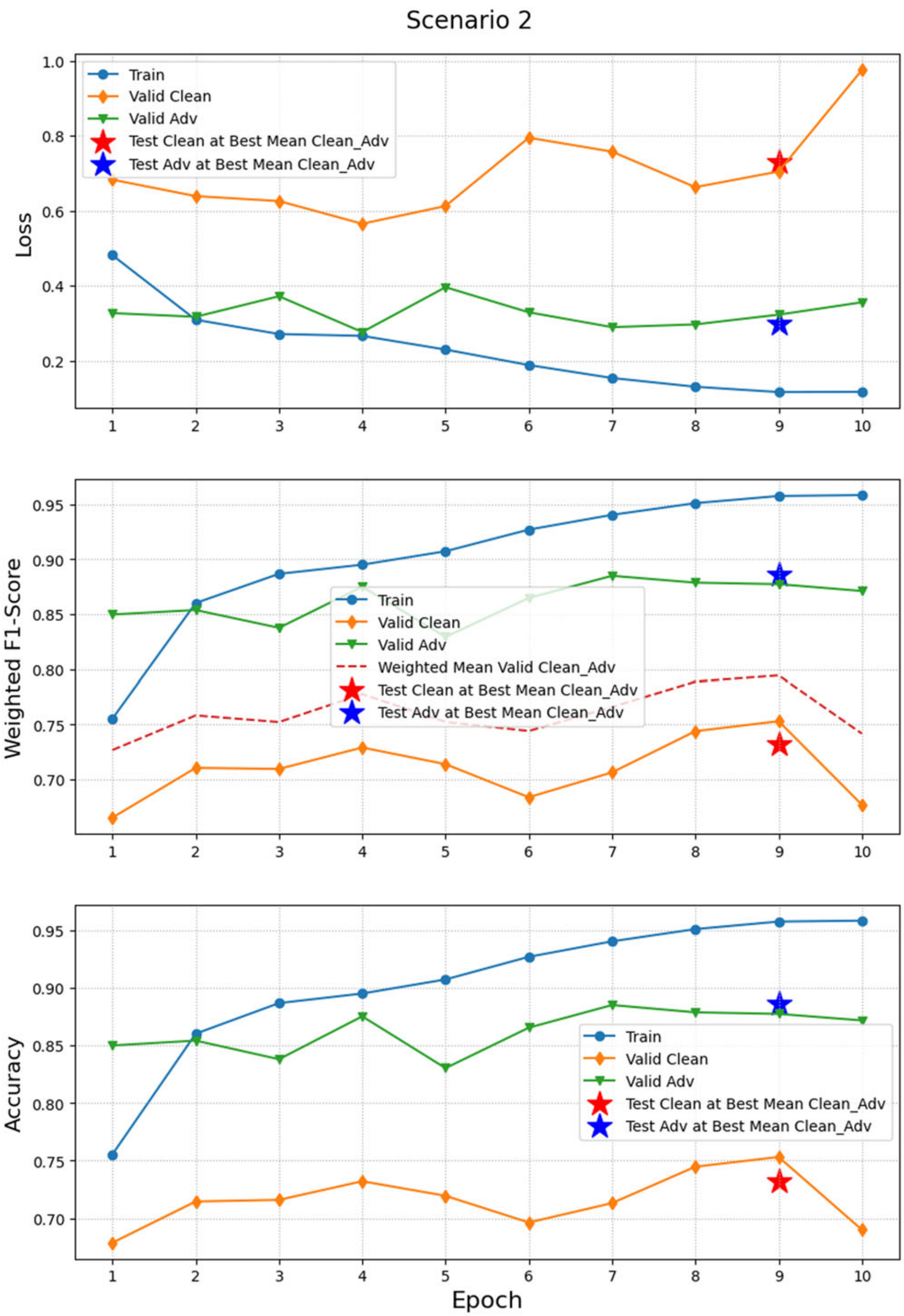

Figure 5. LSTM-based model scenario 2 results. Top is the loss function value, middle is the weighted F1-score value, and bottom is the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

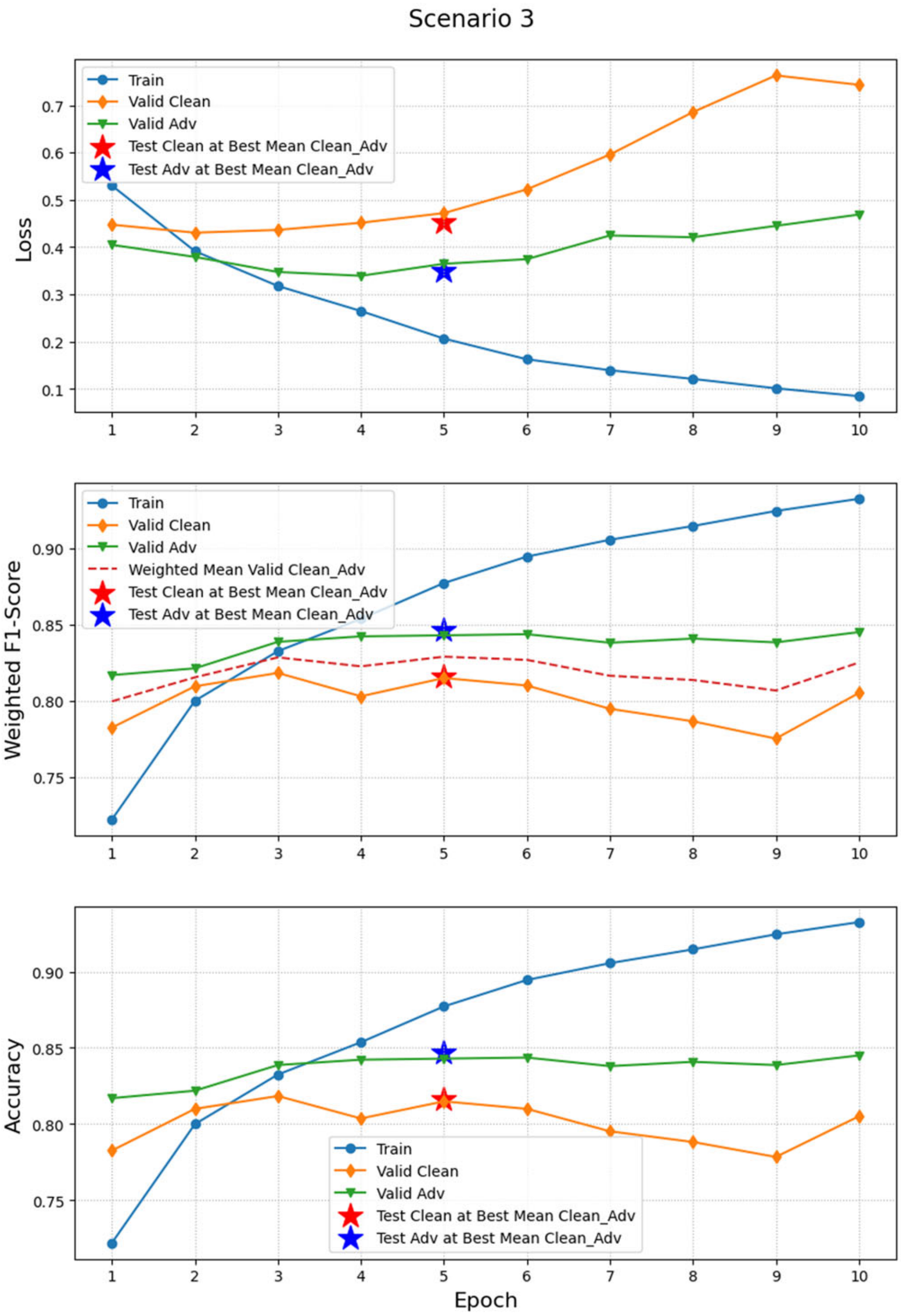

Figure 6. LSTM-based model scenario 3 results. Top is the loss function value, middle is the weighted F1-score value, and bottom is the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

4.3 Adversarial training

Next comes the AT step where the task is to build a *robust* model to AA. As explained in Section 3.7, we experiment with three different scenarios of AT. During the AT stage, we create a new metric which is the weighted mean of the weighted F1-score (WMWF1-score) on both the clean and adversarial examples of the validation set. The mathematical formulation of this metric is presented in equation (5). The weights are heuristically given to the two contributors based on experimentation. For the first and second scenarios, we give a clean-to-adversarial weight ratio of 1:2. This is because the model in these two modes is trained on the adversarial examples only and as the model progresses in training, it starts to overfit the adversarial examples at the expense of the clean ones. In this case, if we give full credit to the adversarial examples, the model will be significantly biased towards these examples, which would eventually lead to a significant drop in the model's performance on the clean data. We have experimented with other values for this ratio such as 2:1 for these two scenarios, however, we found the latter hinders the model from a significant boost in the performance on the adversarial data at the expense of a slight improvement on the clean one. For the third scenario, on the other hand, we reverse the values and give a clean-to-adversarial ratio of 2:1 as we note that the model starts early to have better performance on the adversarial data; meanwhile, it suffers on the clean data. Hence, our goal was to find a model that performs almost equally on both types of data, and the 2:1 ratio has proven to be working best for this purpose among other ratios.

$$WMWF1-score=\sum_{t=1}^{2} w_t\,\frac{1}{n}\sum_{c=0}^{C-1} n_c\frac{TP}{TP+\frac{1}{2}(FP+FN)} \tag{5}$$

where 2 is the number of input types (clean and adversarial), w_t is the chosen weight for input type t, C is the number of classes, n_c is the number of samples in class C, TP are the true positives, FP are the false positives, and FN are the false negatives.

4.3.1 Scenario 1

Regarding the first scenario, where the perturbations are applied to the inputs only and the model is trained on adversarial examples only, we see that the model fails to improve after the first epoch, based on the WMWF1-score; refer to Figure 4. Although there is some improvement in the performance on the adversarial data of the validation set, this improvement is accompanied by an overfitting to this data and a steep drop on the clean one. This behavior is not common among the other architectures as shown in the figures in Appendix One on page 46. Those models continue their improvement on the WMWF1-score until after the third epoch in the worst case, i.e. the case of the GRU-based model, Figure A1. However, five out of the seven models fail to improve on the clean validation data after the fifth epoch which, in turn, indicates the tendency of these models to overfit the adversarial data as well. This tendency to overfit the adversarial data has been mentioned in several research papers including Rice *et al.* (2020); Cai *et al.* (2018); Zhang *et al.* (2020b). In order to remedy the problem of overfitting the adversarial examples, we adopt saving the weights of the best-performing model on the WMWF1-score. This method, at its core, is similar to early stopping which has been proposed as a solution for this problem by Zhang *et al.* (2020b). However, in our work, we continue the training process until the tenth epoch, and during this period, the weights of the best-performing model on the custom metric are saved and then used for evaluation and comparison. For this AT mode, the best-performing model is the GRU-based model with a WMWF1-score of 0.780 where the weighted F1-scores on both the clean and the adversarial examples of the validation set are 0.732 and 0.875, respectively. The performance on the test set is comparable with weighted F1-scores of 0.745 and 0.867 on both the clean and the adversarial data, respectively.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

4.3.2 Scenario 2

For the second scenario, there is a noticeable improvement in the training process as the models get the chance to train longer before starting to overfit the adversarial samples. This behavior can be noticed through the fact that all the models need at least five epochs to converge, meanwhile, some models show their tendency to improve even after the tenth epoch such as in the case of the MHA-based model illustrated in Figure A2. Overall, all the models need one to eight additional epochs over scenario 1 to obtain the convergence weights. This, in turn, indicates the effectiveness of employing weight perturbations beside the input perturbations. Moreover, not only the training process was allowed to run for more epochs before overfitting, instead, the overall performance of the models also improved. For the case of the LSTM-based model, the weighted F1-scores on the clean and adversarial examples of the test set are 0.732 and 0.886, respectively, as compared to 0.706 and 0.836, respectively, in scenario 1; refer to Figure 5.

4.3.3 Scenario 3

Finally, for the third scenario where the perturbations are applied to both the inputs and the weights and the model is trained on both clean and adversarial data, we can see that the overall performance of the models has improved, especially on the clean data. Besides the improvement in performance, it is remarkable the faster convergence of the models to the optimal weights in most of the cases (5 out of 7 models) as compared to scenario 2. The LSTM-based model in this scenario has shown a high and comparable performance on both the clean and adversarial data with weighted F1-scores of 0.815 and 0.843, respectively, on the validation set, and 0.816 and 0.847, respectively, on the test set; refer to Figure 6.

4.4 General results

Tables 2 and 3 present the results of the standard and adversarial training modes, respectively. Based on the results of the standard training setting, we can see that the best-performing model on the clean test set is the LSTM-based model. However, the score on the adversarial data is much lower than the one on the clean data; 0.576 compared to 0.824, respectively, on both metrics. Moreover, we notice that the addition of a 1D CNN layer to the GRU and the MHA models helped the models perform better on the clean data, but had a negative impact on the adversarial one. The LSTM model, on the other hand, was negatively affected by the addition of the CNN layer on both data types. With regard to AT, and in general, it is noticeable the positive impact of the additional CNN layer to the LSTM and GRU models on the performance of these models on the clean data in the first two scenarios. On the contrary, the pure models outperformed the CNN-equipped models on the adversarial task which indicates their attendance to overfit the training data type and indicates the impact of the additional 1D CNN layer on the clean/adversarial generalization. Finally, for scenario 3, the best-performing model on the clean validation set is the LSTM-based model, meanwhile, the LSTM-CNN-based model outperformed it on the clean test set.

4.5 Remarks

Based on these results, there are several points that need to be discussed and clarified. The first point is that we notice some models perform better according to the value of the loss function; however, these models are outperformed by other models in terms of both the weighted F1-score and the accuracy metric. For instance, the best-performing model under standard training conditions with respect to the loss function is the LSTM model with a loss value of 0.414 on the clean validation set, refer to Table 2. Nevertheless, this model is left behind regarding the two metrics with weighted F1 and accuracy scores of 0.812 as compared to the GRU-based model which scores 0.815 and 0.816 on both metrics, respectively, in spite of its higher loss value of 0.589. This

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Table 2. Results of the seven models on the validation and test splits after standard training. Loss is the binary cross-entropy loss, F1 is the weighted F1-score, and Acc is the accuracy score. Both clean and adversarial versions for each split are used for evaluation

Split	Valid\Clean			Test\Clean			Valid\Adversarial			Test\Adversarial		
Metric	Loss	F1	Acc	Loss	F1	Acc	Loss	F1	Acc	Loss	F1	Acc
Model												
LSTM	**0.414**	0.812	0.812	**0.402**	**0.824**	**0.824**	0.874	0.561	0.561	0.830	0.576	0.576
GRU	0.589	0.815	0.816	0.570	0.819	0.819	1.340	0.596	0.597	1.236	0.606	0.606
CNN	0.566	0.786	0.787	0.546	0.788	0.789	0.919	0.607	0.609	0.934	0.596	0.599
MHA	0.547	0.783	0.783	0.565	0.777	0.777	0.800	**0.676**	**0.676**	0.814	**0.662**	**0.662**
LSTM-CNN	0.447	0.802	0.802	0.435	0.819	0.819	1.194	0.470	0.471	1.160	0.474	0.475
GRU-CNN	0.420	**0.817**	**0.818**	0.420	0.809	0.809	0.918	0.532	0.533	0.906	0.546	0.546
MHA-CNN	0.446	0.799	0.799	0.461	0.790	0.790	**0.780**	0.588	0.591	**0.791**	0.574	0.577

phenomenon can be attributed to the slight miscorrelation between the loss function and the evaluation metrics as these metrics depend on the principle of thresholding in their work. Standard thresholding in binary or multi-class classification tasks means mapping the output probability of the corresponding class to 1 if the probability is higher than 0.5; otherwise, it is mapped to 0. Some applications require different threshold values based on the point of interest in those applications; however, our study follows the standard case. The cross-entropy loss, on the other hand, rewards or punishes models based on their confidence in the predicted class label. For example, if a model predicts class 1 with probabilities of 1.0 and 0.51 for two different samples where the ground-truth label is 1, this will produce loss values of 0 and 0.673, respectively. However, the F1 and the accuracy scores in both cases will be 1 due to thresholding. Thus, the cross-entropy loss function is sensitive to the probability values; meanwhile, the evaluation metrics used in this study only care about the mapped predictions.

The second point that we need to pay attention to is that after applying adversarial training on the models, we notice that the models better predict the adversarial samples either on the validation or the test splits than it does on the clean ones. There are actually three potential hypotheses that can explain this phenomenon. First, applying perturbations to the input representations only is not guaranteed to generate the worst-case attack, that is because these gradients work best if and only if they are used to generate perturbations for the inputs in conjunction with perturbing the trainable weights of the model as these gradients are derived based on the chain rule. This chain of derivatives (gradients) must be taken as a whole in order to guarantee applying the worst-case attack. Hence, applying perturbations to the inputs only is just an optimistic approximation for the best attack. Second is the nearest neighbors search process which replaces the newly perturbed sample with its nearest neighbor from a predefined dictionary. This process forms a second approximation for the already approximated adversarial attack. Finally, the adversarial examples during training are continuously varying based on the updated trainable weights and the gradients backpropagated to the input representations. Thus, it is possible that the model gets trained on adversarial samples that are similar to the ones generated in the validation and test sets after applying the attack which ultimately means overfitting to the adversarial samples. Due to the dynamic nature of generating these adversarial samples during training, tracking these samples and comparing them to the ones generated in the validation and test sets is so complex that it needs to be studied in separate research, and hence we leave it for future work.

Third, we notice the outperformance of the LSTM/GRU-based models compared to MHA-based models in most scenarios. This can be attributed to the relatively small amount of

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Table 3. Results of the seven models on the validation and test splits after adversarial training. Loss is the binary cross-entropy loss, F1 is the weighted F1-score, and Acc is the accuracy score. Both clean and adversarial versions for each split are used for evaluation

	Split	Valid\Clean			Test\Clean			Valid\Adversarial			Test\Adversarial		
	Metric	Loss	F1	Acc	Loss	F1	Acc	Loss	F1	Acc	Loss	F1	Acc
Training Scenario	Model												
Scenario 1	LSTM	0.596	0.704	0.704	0.604	0.706	0.706	0.346	0.848	0.848	0.362	0.836	0.836
	GRU	0.538	0.732	0.732	0.539	0.745	0.745	**0.294**	**0.875**	**0.875**	**0.303**	**0.867**	**0.867**
	CNN	0.544	0.736	0.736	0.539	0.737	0.738	0.493	0.766	0.766	0.493	0.774	0.774
	MHA	0.563	0.724	0.724	0.550	0.725	0.725	0.519	0.747	0.747	0.507	0.755	0.755
	LSTM-CNN	0.611	0.739	0.740	0.582	0.749	0.750	0.505	0.783	0.783	0.492	0.796	0.796
	GRU-CNN	**0.495**	**0.759**	**0.759**	**0.497**	**0.759**	**0.759**	0.425	0.799	0.799	0.439	0.793	0.794
	MHA-CNN	0.586	0.721	0.721	0.563	0.737	0.737	0.491	0.775	0.775	0.500	0.771	0.771
Scenario 2	LSTM	0.704	0.753	0.753	0.730	0.732	0.732	**0.323**	**0.877**	**0.877**	**0.299**	**0.886**	**0.886**
	GRU	0.688	0.696	0.698	0.671	0.701	0.704	0.349	0.857	0.857	**0.299**	0.868	0.868
	CNN	0.531	0.730	0.731	0.531	0.749	0.749	0.481	0.755	0.755	0.468	0.770	0.771
	MHA	0.550	0.729	0.730	0.548	0.717	0.718	0.517	0.737	0.738	0.515	0.738	0.739
	LSTM-CNN	0.576	**0.773**	**0.773**	0.579	**0.760**	**0.761**	0.442	0.815	0.815	0.499	0.801	0.802
	GRU-CNN	0.528	0.754	0.755	**0.507**	0.757	0.757	0.446	0.795	0.796	0.442	0.797	0.798
	MHA-CNN	**0.525**	0.736	0.737	0.525	0.732	0.734	0.487	0.759	0.759	0.498	0.752	0.752
Scenario 3	LSTM	0.472	**0.815**	**0.815**	0.451	0.816	0.816	0.365	**0.843**	**0.843**	0.349	0.847	0.847
	GRU	0.440	0.811	0.811	**0.414**	0.817	0.817	**0.361**	0.842	0.842	**0.334**	**0.849**	**0.849**
	CNN	0.520	0.775	0.775	0.483	0.794	0.794	0.503	0.775	0.775	0.484	0.786	0.786
	MHA	0.501	0.778	0.778	0.497	0.776	0.776	0.502	0.776	0.776	0.503	0.770	0.770
	LSTM-CNN	**0.423**	0.813	0.813	0.420	**0.820**	**0.820**	0.409	0.814	0.814	0.418	0.805	0.805
	GRU-CNN	0.444	0.806	0.806	0.449	0.808	0.808	0.403	0.818	0.818	0.431	0.808	0.808
	MHA-CNN	0.459	0.781	0.783	0.465	0.771	0.773	0.406	0.813	0.814	0.415	0.797	0.798

data used for training the models. It has been mentioned in previous works the tendency of transformer-based models to overfit when less data is used for training which hinders these models' generalization (Zeyer *et al.* 2019). We believe that using a larger amount of data for training has the potential to yield better results for the transformer-based models which can be investigated in future works.

Fourth, we notice the drop in performance of the LSTM and GRU models when adding the 1D CNN layer. The working mechanism of the 1D CNN layer relies on mapping multiple inputs in a specific frame (kernel) into a single value by applying the convolution operation. This type of mapping enables the models to pay attention to multiple positions (token features) using one kernel, however, this kind of pooling might have been the reason for losing the signal of polarity in some sequences that negatively affected the performance. Further study is needed to be done in future work to further investigate this phenomenon.

Finally, to the best of our knowledge, this is the first work to employ AT to improve the robustness of Arabic NLP models. Works on other languages have shown worse results compared to

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

ours. Xu *et al.* (2022) for example, have employed PGD-AWP for AT. However, the accuracy of the model under attack did not exceed 57.60%.

5. Conclusion

In this work, we have designed a framework to attack trained models by means of synonym replacement. These synonyms are automatically selected from a predefined dictionary containing the 10 nearest neighbors to each token based on the AraVec pre-trained embeddings and the cosine similarity metric. After proving the effectiveness of this framework in attacking the trained models, we experimented with three different scenarios to reinforce these models and make them robust to such attacks. These scenarios revolved around training the models on perturbed samples and perturbed weights to enable the model to learn the weights necessary for making it robust. It was proven that the best scenario was the one that employed both clean and adversarial examples in training in conjunction with employing weight perturbation. To our knowledge, this is the first work in the Arabic NLP domain that aims to build a robust deep learning framework via employing AT.

For future work, we propose investigating the point mentioned in the Remarks Subsection where after adversarially training the models, these models become apparently better at predicting the adversarial examples as compared to the clean ones. We presented three hypotheses including the suspicion that the distribution of the adversarial examples in both the validation and test sets becomes closer to the adversarial examples in the training set. However, due to the continuously changing nature of these generated samples during training, which is attributed to the continuous updating of the models' weights that ultimately affects the gradients, tracking these samples in the training set requires separate research to study this phenomenon. On the other hand, this research was based on the LABR dataset which was collected from book readers' reviews. This sampling of reviews is actually a sampling of reviews generated by a more educated class of people who tend to use MSA in their writings. Hence, the effectiveness of the proposed framework can be more deeply evaluated if it is tested against dialectical contents.

Competing interests and funding. The authors of this manuscript declare that they have no competing interests with any party.

References

Abdul-Mageed M. and Diab M. (2012). Toward building a large-scale arabic sentiment lexicon. In *Proceedings of the 6th international global WordNet conference*, pp. 18–22.

Abdul-Mageed M. and Diab M. (2014). Sana: A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*.

Abdul-Mageed M., Zhang C., Elmadany A. and Ungar L. (2020). Toward micro-dialect identification in diaglossic and code-switched environments.arXiv preprint arXiv: 2010.

Abdullah M. and Shaikh S. (2018). Teamuncc at Semeval-2018 task 1: Emotion detection in English and Arabic Tweets using deep learning. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pp. 350–357.

Al Sallab A., Hajj H., Badaro G., Baly R., El-Hajj W. and Shaban K. (2015). Deep learning models for sentiment analysis in Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pp. 9–17.

Al-Azani S. and El-Alfy E.-S. (2018). Emojis-based sentiment classification of Arabic microblogs using deep recurrent neural networks. In *2018 international conference on computing sciences and engineering (ICCSE)*, IEEE, pp. 1–6.

Al-Azani S. and El-Alfy E.-S. M. (2017). Hybrid deep learning for sentiment polarity determination of arabic microblogs. In *International Conference on Neural Information Processing*, Springer, pp. 491–500.

Al-Dujaili A., Huang A., Hemberg E. and O'Reilly U.-M. (2018). Adversarial deep learning for robust detection of binary encoded malware. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, pp. 76–82.

Al-Sallab A., Baly R., Hajj H., Shaban K. B., El-Hajj W. and Badaro G. (2017). Aroma: A recursive deep learning model for opinion mining in Arabic as a low resource language. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* **16**(4), 1–20.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Al-Smadi M., Al-Ayyoub M., Jararweh Y. and Qawasmeh O. (2019a). Enhancing aspect-based sentiment analysis of Arabic hotels' reviews using morphological, syntactic and semantic features. *Information Processing & Management* **56**(2), 308–319.

Al-Smadi M., Talafha B., Al-Ayyoub M. and Jararweh Y. (2019b). Using long short-term memory deep neural networks for aspect-based sentiment analysis of Arabic reviews. *International Journal of Machine Learning and Cybernetics* **10**(8), 2163–2175.

Alayba A. M., Palade V., England M. and Iqbal R. (2017). Arabic language sentiment analysis on health services. In *2017 1st International Workshop on Arabic Script Analysis and Recognition (asar)*. IEEE, pp. 114–118.

Alayba A. M., Palade V., England M. and Iqbal R. (2018a). A combined CNN and LSTM model for Arabic sentiment analysis. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, pp. 179–191.

Alayba A. M., Palade V., England M. and Iqbal R. (2018b). Improving sentiment analysis in Arabic using word representation. In *2018 IEEE 2nd International Workshop On Arabic and Derived Script Analysis and Recognition (ASAR)*. IEEE, pp. 13–18.

Aly M. and Atiya A. (2013). LABR: A large scale arabic book reviews dataset. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 494–498.

Antoun W., Baly F. and Hajj H. (2020). Arabert: Transformer-based model for Arabic language understanding, arXiv preprint arXiv: 2003.00104.

Badaro G., Baly R., Hajj H., El-Hajj W., Shaban K. B., Habash N., Al-Sallab A. and Hamdi A. (2019). A survey of opinion mining in Arabic: A comprehensive system perspective covering challenges and advances in tools, resources, models, applications, and visualizations. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* **18**(3), 1–52.

Badaro G., Baly R., Hajj H., Habash N. and El-Hajj W. (2014). A large scale Arabic sentiment lexicon for Arabic opinion mining. In *Proceedings of the EMNLP. 2014 workshop on arabic natural language processing (ANLP)*, pp. 165–173.

Badaro G., El Jundi O., Khaddaj A., Maarouf A., Kain R., Hajj H. and El-Hajj W. (2018). EMA at SemEval-2018 task 1: Emotion mining for Arabic. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pp. 236–244.

Baly R., Hajj H., Habash N., Shaban K. B. and El-Hajj W. (2017). A sentiment treebank and morphologically enriched recursive deep models for effective sentiment analysis in Arabic. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)* **16**(4), 1–21.

Baly R., Hobeica R., Hajj H., El-Hajj W., Shaban K. B. and Al-Sallab A. (2016). A meta-framework for modeling the human reading process in sentiment analysis. *ACM Transactions on Information Systems (TOIS)* **35**(1), 1–21.

Barhoumi A., Estève Y., Aloulou C. and Belguith L. (2017). Document embeddings for Arabic sentiment analysis. In *Conference on Language Processing and Knowledge Management, LPKM 2017*.

Belinkov Y. and Bisk Y. (2017). Synthetic and natural noise both break neural machine translation, arXiv preprint arXiv: 1711.02173.

Bird S., Klein E. and Loper E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.".

Blohm M., Jagfeld G., Sood E., Yu X. and Vu N. T. (2018). Comparing attention-based convolutional and recurrent neural networks: Success and limitations in machine reading comprehension.arXiv preprint arXiv: 1808.08744.

Bose R., Dey R. K., Roy S. and Sarddar D. (2020). Sentiment analysis on online product reviews. In *Information and Communication Technology for Sustainable Development*. Springer, pp. 559–569.

Cai Q.-Z., Du M., Liu C. and Song D. (2018). Curriculum adversarial training.arXiv preprint arXiv: 1805.04807.

Chen X., Sun Y., Athiwaratkun B., Cardie C. and Weinberger K. (2018). Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics* **6**, 557–570.

Cheng Y., Jiang L. and Macherey W. (2019). Robust neural machine translation with doubly adversarial inputs.arXiv preprint arXiv: 1906.

Chung J., Gulcehre C., Cho K. and Bengio Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv: 1412.3555.

Clark E. M., James T., Jones C. A., Alapati A., Ukandu P., Danforth C. M. and Dodds P. S. (2018). A sentiment analysis of breast cancer treatment experiences and healthcare perceptions across Twitter. arXiv preprint arXiv: 1805.09959.

Conneau A., Khandelwal K., Goyal N., Chaudhary V., Wenzek G., Guzmán F., Grave E., Ott M., Zettlemoyer L. and Stoyanov V. (2019). Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv: 1911.

Dahou A., Xiong S., Zhou J., Haddoud M. H. and Duan P. (2016). Word embeddings and convolutional neural network for Arabic sentiment classification. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, pp. 2418–2427.

Darwish K., Habash N., Abbas M., Al-Khalifa H., Al-Natsheh H. T., Bouamor H., Bouzoubaa K., Cavalli-Sforza V., El-Beltagy S. R., El-Hajj W. et al. (2021). A panoramic survey of natural language processing in the Arab world. *Communications of the ACM* **64**(4), 72–81.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Duwairi R. and Abushaqra F. (2021). Syntactic-and morphology-based text augmentation framework for Arabic sentiment analysis. *PeerJ Computer Science* **7**, e469.

Ebrahimi J., Rao A., Lowd D. and Dou D. (2017). Hotflip: White-box adversarial examples for text classification. arXiv preprint arXiv: 1712.06751.

El-Beltagy S. R. (2016). Nileulex: A phrase and word level sentiment Lexicon for Egyptian and modern standard Arabic. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 2900–2905.

El-Beltagy S. R. and Ali A. (2013). Open issues in the sentiment analysis of arabic social media: A case study. In *2013 9th International Conference on Innovations in Information Technology (IIT)*, IEEE, pp. 215–220.

El-Beltagy S. R., Kalamawy M. E. and Soliman A. B. (2017). Niletmrg at semeval-2017 task 4: Arabic sentiment analysis.arXiv preprint arXiv: 1710.08458.

El-Beltagy S. R., Khalil T., Halaby A. and Hammad M. (2016). Combining lexical features and a supervised learning approach for Arabic sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, pp. 307–319.

ElSahar H. and El-Beltagy S. R. (2015). Building large Arabic multi-domain resources for sentiment analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, pp. 23–34.

Eskander R. and Rambow O. (2015). SLSA: A sentiment lexicon for standard Arabic. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2545–2550.

Farha I. A. and Magdy W. (2019). Mazajak: An online arabic sentiment analyser. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pp. 192–198.

Ganin Y. and Lempitsky V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, PMLR, pp. 1180–1189.

Ganin Y., Ustinova E., Ajakan H., Germain P., Larochelle H., Laviolette F., Marchand M. and Lempitsky V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* **17**(1), 2096–2030.

Gao J., Lanchantin J., Soffa M. L. and Qi Y. (2018). Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, pp. 50–56.

González J.-A., Pla F. and Hurtado L.-F. (2017). ELiRF-UPV at SemEval-2017 task 4: Sentiment analysis using deep learning. In *Proceedings of the 11th international workshop on semantic evaluation (SEMEVAL-2017)*, pp. 723–727.

Goodfellow I. J., Shlens J. and Szegedy C. (2014). Explaining and harnessing adversarial examples, arXiv preprint arXiv: 1412.6572.

Goyal H., Singh A. and Kumar P. (2021). Paw at semeval-2021 task 2: Multilingual and cross-lingual word-in-context disambiguation: Exploring cross lingual transfer, augmentations and adversarial training. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pp. 743–747.

Gridach M., Haddad H. and Mulki H. (2017). Empirical evaluation of word representations on arabic sentiment analysis. In *International Conference on Arabic Language Processing*, Springer, pp. 147–158.

Gupta V. (2021). Multilingual and multilabel emotion recognition using virtual adversarial training.arXiv preprint arXiv: 2111.06181.

Han K., Xiao A., Wu E., Guo J., Xu C. and Wang Y. (2021). Transformer in transformer. arXiv preprint arXiv:210300112.

Han T., Liu C., Yang W. and Jiang D. (2019). A novel adversarial learning framework in deep convolutional neural network for intelligent diagnosis of mechanical faults. *Knowledge-Based Systems* **165**, 474–487.

Hochreiter S. and Schmidhuber J. (1997). Long short-term memory. *Neural Computation* **9**(8), 1735–1780.

Huang R., Xu B., Schuurmans D. and Szepesvári C. (2015). Learning with a strong adversary. arXiv preprint arXiv: 1511.03034.

Iyyer M., Wieting J., Gimpel K. and Zettlemoyer L. (2018). Adversarial example generation with syntactically controlled paraphrase networks.arXiv preprint arXiv: 1804.06059.

Jia R. and Liang P. (2017). Adversarial examples for evaluating reading comprehension systems.arXiv preprint arXiv: 1707.07328.

Joty S., Nakov P., Màrquez L. and Jaradat I. (2017). Cross-language learning with adversarial neural networks: Application to community question answering. arXiv preprint arXiv: 1706.06749.

Kalchbrenner N., Grefenstette E. and Blunsom P. (2014). A convolutional neural network for modelling sentences.arXiv preprint arXiv: 1404.2188.

Khalifa S., Habash N., Abdulrahim D. and Hassan S. (2016). A large scale corpus of Gulf Arabic. arXiv preprint arXiv: 1609.02960.

Khalil T., Halaby A., Hammad M. and El-Beltagy S. R. (2015). Which configuration works best? An experimental study on supervised Arabic Twitter sentiment analysis. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, IEEE, pp. 86–93.

Kullback S. and Leibler R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics* **22**(1), 79–86.

Kumar A., Irsoy O., Ondruska P., Iyyer M., Bradbury J., Gulrajani I., Zhong V., Paulus R. and Socher R. (2016). Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, PMLR, pp. 1378–1387.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Kurakin A., Goodfellow I. and Bengio S. (2016). Adversarial machine learning at scale. arXiv preprint arXiv: 1611.01236.

Le Q. and Mikolov T. (2014). Distributed representations of sentences and documents. In *International Conference on Machine Learning*, PMLR, pp. 1188–1196.

LeCun Y., Bottou L., Bengio Y. and Haffner P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324.

Liang B., Li H., Su M., Bian P., Li X. and Shi W. (2017). Deep text classification can be fooled. arXiv preprint arXiv: 1704.08006.

Loshchilov I. and Hutter F. (2017). LM-CMA: An alternative to L-BFGS for large-scale black box optimization.. *Evolutionary Computation* **25**(1), 143–171, Decoupled weight decay regularization. arXiv preprint arXiv: 1711.05101.

Mackey T. K., Miner A. and Cuomo R. E. (2015). Exploring the e-cigarette e-commerce marketplace: Identifying internet e-cigarette marketing characteristics and regulatory gaps. *Drug and Alcohol Dependence* **156**, 97–103.

Madry A., Makelov A., Schmidt L., Tsipras D. and Vladu A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv: 1706.06083.

Maini P., Wong E. and Kolter Z. (2020). Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, PMLR, pp. 6640–6650.

Martelli F., Kalach N., Tola G. and Navigli R. (2021). SemEval-2021 task 2: Multilingual and cross-lingual word-in-context disambiguation (MCLl-WIC). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pp. 24–36.

Mikolov T., Chen K., Corrado G. and Dean J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv: 1301.3781.

Miyato T., Dai A. M. and Goodfellow I. (2016). Adversarial training methods for semi-supervised text classification. arXiv preprint arXiv: 1605.07725.

Miyato T., Maeda S.-i., Koyama M. and Ishii S. (2018). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **41**(8), 1979–1993.

Mohammad S., Bravo-Marquez F., Salameh M. and Kiritchenko S. (2018). Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pp. 1–17.

Mourad A. and Darwish K. (2013). Subjectivity and sentiment analysis of modern standard Arabic and Arabic microblogs. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 55–64.

Niu T. and Bansal M. (2018). Adversarial over-sensitivity and over-stability strategies for dialogue models.arXiv preprint arXiv: 1809.02079.

Pang T., Yang X., Dong Y., Su H. and Zhu J. (2020). Bag of tricks for adversarial training. arXiv preprint arXiv: 2010.00467.

Papernot N., McDaniel P., Goodfellow I., Jha S., Celik Z. B. and Swami A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp. 506–519.

Papernot N., McDaniel P., Swami A. and Harang R. (2016). Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, IEEE, pp. 49–54.

Pasha A., Al-Badrashiny M., Diab M., El Kholy A., Eskander R., Habash N., Pooleery M., Rambow O. and Roth R. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 1094–1101.

Qin H., Chen G., Tian Y. and Song Y. (2021). Improving Arabic diacritization with regularized decoding and adversarial training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 534–542.

Refaee E. and Rieser V. (2014). Subjectivity and sentiment analysis of Arabic Twitter feeds with limited resources. In *Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools Workshop Programme*, vol. **16.**

Ren S., Deng Y., He K. and Che W. (2019). Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1085–1097.

Rice L., Wong E. and Kolter Z. (2020). Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, PMLR, pp. 8093–8104.

Rosenberg I., Shabtai A., Rokach L. and Elovici Y. (2018). Generic black-box end-to-end attack against state of the art API call based malware classifiers. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, Springer, pp. 490–510.

Rosenthal S., Farra N. and Nakov P. (2019). Semeval-2017 task 4: Sentiment analysis in Twitter. arXiv preprint arXiv: 1912.

Salameh M., Mohammad S. and Kiritchenko S. (2015). Sentiment after translation: A case-study on Arabic social media posts. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies*, pp. 767–777.

Samanta S. and Mehta S. (2018). Generating adversarial text samples. In *European Conference on Information Retrieval*, Springer, pp. 744–749.

Schott L., Rauber J., Bethge M. and Brendel W. (2018). Towards the first adversarially robust neural network model on MNIST. arXiv preprint arXiv: 1805.09190.

Shaham U., Yamada Y. and Negahban S. (2018). Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* **307**, 195–204.

Shoukry A. and Rafea A. (2015). A hybrid approach for sentiment classification of Egyptian dialect Tweets. In *2015 First International Conference on Arabic Computational Linguistics (ACLing)*, IEEE, pp. 78–85.

Socher R., Perelygin A., Wu J., Chuang J., Manning C. D., Ng A. Y. and Potts C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642.

Soliman A. B., Eissa K. and El-Beltagy S. R. (2017). Aravec: A set of arabic word embedding models for use in Arabic NLP. *Procedia Computer Science* **117**, 256–265.

Sun M., Tang F., Yi J., Wang F. and Zhou J. (2018). Identify susceptible locations in medical records via adversarial attacks on deep predictive models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–801.

Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I. and Fergus R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv: 1312.6199.

Tang D., Qin B. and Liu T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1422–1432.

Tramèr F., Kurakin A., Papernot N., Goodfellow I., Boneh D. and McDaniel P. (2017). Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv: 1705.07204.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł. and Polosukhin I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems* **30**.

Wang X., Yang Y., Deng Y. and He K. (2021). Adversarial training with fast gradient projection method against synonym substitution based text attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**, pp. 13997–14005.

Wang Y. and Bansal M. (2018). Robust machine comprehension models via adversarial training.arXiv preprint arXiv: 1804.06473.

Wang Y., Zou D., Yi J., Bailey J., Ma X. and Gu Q. (2019). Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations.*

Wankhade M., Rao A. C. S. and Kulkarni C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review* **55**(7), 1–50.

Wu D., Xia S.-T. and Wang Y. (2020). Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems* **33**, 2958–2969.

Xing F. Z., Cambria E. and Welsch R. E. (2018). Natural language based financial forecasting: A survey. *Artificial Intelligence Review* **50**(1), 49–73.

Xu J., Li L., Zhang J., Zheng X., Chang K.-W., Hsieh C.-J. and Huang X.-J. (2022). Weight perturbation as defense against adversarial word substitutions. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 7054–7063.

Zaghouani W., Mohit B., Habash N., Obeid O., Tomeh N., Rozovskaya A., Farra N., Alkuhlani S. and Oflazer K. (2014). Large scale Arabic error annotation: Guidelines and framework.

Zalmout N. and Habash N. (2019). Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling.arXiv preprint arXiv: 1910.

Zeyer A., Bahar P., Irie K., Schlüter R. and Ney H. (2019). A comparison of transformer and LSTM encoder decoder models for ASR. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 8–15.

Zhang H. and Xu W. (2019). Adversarial interpolation training: A simple approach for improving model robustness.

Zhang H., Yu Y., Jiao J., Xing E., El Ghaoui L. and Jordan M. (2019). Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, PMLR, pp. 7472–7482.

Zhang H., Zhou H., Miao N. and Li L. (2020a). Generating fluent adversarial examples for natural languages. arXiv preprint arXiv: 2007.06174.

Zhang J., Xu X., Han B., Niu G., Cui L., Sugiyama M. and Kankanhalli M. (2020b). Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, PMLR, pp. 11278–11287.

Zhang W. E., Sheng Q. Z., Alhazmi A. and Li C. (2020c). Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions On Intelligent Systems and Technology (TIST)* **11**(3), 1–41.

Zhao Z., Dua D. and Singh S. (2017). Generating natural adversarial examples. arXiv preprint arXiv: 1710.11342.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

Appendices

A. Remaining models’ progress plots

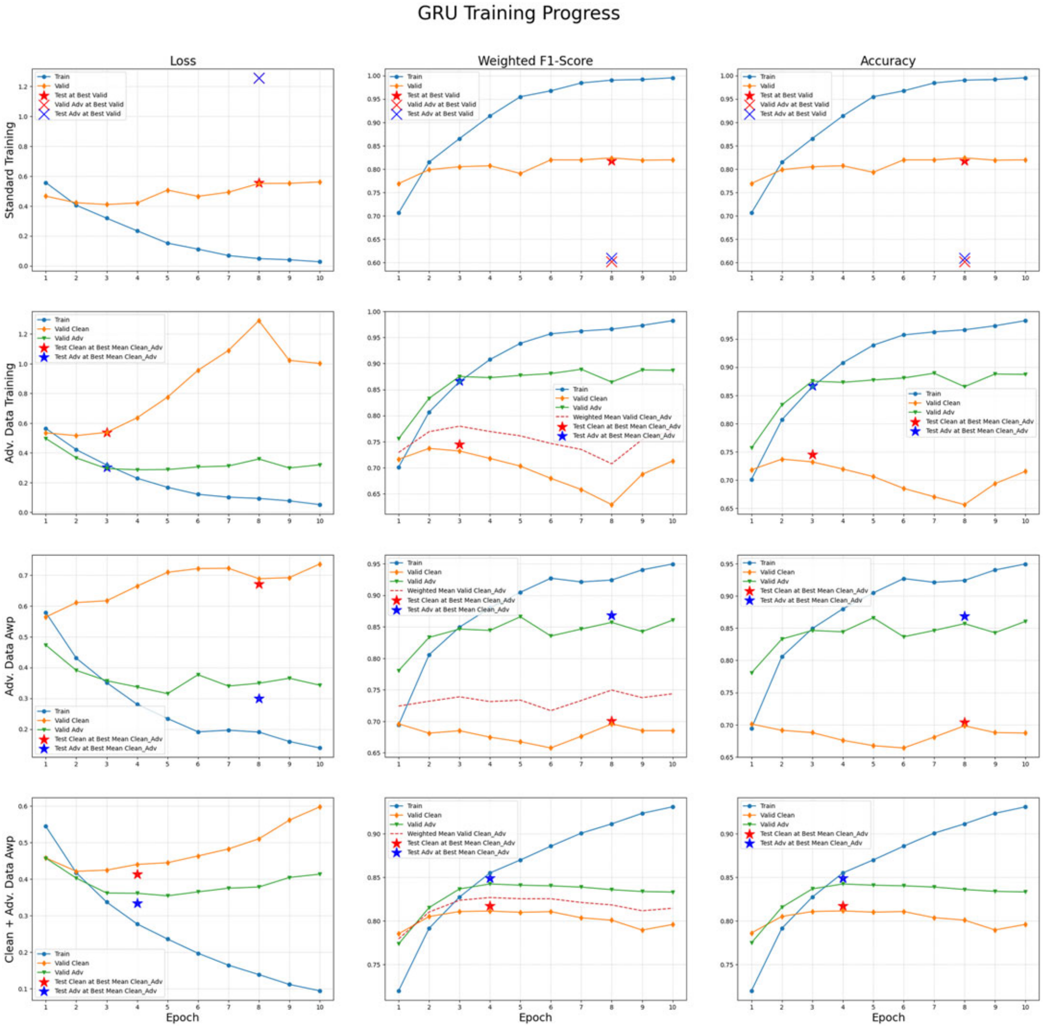

Figure A1. GRU-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

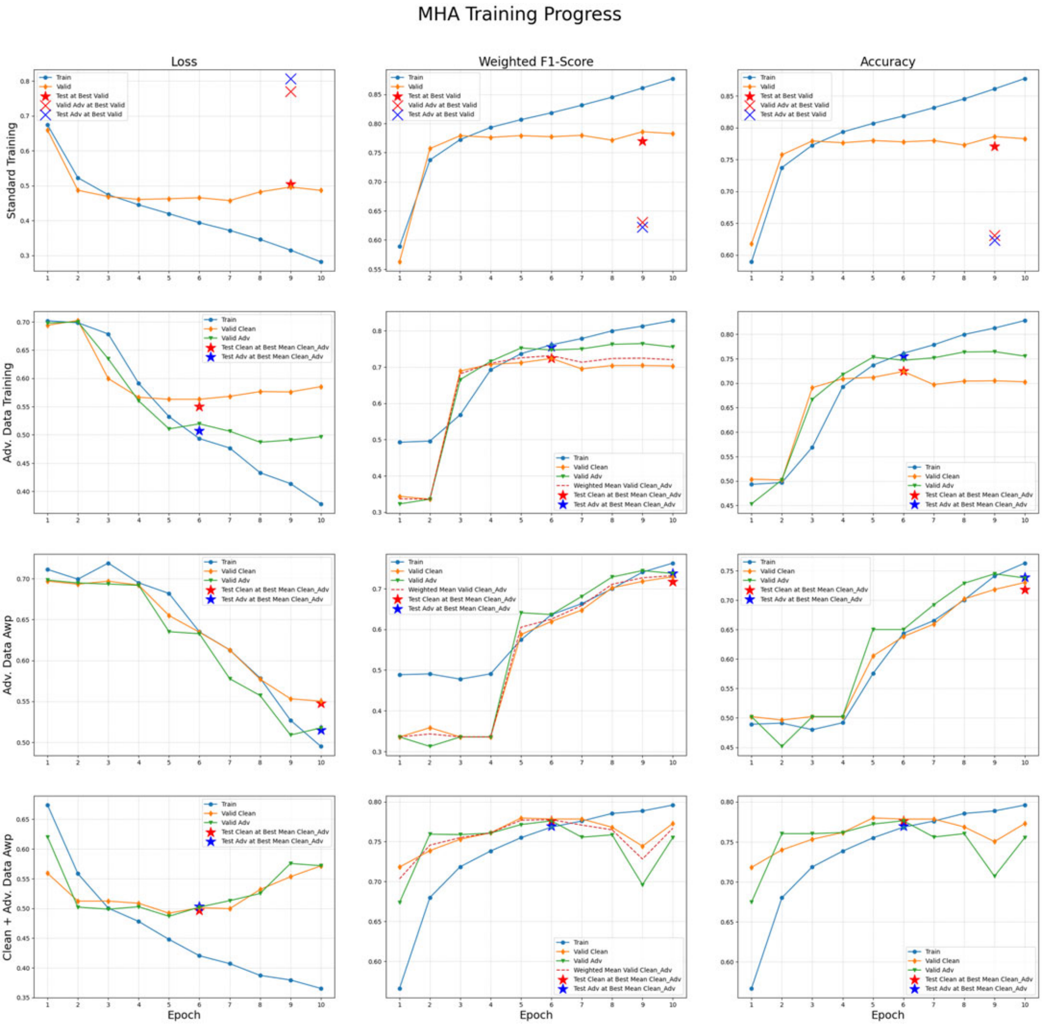

Figure A2. MHA-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

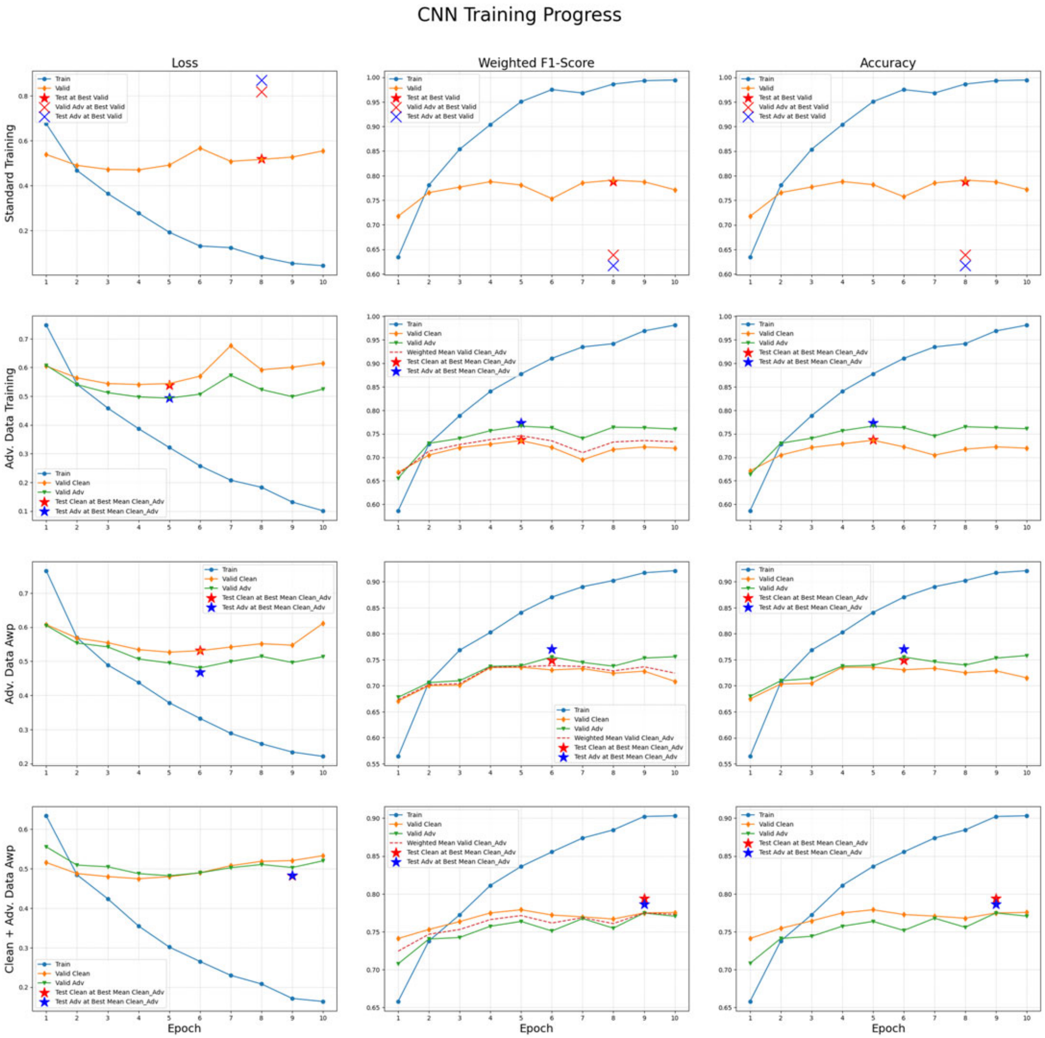

Figure A3. CNN-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

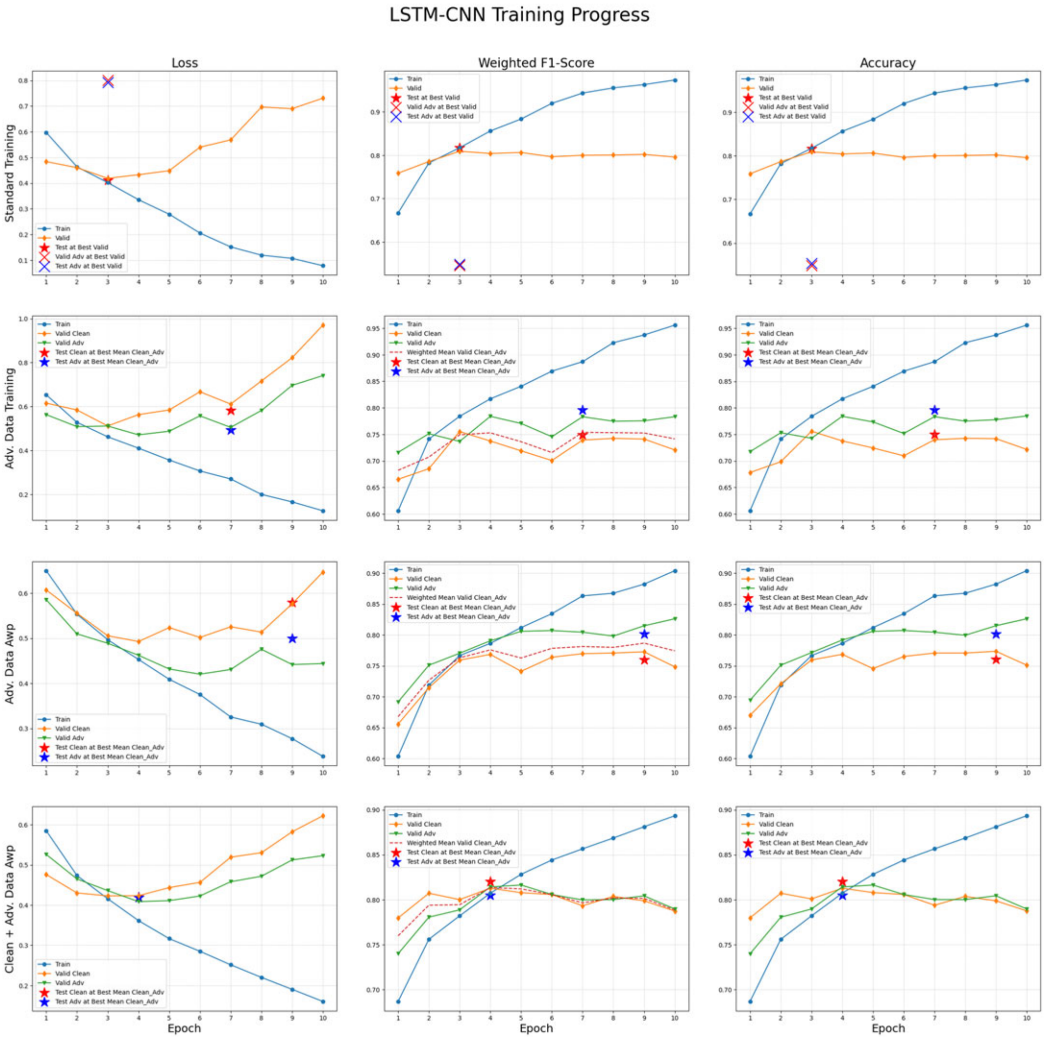

Figure A4. LSTM-CNN-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

https://doi.org/10.1017/nlp.2024.35 Published online by Cambridge University Press

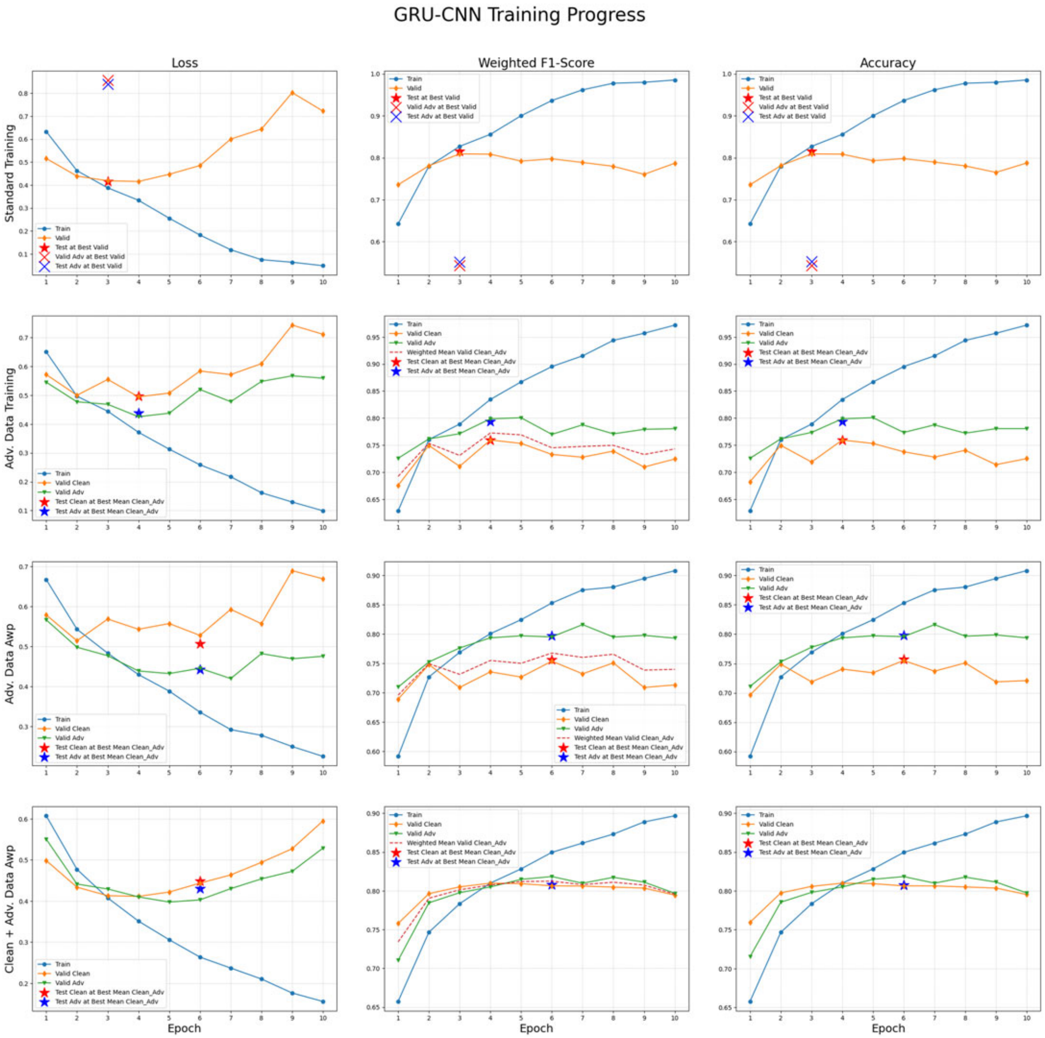

Figure A5. GRU-CNN-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

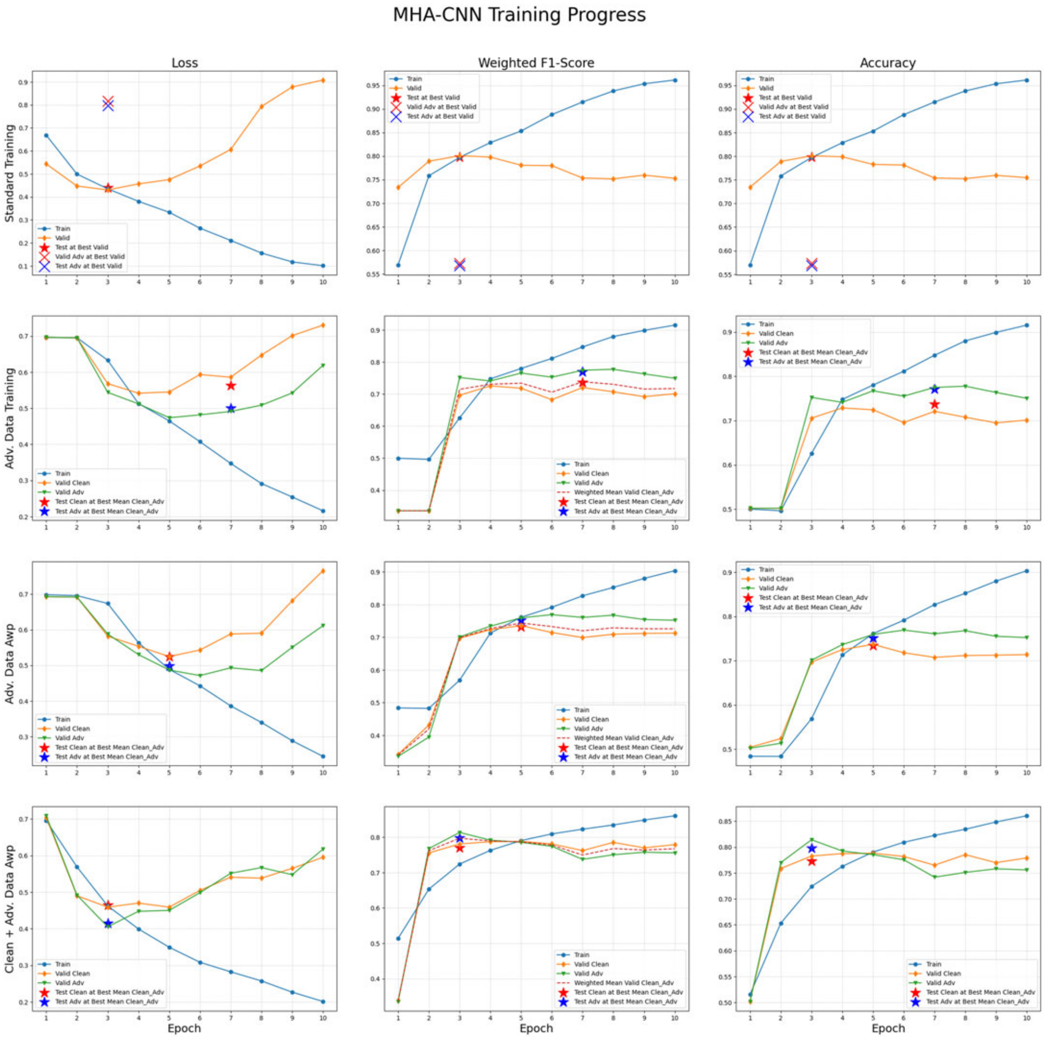

Figure A6. MHA-CNN-based model results. The first row shows the results of the standard training process. The second row shows the results of training on adversarial examples only (scenario 1). The third row shows the results of training on perturbed examples and weights (scenario 2). The fourth row shows the results of training on both clean and perturbed examples where the weights are also perturbed (scenario 3)). The left column shows the loss function value. The second column shows the weighted F1-score value. The third column shows the accuracy value.

Cite this article: Radman A and Duwairi R. Towards a robust deep learning framework for Arabic sentiment analysis. *Natural Language Processing* https://doi.org/10.1017/nlp.2024.35