Natural Language Engineering (2024), **30**, pp. 1130–1154
doi:10.1017/S1351324923000475

ARTICLE

Neural Arabic singular-to-plural conversion using a pretrained Character-BERT and a fused transformer

Azzam Radman, Mohammed Atros and Rehab Duwairi

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

(Received 16 August 2022; revised 13 September 2023; accepted 13 September 2023; first published online 11 October 2023)

Abstract
Morphological re-inflection generation is one of the most challenging tasks in the natural language processing (NLP) domain, especially with morphologically rich, low-resource languages like Arabic. In this research, we investigate the ability of transformer-based models in the singular-to-plural Arabic noun conversion task. We start with pretraining a Character-BERT model on a masked language modeling task using 1,134,950 Arabic words and then adopting the fusion technique to transfer the knowledge gained by the pretrained model to a full encoder–decoder transformer model, in one of the proposed settings. The second proposed setting directly fuses the output Character-BERT embeddings into the decoder. We then analyze and compare the performance of the two architectures and provide an interpretability section in which we track the features of attention with respect to the model. We perform the interpretation on both the macro and micro levels, providing some individual examples. Moreover, we provide a thorough error analysis showing the strengths and weaknesses of the proposed framework. To the best of our knowledge, this is the first effort in the Arabic NLP domain that adopts the development of an end-to-end fused-transformer deep learning model to address the problem of singular-to-plural conversion.

Keywords: Morphological inflection generation; singular-to-plural conversion; Arabic NLP; fusion transformer; Character-BERT

1. Introduction

Morphological inflection generation (MIG) is the task of automatically extracting an inflected form (e.g., stung) from a source lemma (e.g., sting) given some morpho-syntactic illustration for the source lemma (e.g., verb) and the targeted inflection form (e.g., past). From this general definition stems a variety of natural language processing (NLP) tasks depending on the types of source lemma and targeted inflection form like extracting plural form of singular words, gerund form of verbs, and past tense of verbs.

The process of generating an inflected form can be trivial or nontrivial depending on the inflected form being regular or irregular (Leminen *et al.* 2019). Regular inflected forms can be found by pasting together the source lemma with the suitable suffix or prefix of re-inflection (e.g., book + s). Whereas, irregular inflected forms cannot be found by this simple concatenation (e.g., woman and women).

Languages could be described as being morphologically rich or poor. In morphologically rich languages, such as Arabic, the surface form of a word has more inflections (compared to morphologically poor languages) to account for syntactic and semantic properties. Languages could also be described as being low-resource languages (LRLs) or high-resource languages (HRLs). LRLs can be described as less computerized or less studied languages (Singh 2008).

Morphologically rich languages impose challenges on NLP models due to the problem of data sparsity attributed to source lemmas having multiple inflected forms featured with low frequencies in corpora. This problem is amplified for the case of LRLs leading to most of the NLP research being limited to only 20 languages (Magueresse, Carles, and Heetderks 2020). A motivation for developing well-performing MIG models would be to enrich the literature with morphological inflection tables to assist models suffering from data sparsity problems. There are various NLP tasks that could benefit from the virtues of MIG modeling including keyword spotting (Narasimhan *et al.* 2014) and machine translation (MT) (Bojar *et al.* 2017).

First NLP models for MIG were rule-based (Koskenniemi *et al.* 1983). These models achieve remarkable results since they are built using well-crafted linguistic rules which take into account both regular and irregular inflected forms. However, their dependency on linguistic expertise forms a barrier in the face of their high adoption. Later, machine learning models were employed for the task of MIG (Dreyer and Eisner 2011) and Nicolai, Cherry, and Kondrak (2015).

Following their success in many NLP tasks, recurrent neural network (RNN)-based sequence-to-sequence models have replaced machine learning models for morphological inflection transduction. The literature is rich with such models that have achieved impressive results for this task such as Kann and Schütze (2016) and Schütze *et al.* (2016). Currently, transformers present themselves as state-of-the-art models for most NLP tasks. In their work, Wu *et al.* (2020) showed that increasing the batch size of the vanilla transformer was enough to outperform recurrent models on the tasks of MIG and grapheme-to-phoneme conversion.

Arabic, a morphologically rich and low-resource language, is under-researched from the perspective of NLP in general and scarcely researched from the perspective of MIG. Recently, SIGMORPHON[a] shared tasks included some Arabic datasets and challenged participants to develop MIG models for the Arabic language. Motivated by the scarcity of Arabic MIG research and SIGMORPHON 2022 shared task 0 part 2 (Automatic Morphological Acquisition Trajectories), we intend to contribute to the Arabic MIG research. This task is inspired by children's unsupervised ability to learn and generalize their morphological knowledge. Subtask 2 of this challenge urges participants to automatically generate plural forms of Arabic singular forms given some morpho-syntactic features (human/not human, feminine/masculine). In this study, we approach this task by pretraining a CBERT on a masked language modeling (MLM) task and then fusing this model into a full transformer encoder–decoder model. To our knowledge, this the first effort in Arabic that aims at addressing the singular-to-plural conversion problem via transformer-based techniques.

It is worth remarking that there are three forms of plural nouns in the Arabic language: masculine, feminine, and irregular. The first two of which are the easier to predict in terms of the morphological structure as they only need two additional suffixes to the end of the word to convert the singular form into plural, namely "ين"–"yn" or "ون"–"wn" and "ات"–"At" for the masculine, and feminine nouns, respectively. However, for the irregular form, the task is more challenging where internal changes are needed to generate plural forms. These changes are not arbitrary and are governed by rules. However, these rules are variable and require finding the root of the singular noun and its pattern so that they can be matched to the suitable pattern of the irregular plural form. Table 1 shows examples of Arabic plural nouns.

In this work, we investigate the possibility of generating each of these three forms and show the success of the proposed framework in predicting previously unseen irregular plural inflections. Systems that translate singular-to-plural have great potential on improving applications such as MT, text generation and summarization, named entity recognition, language learning and education, and data augmentation to name a few.

The remaining parts of this paper are organized as follows: in the second section, we review works related to morphological re-inflection generation in both Arabic and English and we shed light on their approaches to deal with morphological data sparsity. The third section describes

[a]https://github.com/sigmorphon/2022InflectionST

Table 1. Examples of the different Arabic re-inflected plural forms

Singular noun	Transliteration	Meaning	Plural noun	Transliteration	Meaning	Type of plural form
"معلم"	"mElm"	Male teacher	"معلمين"	"mElmyn"	Male teachers	Masculine
"معلمة"	"mElmp"	Female teacher	"معلمات"	"mElmAt"	Female teachers	Feminine
"رف"	"rf"	Shelf	"رفوف"	"rfwf"	Shelves	Irregular

materials and methods used in this work including data, preprocessing steps, model architecture, and evaluation metrics. Then, the results and performance of our model during training and validation are presented in the fourth section. The fifth section is a detailed discussion of the model's performance that highlights weaknesses and strengths of the model using examples from the validation and the test datasets. Finally, in the sixth section, we interpret the predictions of the model using parallel coordinates plots and heat maps based on the saliency maps. The seventh section concludes this work and discusses possible future work.

2. Related work

From the general definition of morphological re-inflection, as generating an inflected form from a source lemma given morphological and syntactic properties of both source and target words, emerges a variety of special cases and branches. The literature is rich with NLP approaches and models developed to tackle the modeling challenges imposed by such special cases. In this section, we review some of the existing work on morphological inflection NLP modeling, shedding the light on the challenges that required the special approach followed by each work.

Low-resource languages impose challenges on NLP models and hinder the high capabilities of neural language models which rely on the existence of abundant annotated datasets to perform satisfactorily (Cotterell *et al.* 2018). From the perspective of morphological generation, this problem can be spotted in morphologically rich languages in which a lemma might have various inflected forms with low frequencies raising the problem of data sparsity (Jin *et al.* 2020). Existing work approaches this problem from different angles. One of the proposed solutions for morphological inflection on low-resource languages is the use of a cross-lingual transfer approach. Kann *et al.* (2017) showed that knowledge obtained by training the model to morphologically complete paradigms on HRLs can be transferred to LRLs. Kann *et al.* (2020) went a step further by fine-tuning the model trained on HRLs on examples from LRLs.

Unsupervised learning represents a valuable option for developing NLP models when faced with the problem of annotated data scarcity (Vlachos, 2011). Inspired by children's unsupervised ability to absorb morphological knowledge (Berko, 1958), unsupervised learning was employed by Jin et al. (2020) for morphological completion of paradigms. Exploiting only raw text and lists of lemmas, they developed a novel and totally unsupervised model to aid in the construction of inflection tables.

Before the introduction of transformers (Vaswani *et al.* 2017), RNN-based sequence-to-sequence models were shown to achieve remarkable results on the task of morphological re-inflection generation for morphologically rich languages. Faruqui *et al.* (2015) employed a simple RNN encoder–decoder model that works on the character level to generate the output of character sequence inflected form from the input of character sequence root form. Their model was language-independent and achieved impressive results when compared to previous machine learning and lexicon-based morphological transducers. However, their model also suffered from the aforementioned data scarcity problem when tested on some morphologically rich languages. As an alternative, Aharoni and Goldberg (2016) made use of a hard attention mechanism to overcome the data high demand problem of soft attention and RNN-based sequence-to-sequence models. Exploiting the alignment of characters between inflected forms with the characters of corresponding root forms, they forced their RNN-based model to only pay attention to the relevant hidden state in input corresponding to the aligned hidden state of the output. Hence, at each time step, the model learns when to produce an output character and when to advance its attention to the next character in the input sequence. This controllable attention mechanism helped them outperform the previous frameworks on small datasets.

The previously mentioned work has drawn researcher's attention to investigate the virtues of applying a hard attention mechanism on the task of MIG for LRLs. Wu et al. (2020) emphasized that the key to obtain high performance from hard monotonic models is the joint training on hard monotonic alignment and transduction. Their model outperformed soft attention and hard non-monotonic attention models on three transduction tasks: grapheme-to-phoneme, morphological inflection, and named entity transliteration.

When investigating the scarcity of datasets for MIG tasks, one cannot ignore the tremendous efforts made to enrich the literature with annotated datasets, hence enabling supervised MIG. Data augmentation techniques are widely employed to extend the size of resource-poor languages' datasets for transduction tasks such as MT (Liu, Ryan, and Hulden, 2021) and MIG (Anastasopoulos and Neubig, 2019a).

Back translation is one of the famous data augmentation techniques that have been employed in MT tasks for LRLs. Recently, Liu and Hulden (2021) applied back translation to enlarge MIG datasets. However, in their discussion, they concluded that the back translation technique can only be applied in MIG when the unlabeled dataset is of the same quality as the labeled dataset.

Efforts to enrich morphological inflection datasets are not restricted to data augmentation techniques. Guriel *et al.* (2022) argued that the flat annotation scheme presented in the famous morphological inflection repository UniMorph[b] produces morphological inflection tables that are not generalizable to be used with all languages, especially those that feature complex argument marking. In their work, they employed layered annotation to change the annotation schema of UniMorph from flat to hierarchical, resulting in an expansion of the number of tables and verb forms by four and six times, respectively.

Arabic is a perfect example of a rich morphological language with low resources (Mousa *et al.* 2013). As is the case with most rich morphological languages, Arabic MIG research is severely undermined by this scarcity of resources. When reviewing advances in Arabic MIG research, a modest number of researches float on the surface, where most of them are rule-based such as Attia (2005), Riesa and Yarowsky (2006), Cavalli-Sforza *et al.* (2000), and Shaalan *et al.* (2006). Zollmann *et al.* (2006) used Buckwalter Arabic Morphological Analyzer (Buckwalter 2004) to find inflected forms of Arabic words to be used in an Arabic to English speech translation task in an attempt to fill in the morphological inflection gap between the two languages.

Although literature is rich with work that investigates the computational MIG for languages other than Arabic, the task of singular to plural transduction is not sufficiently addressed. The English plural system is highly dominated by regular forms with handful of exceptions to the regular suffixation process (Marchman, Plunkett, and Goodman 1997). This explains the scarcity of research addressing the computational acquisition of English nouns' plurals. This also justifies the reliance on rule-based methodologies that make use of existing linguistic dictionaries to build computational pluralization models as in Conway (2023). One of the earliest uses of neural networks for pluralization of English nouns was conducted by Plunkett and Juola (1999). They used a simple multi-perceptron neural network to model the acquisition of English verbs past tense and plurals of English nouns. The work reported by Plunkett and Juola (1999) showed that their model mimics children's acquisition of morphological knowledge represented by an initial error-free period followed by an over-regularization period; that is, applying regular suffixes to nouns and verbs that require irregular inflections. The German plural system, on the other

[b]https://unimorph.github.io

hand, relies on the addition of a regular suffix (/-e/, /-er/, /-en/, /-s/ or /-ø/) to the singular form which makes the modeling of pluralization process closer to a classification problem. Dankers *et al.* (2021) employed a simple unidirectional recurrent encoder–decoder model based on Long-Short-Term-Memory (LSTM) layers with no attention mechanism, while the research reported by Beser (2021) compared the performance of a bidirectional LSTM model to a vanilla transformer model. The performance of both approaches relied immensely on the frequency of each suffix in a German corpus. Arabic plural system, on the other hand, is more complex as it relies on both suffixation and alteration to the stem of a word with a large proportion of nouns that require the later. Rule-based models can perform well on this problem, but they do not mimic the natural acquisition of morphological knowledge followed by children. This promotes for the application of the most recent data-driven and high-performing approaches to model the problem of Arabic pluralization.

Recently, SIGMORPHON shared tasks that included Arabic as one of the languages for the task of MIG (Kodner and Khalifa 2022). Participants reported their results on the tasks of English past tense inflection, German plural noun inflection, and Arabic plural nouns inflection. As anticipated, participants' models achieved lowest scores on the task of Arabic nouns plural inflection. Kakolu Ramarao *et al.* (2022) employed a vanilla transformer architecture that takes as an input the individual characters of the source lemma, morpho-syntactic tag of the input, and morpho-syntactic tag of the output. To account of the problem of data sparsity, they use a data hallucination technique based on alignment and replacement steps similar to that of Anastasopoulos and Neubig (2019b). The alignment step determines ordered sequences of characters that are shared between the source lemma and the inflected form. The replacement step assures that when replacing any character, the new character has co-occurred with the preceding character in the training set. The authors reported their best average accuracy of 55.5% when the number of hallucinated data is 1000. Elsner and Court (2022) augment each training instance with a representative exemplar lemma and its output form. Training instances are also augmented with rule-based features constructed by aligning source lemma with its inflected form and exemplar lemma with its inflected form. During inference time, the character-level transformer architecture is also presented with the rule-based features and exemplars to guide its predictions. Instead of relying on data augmentation, Wehrli *et al.* (2022) proposed a system based on neural transducer that employs edit actions to tackle the problem of data sparsity. Their system is an ensemble of 10 character-level LSTM encoder–decoder architectures that differ from each other in the used dropout probability. Using a Levenshtein distance objective, the model is encouraged to choose the best edit action at each time step, which will result in the minimum Levenshtein distance between predicted and the true inflected forms. This enabled their model to achieve the best result in the competition with an accuracy of 59.6%. As discussed above, all participants rely on morpho-syntactic information as an input to their models. Moreover, some participants augmented their training data and used rule-based features to improve their model's predictions. In this work, morpho-syntactic features, augmented instances, and rule-based features are not considered. Although such extra information might help the performance of the model, we believe that our approach better resembles morphological acquisition and generalization of children.

3. Materials and methods

3.1 Dataset

The pretraining dataset was extracted from the Arabic Wikipedia (test split) corpus which was used for pretraining AraBERT (Antoun, Baly, and Hajj 2020). The dataset have been cleaned and split into train and test splits and is available on Kaggle.[c] We extracted all the unique words and removed the outlying ones that are composed of more than 16 characters. We ended up with 1,134,950 unique words which were split into train (90%) and validation (10%) subsets.

[c]https://www.kaggle.com/datasets/abedkhooli/arabic-bert-corpus/discussion/129597

The downstream task dataset is provided by the competition organizers.[d] It is composed of 1000 training singular–plural pairs, whereas the development dataset consists of 343 pairs. In a later stage of the competition, the organizers released the test set that consists of 600 examples. These examples are also included in the evaluation stage of this work. It is worth remarking that additional metadata that classifies the nouns as human/nonhuman and masculine/feminine are provided in the dataset; however this extra information is not fed into the proposed model which adds value to this work as the network needs to implicitly infer this knowledge in order to make a correct prediction.

3.2 Hardware configuration

We use distributed training on eight TPUs v3-8 provided by Kaggle Notebooks to pretrain the Character-BERT (CBERT) (El Boukkouri *et al.* 2020) on the MLM task. Training the model through one epoch takes 15 s. As the downstream task contains a limited amount of data (1000 training samples and 384 validation samples), the fine-tuning task is much less computationally expensive. Hence, the fine-tuning stage takes place on a single Google Colab Nvidia K80/T4 GPU and takes around 10 s to finish a full epoch.

3.3 Data preprocessing

First of all, all the punctuation marks and special characters are replaced with the "#" token which denotes a "separator". For the MLM task, each word is treated as a full sequence. We add two"#" tokens to indicate the start and the end of each sequence, however when masking, these separator tokens are excluded from the random selection of the masked tokens by setting their probability of selection to zero. Each sequence is then tokenized on a character-level basis where each character is given a unique index number. As the length of the Arabic words varies significantly, the number of masked characters is manually selected. We randomly mask one token for the words that are in the range of [3, 5] characters, randomly mask two tokens for the words that contain eight characters, randomly mask three tokens for the words that are in the range of [9, 13] characters, and randomly mask four tokens for the words that contain more than 13 characters. Finally, three versions of the dataset are constructed: the original one, the masked one, and the mask vectors. The latter contains only zeros and ones indicating whether the corresponding positions have been masked before being fed to the model. This mask vector is essential as the loss function will be calculated only based on those masked tokens as remarked in the original BERT paper (Devlin *et al.* 2019).

Secondly, for the fine-tuning stage, we have two separate datasets: the encoder input and the decoder input. The diacritics are removed from both parts as they need special handling and are not usually used in written Arabic. Then, each word is tokenized the same way as in the pretraining stage, and the sequences are padded to be equal in length. The chosen maximum length is 11; however, here for both the encoder and decoder inputs, we add the special character "#" to indicate the start and end of each sequence. For the labels which are fed only to the loss function, we add only the "#" token to the end of the sequence since the model is expected to predict the first token in the first pass of the decoder input where only the start "#" token at the beginning is unmasked in the decoder input sequence. The decoder inputs, which are the plural form of the nouns, are one token shifted to the left and masked such that the model is prevented from leaping into future tokens. Hence, the dataset for the fine-tuning section contains four parts: the pretrained CBERT inputs (singular nouns) which are the same as the main encoder inputs that is to be fine-tuned, the decoder inputs which are the plural form of the nouns shifted left, the masks used in the decoder

[d]https://github.com/sigmorphon/2022InflectionST/tree/main/part2

part to prevent the model from seeing the current and future token embeddings, and the labels which are the plural forms with an additional "#" token at the end of the sequence.

3.4 Model architecture

This work is inspired by Dong *et al.* (2022) who aimed at converting the graphemes into phonemes, that is, predicting the pronunciation of words via their spellings. In our work, we evaluate two different architectures under three different cases. The first architecture consists of two main parts: the pretrained CBERT that is pretrained on an MLM task where the masks are applied on a character-level basis and the CBERT-fused transformer (encoder–decoder) that fuses the pretrained CBERT output contextualized embeddings into the transformer-based singular-to-plural model. We hereafter refer to this architecture as "*fused*". The second architecture directly employs the pretrained CBERT as an encoder for the transformer decoder. We hereafter refer to this architecture as "*direct*".

The three different cases under which the two architectures are evaluated can be described as follows: the first case keeps CBERT and the embeddings frozen and only sets the additional fine-tuned layers to "*trainable*" during all the 100 epochs. The second case is identical to the first case during the first 50 epochs and then sets all the model's weights including the ones of the embedding layer to "*trainable*" for the remaining 50 epochs. The third case is identical to the first case with an additional weight-decay-based regularization to remedy the noticeable overfitting.

Figure 1 shows the first proposed framework in this study. The fusion process takes place as follows: after pretraining CBERT, we freeze its weights during the fine-tuning process, in the first and second cases and set them to *trainable* during the last 50 of 100 epochs in the second case, as the dataset provided for this stage is relatively tiny (1000 samples). The singular form of the nouns is fed to CBERT as tokenized sequences and the contextualized embeddings outputted from the CBERT are fed as keys and values to both the CBERT-encoder and CBERT-decoder multi-head attention layers. The encoder part of the transformer-based model, which is used for fine-tuning, contains two multi-head attention layers. The first is the CBERT-encoder attention where the keys and values come from the output of CBERT; meanwhile, the queries are the embeddings of the encoder of the transformer-based part. The second is the multi-head self-attention mechanism where all the keys, values, and queries are the token embeddings provided by the embedding layer of the transformer-based part. These embeddings are the same as the ones obtained during CBERT pretraining and are frozen during fine-tuning.

The decoder of the transformer-based part contains a CBERT-decoder multi-head attention layer that is identical in structure to the CBERT-encoder multi-head attention block. However, it is equipped with additional features. The encoder–decoder attention layer receives the keys and values from the transformer encoder; meanwhile, the queries are provided by the proceeding masked multi-head attention layer. Moreover, the inputs to the decoder part are shifted left as Arabic is written from right to left. This shifting enforces the feeding of the "#" special character as a first token to the decoder. The decoder is prevented from benefiting from the current and future tokens by means of masks applied in the multi-head attention layer that sets the values of the corresponding masked tokens in the softmax attention scores to zeros.

In both the encoder and decoder parts of the *fused* model, a drop net (Tan and Motani 2020) is added to average out the output of the two proceeding attention mechanisms. The addition and layer normalization layers depicted in the figure perform a skip connection process that enables the model to maintain the signal where the model becomes deeper and more susceptible to the vanishing gradient problem. The feed-forward network consists of two hidden layers, the first of which is activated using the Gaussian error linear units (GELUs) (Hendrycks and Gimpel, 2016) to add nonlinearity that enables the model to extract nonlinear features, and the second is a linear layer that unifies the output size of the mechanism with the outputs of all other attention blocks in the framework for the sake of consistency.

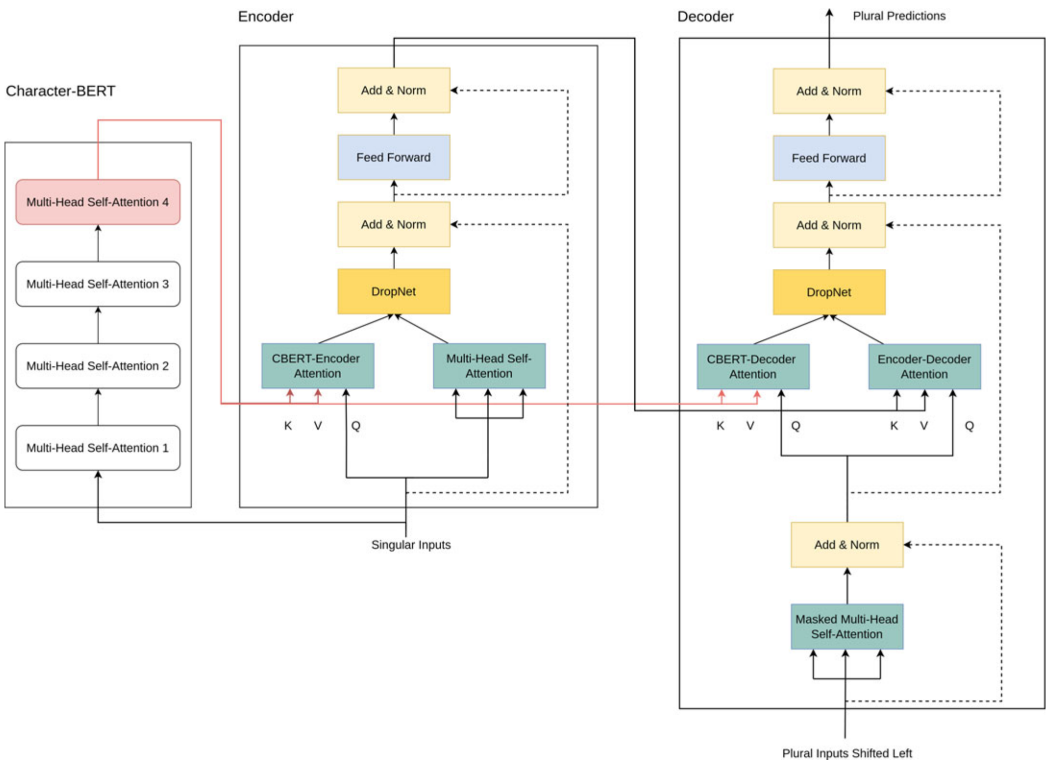
Character-BERT
Multi-Head Self-Attention 4
Multi-Head Self-Attention 3
Multi-Head Self-Attention 2
Multi-Head Self-Attention 1
Encoder
Add & Norm
Feed Forward
Add & Norm
DropNet
CBERT-Encoder Attention
Multi-Head Self-Attention
K
V
Q
Singular Inputs
Decoder
Plural Predictions
Add & Norm
Feed Forward
Add & Norm
DropNet
CBERT-Decoder Attention
Encoder-Decoder Attention
K
V
Q
K
V
Q
Add & Norm
Masked Multi-Head Self-Attention
Plural Inputs Shifted Left

Figure 1. The first proposed framework – *fused* architecture. The figure shows CBERT (left) which is used to produce the contextualized token embeddings, the encoder (middle) that receives the inputs from the outputs of CBERT and the singular form, and the decoder (right) that receives the inputs from the plural form, the CBERT outputs, and the encoder outputs.

The *direct* architecture shown in Fig. 2 directly employs CBERT as an encoder for the transformer model. The architecture is identical in configurations to the *fused* model, except for the removal of the separate encoder part, the drop net, and encoder–decoder attention layer.

During the pretraining stage, a batch size per replica of 256 is used, and since a distributed training on eight TPUs is adopted, the global batch size becomes 2048. Other hyperparameters and their values are: learning rate of 1e-5, number of heads is eight, embedding size is 256, number of blocks is four, key, value, and query dimensions are 64, and the buffer size used to shuffle the training dataset is 1024. The pretraining stage takes place over 100 epochs with 15 s required to finish training on a single epoch. Adam (Kingma and Ba 2015) optimizer is employed in conjunction with the categorical cross-entropy, while watching the categorical accuracy when saving the best model based on the performance on the validation set.

The full fused model is trained with a maximum length of 11, which is the length of the longest sequence in both the training and validation sets and was later found to be the maximum length in the test set. However, here either pads "p" or separator "#" tokens or a combination of them are added to the input sequence based on the part the sequence is being fed into as described in Section 3.3. All other configurations are the same as the pretraining stage.

3.5 Train-test splits

This work includes two tasks, namely pretraining and fine-tuning. Thus, we have two separate datasets, and each of them is split into two subsets: training and validation splits. We do not

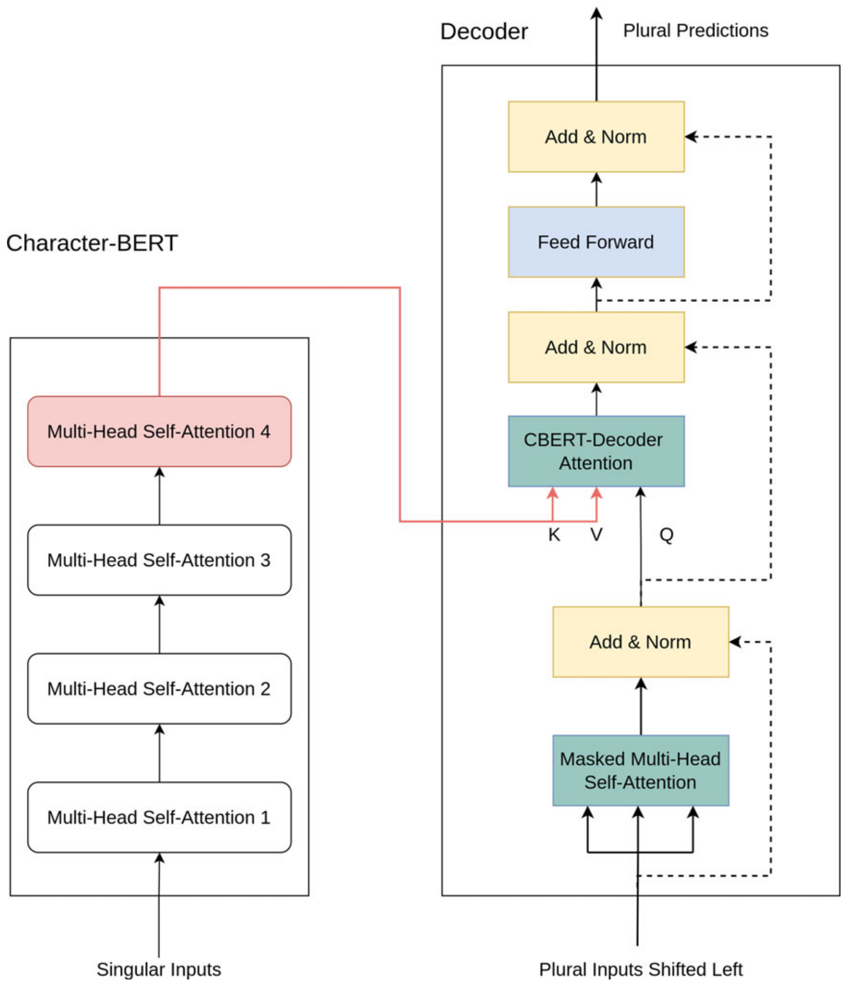
Decoder
Plural Predictions
Add & Norm
Feed Forward
Add & Norm
CBERT-Decoder Attention
K
V
Q
Add & Norm
Masked Multi-Head Self-Attention
Plural Inputs Shifted Left
Character-BERT
Multi-Head Self-Attention 4
Multi-Head Self-Attention 3
Multi-Head Self-Attention 2
Multi-Head Self-Attention 1
Singular Inputs

Figure 2. The second proposed framework – *direct* architecture. The figure shows CBERT (left) which is used to produce the contextualized token embeddings and the decoder (right) that receives the inputs from the plural form embeddings and the encoder outputs.

use cross-validation in the pretraining stage as we aim at generating a single model to extract the contextualized embeddings from. Performing K-fold cross-validation would lead to having K different models, and hence K different embeddings for each token. For the pretraining stage, 90% of the data was used for training and 10% for validation based on which the best model weights were saved. For the fine-tuning task, the dataset is already split by the competition organizers into training and development sets. It is possible to combine the two datasets and perform K-fold cross-validation which would provide a more accurate evaluation for the model performance; however, the process would be lengthy and computationally expensive.

3.6 Evaluation metrics

The categorical accuracy, which represents the top one accuracy, and the top five categorical accuracy alongside the categorical cross-entropy, which is used as a loss function to optimize the model's weights, are monitored during training, and the best-performing model based on the categorical accuracy is used as a final model.

The main metric measure that plausibly represents the model ability to convert a word into another form is the Levenshtein distance (Levenshtein *et al.* 1966) which accounts for the minimum number of edits required to convert one word into another word, where each deletion, insertion, or substitution is counted as one edit. A perfect match between the input pair would result in a zero distance, and the higher the distance is, the worse the performance is.

4. Results

There are three types of Arabic plural forms: sound masculine, sound feminine, and irregular, with the latter being the harder to predict as they do not usually follow a certain rule. The validation set consists of 343 singular–plural pairs. Out of these 343 examples, 50 are masculine, 146 are feminine, and 147 are irregular. For each position in the predicted sequence, the model outputs a probability distribution over all the possible 41 characters, where the 38 are the Arabic characters including all variants, and the additional three for the "p", "#", and "_" tokens which are responsible for masking in the pretraining task. The character with the highest probability score is used as a final prediction for the corresponding position. The model is allowed to output a sequence of length 11. However, we consider the final output sequence until the appearance of the first "#" token in the predicted sequence starting from the right-most position.

In this paper, experiments have been conducted to decide on whether to fuse CBERT into the transformer or to replace the transformer's encoder with CBERT directly. For each of the previously mentioned schemas, three cases have been considered:

- Case 1: Freezing CBERT during the whole 100 fine-tuning epochs.
- Case 2: Freezing CBERT during the first 50 epochs and then configuring it to be trainable during the remaining 50 epochs.
- Case 3: Same as the first case with the addition of 1e-4 weight decay in the Adam optimizer.

The results of each schema and case on both the validation and test sets are shown in Table 2. A setting that results in inferring a higher number of plural inflected forms with zero edits and a lower number of plural inflected forms with more than two edits is considered the best setting. Tables 3 and 2 show that this condition is met, both on the validation and test sets, by fusing CBERT into the transformer and fine-tuning its parameters on the last 50 epochs.

The performance and results of only the best schema will be considered for the remaining parts of this paper. Figure 3 presents the progress of the model training with regard to the loss value, top one accuracy, and top five accuracy metrics. It is noticeable that the loss value for the training set starts with a relatively high value of over 120 and then starts to decrease as the model converges to the optimized weights; meanwhile, the validation loss is almost maintained at a specific level from the beginning until the end of the training process. This is actually attributed to the class weights that are being used in the training set only, where the padding "p" token is given a weight of zero in order to prevent the model from focusing on this highly frequent token. This class weighting strategy is not applied to the validation set as the loss function is not used to update weights in the evaluation stage. However, with the progress of the model, and as the prediction of the "p" vanishes, the validation loss becomes more representative. The top five accuracy, as shown in Fig. 3 (lower-right), does not seem to be significantly different from the top one accuracy (lower-left). This indicates that when the model fails at predicting the exact current position's character, its other top four predictions are also likely to be incorrect. It is also worth mentioning that the accuracy scores are based on all the possible characters including the "p" character which is given

Table 2. Results of the *fused* and *direct* architectures on the test set under the three different cases. The readings are out of 600 which is the total number of samples in the test set

	Fused			Direct		
Levenshtein distance	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
Zero	198	**224**	**224**	172	202	120
One	61	**63**	**63**	52	61	46
Two	185	164	164	182	166	**204**
Total number of pairs inferred with less than three edits	444	**451**	**451**	406	429	370
Three	**96**	101	101	143	118	174
Four	53	**41**	**41**	43	44	45
Five	**7**	**7**	**7**	**7**	9	**7**
Six	**0**	**0**	**0**	**0**	**0**	2
Seven	**0**	**0**	**0**	**0**	**0**	2
Total number of pairs inferred with more than three edits	156	**149**	**149**	193	171	230

Table 3. Results of the *fused* and *direct* architectures on the validation set under the three different cases. The readings are out of 343 which is the total number of samples in the validation set

	Fused			Direct		
Levenshtein distance	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
Zero	122	**126**	94	102	115	80
One	**33**	29	25	23	31	20
Two	96	97	104	108	91	**118**
Total number of pairs inferred with less than three edits	251	**252**	223	233	237	218
Three	**61**	63	86	76	73	93
Four	29	**27**	29	33	33	32
Five	2	1	5	1	**0**	**0**
Six	**0**	**0**	**0**	**0**	**0**	**0**
Seven	**0**	**0**	**0**	**0**	**0**	**0**
Total number of pairs inferred with more than three edits	92	**91**	120	110	106	125

a weight of zero during loss calculation which, eventually, leads to the scarcity or even absence of the padding character prediction.

The proposed framework was capable of achieving promising results based on the Levenshtein distance metric where the predicted plural form of the singular input words achieved scores

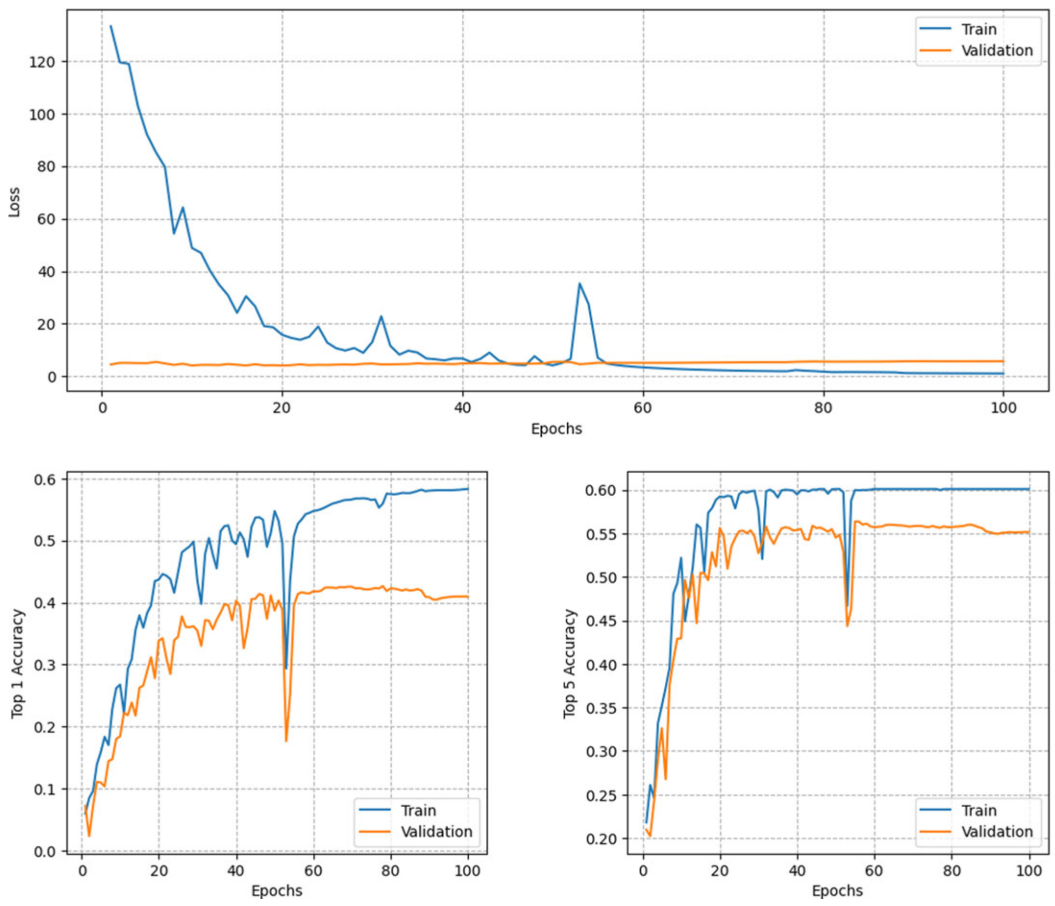

Figure 3. Training progress over 100 epochs. Upper graph shows the decrease in the loss value where only the loss of the training set is weighted based on the reversed frequencies of the tokens in the training set, giving the padding token a weight of zero. The loss of the validation set is not weighted and becomes more realistic as the training progresses and the prediction of the padding token diminishes. The lower left and right figures show the increase in the top one accuracy and top five accuracy metrics, respectively.

ranging between zero and seven minimum edits. Fig. 4 and Table 4 break down the distribution of the plural forms (feminine, masculine, and irregular) over the corresponding scores.

5. Discussion

In general, the model performs well at generating plural inflected forms with 126 predicted instances from the validation set and 224 predicted instances from the test set perfectly matching ground-truth instances. The occasions where the model did not perfectly match the ground-truth instances are dominated by irregular inflections. This highlights the ability of the model to generalize and generate plural inflected forms following the regular rules of inflection.

What is of more importance here is to get a deeper insight of the conditions where the model fails to generate inflected forms perfectly matching the ground-truth instances. Table 5 categorizes the instances for which the model was supposed to generate irregular inflection forms.

The fact that the model realized that it needs to generate irregular inflected forms and perfectly succeeded 30 times and 45 times for examples from validation and test sets, respectively, is

Table 4. Distribution of the predictions over the Levenshtein distances and the three plural forms for the best-performing model (*fused - Case 2*)

	Validation set			Test set		
Levenshtein distance	Feminine	Masculine	Irregular	Feminine	Masculine	Irregular
Zero	80	16	30	160	19	45
One	10	9	10	29	5	29
Two	34	15	48	35	23	106
Three	13	7	43	21	11	69
Four	9	2	16	9	4	28
Five	0	1	0	2	2	3

Table 5. Irregular plural form analysis. The readings are represented as (validation set readings and test set readings)

Levenshtein distance	Number of instances	Number of instances realized to be needing irregular inflection	Number of failures
Zero	(30, 45)	(30,45)	(0, 0)
One	(10, 29)	(9,28)	(1, 1)
Two	(48, 106)	(20,56)	(28, 50)
Three	(43, 69)	(5,10)	(38, 59)
Four	(16, 28)	(1,2)	(15, 26)
Five	(0, 3)	(0,0)	(0, 3)

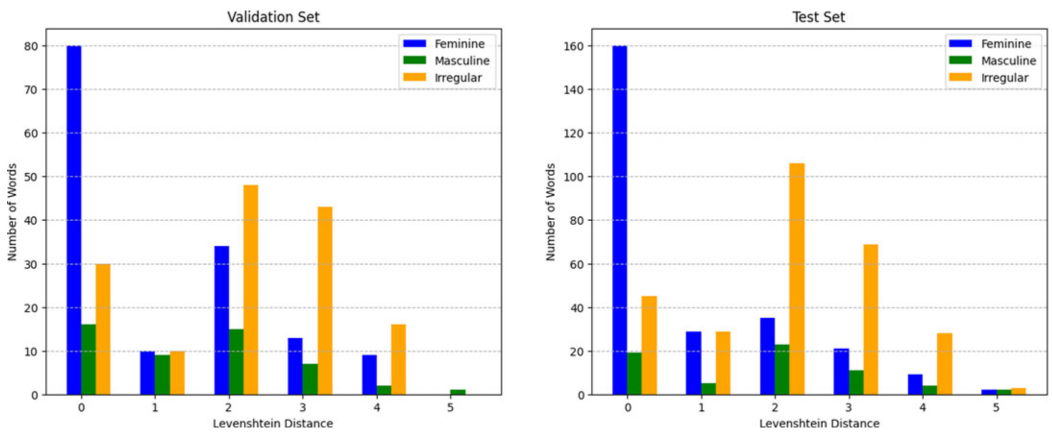

Figure 4. Distribution of plural forms over Levenshtein distance scores.

Table 6. Regular feminine plural form analysis. The readings are represented as (validation set readings and test set readings)

Levenshtein distance	Number of instances	Number of instances realized to be needing feminine inflection	Number of instances treated as regular masculine	Number of failures
Zero	(80,160)	(80,160)	(0,0)	(0,0)
One	(10,29)	(5,24)	(0,0)	(5,5)
Two	(34,35)	(4,5)	(13,11)	(17,19)
Three	(13,21)	(1,0)	(0,2)	(12,19)
Four	(9,9)	(0,0)	(0,1)	(9,8)
Five	(0,2)	(0,0)	(0,0)	(0,2)

promising. Generating irregular inflection forms for some Arabic words composed of three letters might require the addition of the letter "و"–"w" after the second letter in some occasions, for example, "قبر: قبور"–"qbr: qbwr" and the addition of the letter "ا"–"A" in other occasions, for example, "قفر: قفار"–"qfr: qfAr". Moreover, for some other Arabic words composed of three letters, the irregular plural form is obtained by the addition of the letter "أ"–">" at the beginning of the word and the addition of the letter "ا"–"A" after the second letter, for example, "خطر: أخطار"–"xTr: > xTAr". These rules are not exchangeable, and hence, applying the wrong rule to a word results in an inflected form with one or two Levenshtein distances away from the correct inflected form. It is hard for a model that does not rely on hand-crafted rules to decide which of the aforementioned rules to use. Therefore, it is important to highlight occasions in which the model realized the need for generating irregular plural form but failed by using an unsuitable rule. In Table 5, a failure denotes the case when the model produces a completely incomprehensible word or when the model applies a regular inflection rule instead of an irregular inflection rule. The latter dominated the failures of the model and is interesting, since children make the same mistake when learning the generalization of inflection rules.

Table 6 categorizes the instances for which the model was supposed to generate regular feminine inflection forms. As expected, for the case of plural feminine inflection form generation, the model performed better than the case of irregular inflection generation. There were 80 times and 160 times where the model perfectly matched the ground-truth instances from the validation and test sets, respectively. Some ground-truth feminine plural nouns miss a letter which resulted in predicted feminine plural nouns being one Levenshtein distance away from the ground-truth feminine plural forms (singular: "طوربيد"–"Twrbyd", ground truth: "طربيدات"–"TrbydAt", predicted: "طوربيدات"–"TwrbydAt"). For some words, the model predicted masculine plural forms instead of feminine plural forms (singular: "مسلم"–"mslm", ground truth: "مسلمات"–"mslmAt", predicted: "مسلمين"–"mslmyn") resulting in predicted plurals that are two edits away from the ground-truth plurals. This is understandable since we did not feed gender instructions to our model.

Finally, Table 7 categorizes the instances for which the model was supposed to generate regular masculine inflection forms. The masculine inflection form has the least number of instances in the validation and test sets. The model generally performed well with 16 and 19 instances perfectly predicted from the validation and test sets, respectively.

A crucial aspect when analyzing inflected forms generated with one or more Levenshtein distances away from their ground-truth inflected forms is to check whether these inflected forms

Table 7. Regular masculine plural form analysis. The readings are represented as (validation set readings and test set readings)

Levenshtein distance	Number of instances	Number of instances realized to be needing masculine inflection	Number of instances treated as regular feminine	Number of failures
Zero	(16,19)	(16,19)	(0,0)	(0,0)
One	(9,5)	(9,3)	(0,0)	(0,2)
Two	(15,23)	(2,3)	(12,19)	(1,1)
Three	(7,11)	(0,0)	(0,1)	(7,10)
Four	(2,4)	(0,0)	(0,0)	(2,4)
Five	(1,2)	(0,0)	(0,0)	(1,2)

are comprehensible by humans and can be corrected mentally or not. Generally, a promising proportion of generated inflected plurals, that are one or two Levenshtein distances away from their ground-truth plurals, are comprehensible. These understandable plurals are not necessarily generated through the application of the correct inflection form or rule. Table 8 provides some examples of generated plurals, with one or two Levenshtein distances, that are understandable. For the case of examples with one Levenshtein distance, usually irregular plurals generated by mistakenly swapping two irregular inflection rules are mentally understandable. The understandable proportion of regular inflected plurals generated by following the desired inflection rule with the addition or deletion of one letter is also promising. For the case of examples with two Levenshtein distances, the largest proportion of generated plurals that follow a regular inflection rule instead of an irregular inflection rule are understandable. This is also the case when the model mistakenly swaps between two irregular inflection rules. In most cases, the model generates regular masculine plurals instead of regular feminine plurals or vice versa, and the generated inflected forms are also understandable. However, there are also a proportion of incomprehensible generated plurals that are one or two Levenshtein distances away from their ground-truth plurals as demonstrated in Table 9. We believe that this problem can be solved by increasing the size of the training dataset.

6. Interpretability

6.1 Methodology

The work in this section is inspired by the saliency-maps-based interpretation technique first introduced by Simonyan *et al.* (2013). In this section, we interpret the results produced by the proposed model via the gradients calculated with respect to the input embedding vectors, showing the positions where the model was paying attention to the most when generating the characters at each position. The proposed model is composed of three main parts, namely CBERT, the encoder, and the decoder, where each of these parts extracts the embeddings for the fed sequence from the same embedding layer. However, the input sequence to CBERT and the encoder is the singular form surrounded by the "#" token on both sides; meanwhile, the input sequence to the decoder is the plural form starting with the "#" token. The sequence length is equal in all of these three parts and is equal to 11. Each position in the input sequence is represented with a 256-d vector which is equal to the dimensionality of the gradients generated by differentiating the loss function with respect to the input embeddings. Hence, the gradients retrieved for each input sequence is a 11×256 array, and since we have three parts in the model, we will retrieve three 11×256 arrays.

Table 8. Examples of incorrect plurals generated by the model, but they are comprehensible by humans. IR: irregular, RF: regular feminine, RM: regular masculine

Distance	Singular	Gold plural	Predicted plural	Ground-truth form	Predicted form	Proportion
One	“فهم”–“fahm”	“أفهام”–“> hAm”	“أفهم”–“> fhm”	IR	IR	27 out of 92
	“ثمر”–“vmr”	“ثمار”–“vmAr”	“ثمور”–“vmwr”			
	“معتدي”–“mEtdy”	“معتدين”–“mEtdyn”	“معتديين”–“mEtdyyn”	RM	RM	
	“مناوئ”–“mnAw}”	“مناوئين”–“mnAw}yn”	“مناوءين”–“mnAwýn”			
	“إمكانية”–“< mkAnyp}”	“إمكانيات”–“< mkAnyA”	“إمكانات”–“< mkAnAt”	RF	RF	
Two	“قرض”–“qrD”	“قروض”–“qrwD”	“أقراض”–“> qrAD”	IR	IR	105 out of 261
	“حجم”–“Hjm”	“أحجام”–“> HjAm”	“حجوم”–“Hjwm”	IR	IR	
	“عطلة”–“ETlp”	“عطل”–“ETl”	“عطلات”–“ETlAt”	IR	RF	
	“جنحة”–“jnHp”	“جنح”–“jnH”	“جنحات”–“jnHAt”	IR	RF	
	“مسلم”–“mslm”	“مسلمات”–“mslmAt”	“مسلمين”–“mslmyn”	RF	RM	
	“مؤسس”–“m&ss”	“مؤسسات”–“m&ssAt”	“مؤسسين”–“m&ssyn”	RF	RM	
	“محتل”–“mHtl”	“محتلين”–“mHtlyn”	“محتلات”–“mHtlAt”	RM	RF	
	“مقاوم”–“mqAwm”	“مقاومين”–“mqAwmyn”	“مقاومات”–“mqAwmAt”	RM	RF	

Table 9. Examples of incorrect plurals that are difficult to comprehend by humans

Distance	Singular	Gold plural	Predicted plural	Proportion
One	صربي (Srby)	صرب (Srb)	صربة (Srbp)	65 out of 92
	منشأ (mn$)	منشآت (mn$t)	منشآا (mn$)	
	مصاب (mSAb)	مصابين (mSAbyn)	مصابن (mSAbn)	
Two	دمية (dmyp)	دمى (dmY)	دمياى (dmyAY)	156 out of 261
	خطاب (xTAb)	خطابات (xTAbAt)	خطباب (xTbAb)	
	مقدم (mqdm)	مقدمين (mqdmyn)	مقدما (mqdmA)	

To reduce the dimensionality of each position gradient so that it is represented as a scalar, we take the summation of the absolute values of each of these vectors. Ultimately, the vectors under study will have the same dimensions as the input sequences, that is, we will end up with three 11-d vectors for each input sample. For each gradient vector, and for a fairer comparison, we normalize the values such that the maximum and minimum values are one and zero, respectively. These normalized gradients are the ones used in our interpretability analysis.

6.2 Results and discussion

In this section, we depend on two main types of plot that show the model's attention when predicting each position in the final output, namely the parallel coordinates plot and the heat map. Furthermore, we divide the work to include four main cases. In the first case, we start with three examples which were perfectly predicted by the proposed model, that is, with zero Levenshtein distance. The three samples are selected such that they cover the three forms of Arabic plural: masculine, feminine, and irregular. In the second case, we show the performance of the model on the whole validation dataset where we average out the values of the summed absolute gradients of each position of the three parts of the model. Finally, in the third and fourth cases, we repeat what is done in case two, but focusing on the best (zero Levenshtein distance) and worst (five Levenshtein distance) scenarios, trying to extract any features that led to the difference between the model's performance on the two categories. Parallel coordinate plots that show the summed absolute gradient values of each position in the input sequence for each part of the proposed model when generating the irregular plural word "ذرائع"–"*rA}E" from its singular form "ذريعة"–"*ryEp" is depicted in Fig. 5.

The singular noun "ذريعة"–"*ryEp" is a quinquelitral root noun by itself, and its irregular plural is obtained by three modifications to this quinquelitral root: the modification of the internal vowel from "ي"–"y" to "ئ"–"}", deletion of "ة"–"p" of the singular form, and the insertion of an infix "ا"–"A". CBERT generates contextualized embeddings for characters; hence, it is reasonable that most attention is paid to positions near the position of the predicted character. The character that will be omitted "ة"–"p" receives the least attention as it is not informative for other characters' representations. From the encoder parallel coordinate plot, it can be inferred that when the model generates both the infix "A" to be inserted and the modified vowel, and it pays more attention to the internal vowel to be modified and the following letter. When generating the final letter "ع"–"E" in the plural form, the model pays more attention to the last letter in the singular form "ة"–"p" since this letter will be omitted. This is reasonable and conforms with the rules of generating irregular plural forms which requires paying attention to the vowels that will be modified and to letters that will be omitted. The decoder part works in an autoregressive manner by receiving the

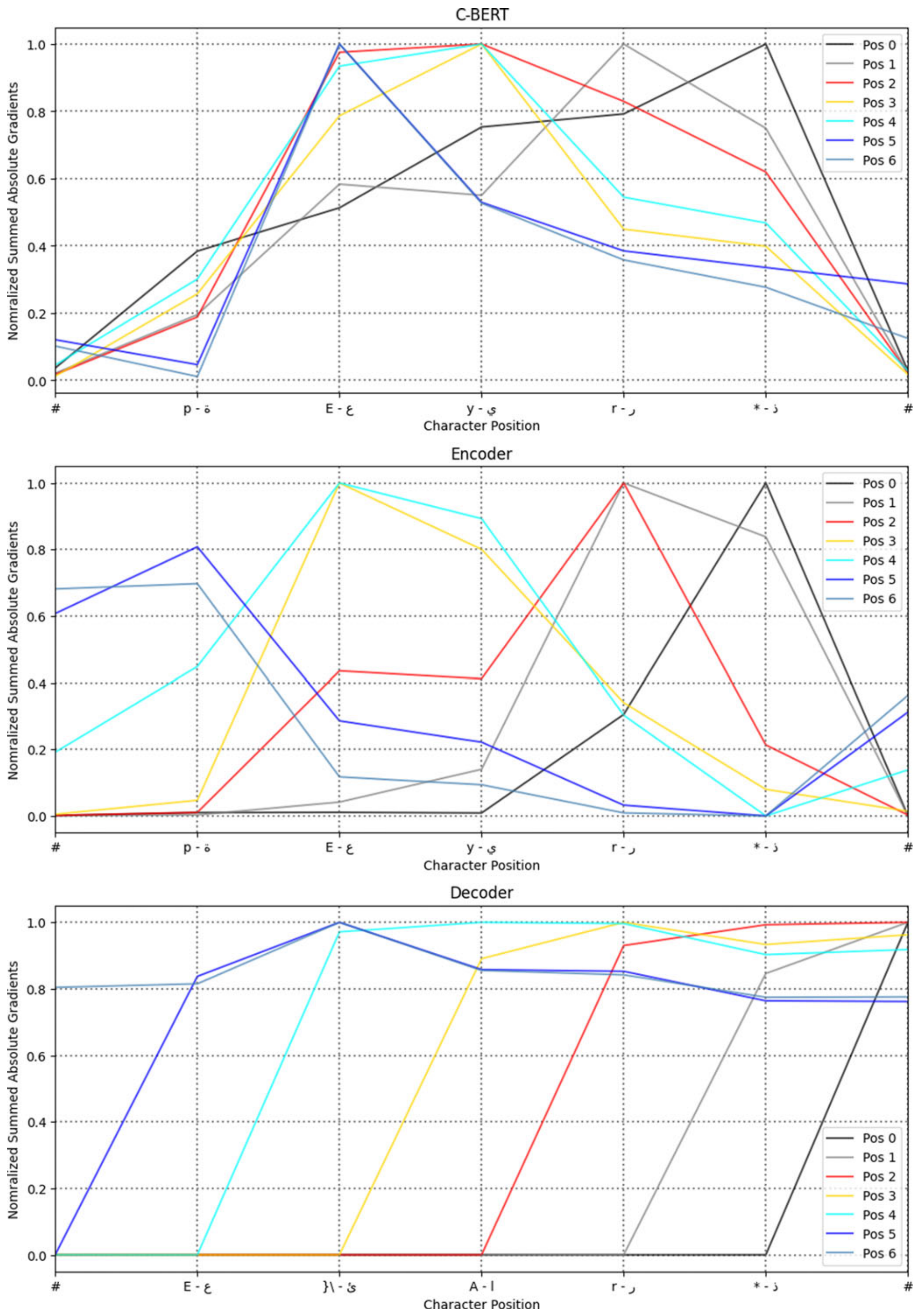

Figure 5. Parallel coordinate plots showing the mean absolute gradient values for each position in the input sequence "ذريعة"–"*ryEp" when predicting each position in the generated sequence "ذرائع"–"*rA}E". A plot for each part in the proposed model is shown: CBERT (top), encoder (middle), and decoder (bottom). This is one example where the model perfectly generates the plural irregular form for a singular word.

predictions of the previous characters, starting with the "#" token, and masking the current and future positions. The model consistently generates near zero gradients to the characters following the last unmasked character, which is quite intuitive that the model makes no use of the masked tokens when generating the final sequence. Fig. 5 depicts this behavior.

By taking one example of the masculine form where additional " ين"–"yn" characters are added to the end of a singular form to generate the plural, and exhibiting the parallel coordinate plots as shown in Fig. 6, we can see that for CBERT, the model consistently pays attention to positions near the position of the predicted character. An interesting remark is that at almost every position, a high attention is paid to the internal vowel. This is impressive since internal vowels play a crucial role in deciding the suitable plural form. A similar pattern is spotted in the encoder part. However, the decoder gradients are not much different from those of the previous example where at each predicted position the model pays most of its attention to the predicted character in the previous position.

As a final individual example, we take a sample from the feminine form, shown in Fig. 7, where an additional " ات"–"At" characters are added to the singular form. For the decoder part, it is obvious that the model behaves similarly as the previous two examples when generating the plural sequence where the highest gradients are delivered to the position of the current decoder input. The encoder and CBERT parts exhibit a similar pattern as seen in the previous examples by paying more attention to positions near the position of the predicted character.

Interpretation of individual model's predictions is time-consuming and does not provide a general explanation of the model's predictions. And hence, we explore heat maps that display the mean of the summed absolute gradients with respect to the inputs of each part of the model over the whole validation set. The heat maps, depicted in Fig. 8, show these aggregated gradients for three groups. The three heat maps composing the first row present the all-data group, the second row shows the group of data in the validation set that were perfectly predicted, that is, with zero Levenshtein distance, and the third row consists of the heat maps of the group with five Levenshtein distance. It can be seen that all the three groups follow the same manner with respect to the decoder part where they consistently pay more attention to the character preceding the current input. This is counterintuitive as it is expected from the model to benefit the most from the current input. However, looking at the decoder part separately does not reflect the complete picture. There are also the two other parts where the model still can gain information from when decoding the input sequences which we will discuss in the upcoming paragraphs.

The decoder works in an autoregressive manner, and hence gradients corresponding to masked positions will have zero values in the heat map. It can also be seen that the model constantly pays attention to the characters that lie in the third, fourth, and fifth positions when generating the sixth character and the following ones. This is because more than 76.4% of the plural words in the validation dataset have a length of less than or equal to six characters. Thus, the following characters in the decoder input sequences will be pads which have been given zero weights in the loss computation during training.

For the CBERT part, the second character in the input receives the most attention when generating the first two characters and the same goes for the fourth character when generating the rest of the plural form. This is understandable given that most of the plural words have a length less than or equal to six. Hence, the remaining characters in the CBERT's inputs are pads which were given zero weights during training. Finally, for the encoder gradients, we notice a slight difference between the behavior of the first and second groups on the one side and the five Levenshtein distance groups on the other side. It is clear that, for all groups, most attention is paid to the characters surrounding the predicted position which is reasonable. After the fourth position, the model somehow starts generating an almost uniform distribution over the last seven positions in the input sequence.

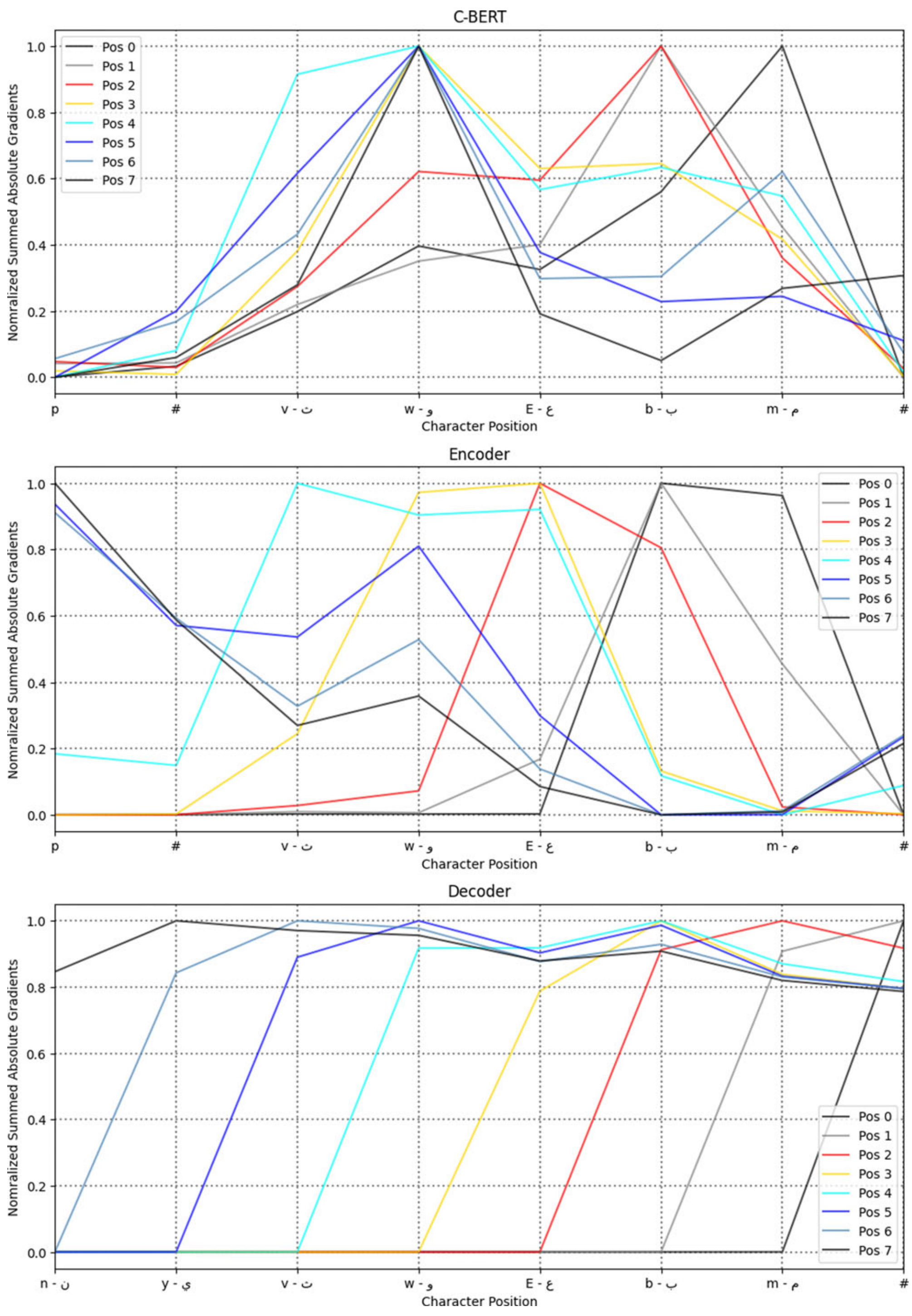

Figure 6. Parallel coordinate plots showing the mean absolute gradient values for each position in the input sequence "مبعوث"–"mbEwv" when predicting each position in the generated sequence "مبعوثين"–"mbEwvyn". A plot for each part in the proposed model is shown: CBERT (top), encoder (middle), and decoder (bottom). This is one example where the model perfectly generates the plural masculine form for a singular word.

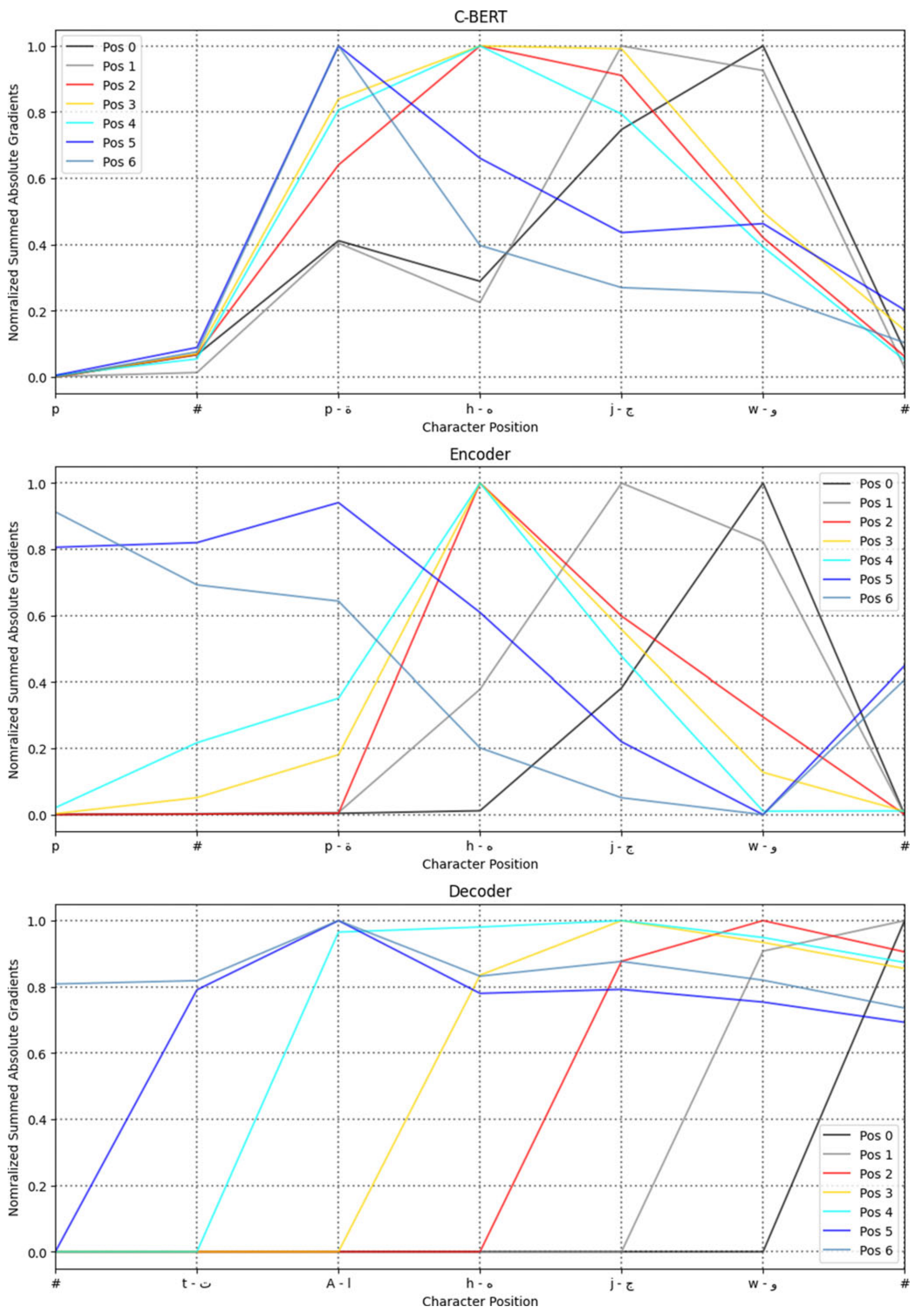

Figure 7. Parallel coordinate plots showing the mean absolute gradient values for each position in the input sequence "وجهة"–"wjhp" when predicting each position in the generated sequence "وجهات"–"wjhAt". A plot for each part in the proposed model is shown: CBERT (top), encoder (middle), and decoder (bottom). This is one example where the model perfectly generates the plural feminine form for a singular word.

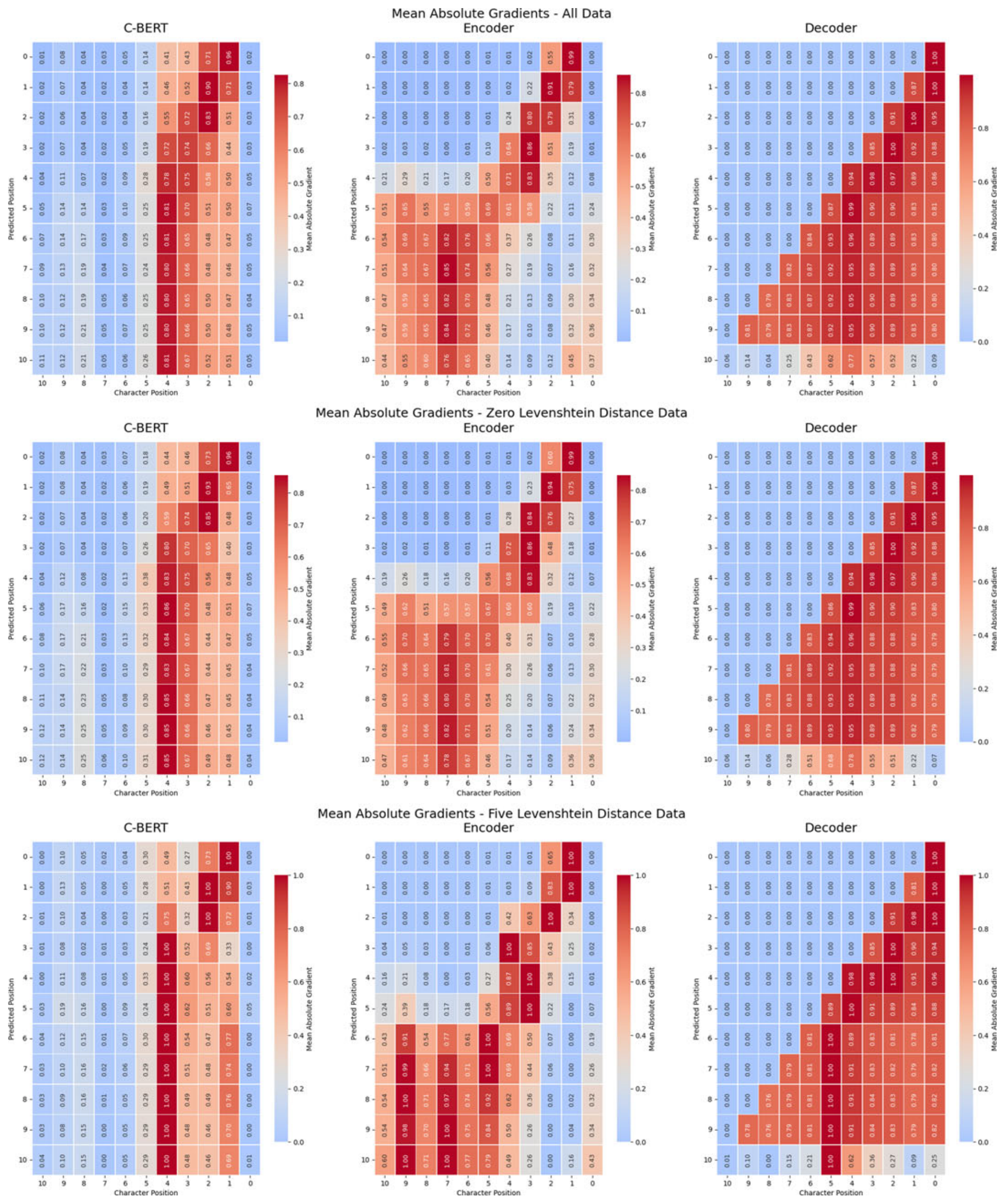

Figure 8. Heat maps showing the mean absolute gradient values for each position in the input sequences of all the data when predicting each position in the generated sequences. A plot for each part in the proposed model is shown: CBERT (right), encoder (middle), and decoder (left). The first row shows the three heat maps of the average of the all data. The second row shows the three heat maps of the samples predicted with zero Levenshtein distance; the values are presented as the mean of the heat maps of these samples. The third row shows the three heat maps of the samples predicted with five Levenshtein distance; the values are presented as the mean of the heat maps of these samples.

7. Conclusion

This work proposes an end-to-end deep learning framework aiming at converting singular nouns into their plural forms for the Arabic language. First, a CBERT is pretrained on an MLM task, and then the pretrained model is fused into a transformer encoder–decoder model. The proposed architecture has proved its effectiveness in the singular-to-plural conversion task with 126 out of 343 validation examples and 224 out of 600 test examples perfectly transformed. The vast majority of the predictions lie in the zero, one, and two ranges on the Levenshtein distance which indicates the minimum number of character edits required to convert one word into another. The good performance of the proposed model is not restricted to the regular (masculine and feminine) plural forms; it has also proved its ability to predict the converted version of dozens of irregular examples. One point to mention is that additional metadata that indicates the class of the noun (human/nonhuman and masculine/feminine) is not used in this work as this kind of metadata is not likely to be available in real-world cases. This point adds more value to this study and shows the robustness of the developed model to first recognize the class of the input noun and then morphologically inflect it into the plural form. For future work, additional features can be incorporated to reinforce the proposed framework and increase its accuracy. Starting with the size of the training dataset, it was apparent in our work that the 1000 samples used for training were not enough to cover the distribution of all Arabic nouns leading to an underfitted model. Another option for enhancement is by utilizing some of the open-source Arabic parsers to first lemmatize the singular nouns and then incorporate these lemmas as a second input of the model. This option has high potential for improving the model's accuracy as Arabic linguists mainly depend on lemmatization in determining the correct infliction of a singular noun to its plural form.

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

References

Aharoni R. and Goldberg Y. (2016). Morphological inflection generation with hard monotonic attention. In *Annual Meeting of the Association for Computational Linguistics*.

Anastasopoulos A. and Neubig G. (2019a). Pushing the limits of low-rearticle-title morphological inflection. In *Conference on Empirical Methods in Natural Language Processing*.

Anastasopoulos A. and Neubig G. (2019b). Pushing the limits of low-rearticle-title morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics, pp. 984–996.

Antoun W., Baly F. and Hajj H. (2020). Arabert: transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11-16 May 2020*, p. 9.

Attia M. (2005). Developing a robust arabic morphological transducer using finite state technology. In *8th Annual CLUK Research Colloquium*. Citeseer, pp. 9–18.

Berko J. (1958). The child's learning of english morphology. *Word-Journal of The International Linguistic Association* **14**(2-3), 150–177.

Beser D. (2021). Falling through the gaps: Neural architectures as models of morphological rule learning. In *Proceedings of the 43rd Annual Meeting of the Cognitive Science Society*, pp. 1042–1048.

Bojar O., Chatterjee R., Federmann C., Graham Y., Haddow B., Huang S., Huck M., Koehn P., Liu Q., Logacheva V., et al. (2017). Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pp. 169–214.

Buckwalter T. (2004). Buckwalter arabic morphological analyzer version 2.0. linguistic data consortium, university of pennsylvania, 2002. ldc cat alog no. Technical report, Ldc2004l02. Technical report.

Cavalli-Sforza V., Soudi A. and Mitamura T. (2000). Arabic morphology generation using a concatenative strategy. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

Conway D. (2023). An algorithmic approach to english pluralization. (accessed 6 Feb 2023), https://users.monash.edu/damian/papers/HTML/Plurals.html.

Cotterell R., Kirov C., Sylak-Glassman J., Walther G., Vylomova E., McCarthy A.D., Kann K., Mielke S.J., Nicolai G., Silfverberg M., Yarowsky D., Eisner J. and Hulden M. (2018). The conll-sigmorphon 2018 shared task: Universal morphological reinflection. In *Conference on Computational Natural Language Learning*.

Dankers V., Langedijk A., McCurdy K., Williams A. and Hupkes D. (2021). Generalising to german plural noun classes, from the perspective of a recurrent neural network. In *CoNLL. 2021 - 25th Conference on Computational Natural Language Learning Proceedings*, pp. 94–108.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota. Association for Computational Linguistics, vol. **1**, pp. 4171–4186, Long and Short Papers.

Dong L., Guo Z., Tan C.-H., Hu Y.-J., Jiang Y. and Ling Z. (2022). Neural grapheme-to-phoneme conversion with pre-trained grapheme models. In *ICASSP. 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6202–6206.

Dreyer M. and Eisner J. (2011). Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 616–627.

El Boukkouri H., Ferret O., Lavergne T., Noji H., Zweigenbaum P. and Tsujii J. (2020). CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics (Online)*, Barcelona, Spain. International Committee on Computational Linguistics, pp. 6903–6915.

Elsner M. and Court S. (2022). OSU at SigMorphon 2022: Analogical inflection with rule features. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, Washington. Association for Computational Linguistics, pp. 220–225.

Faruqui M., Tsvetkov Y., Neubig G. and Dyer C. (2015). Morphological inflection generation using character sequence to sequence learning. In *North American Chapter of the Association for Computational Linguistics*

Guriel D., Goldman O. and Tsarfaty R. (2022). Morphological reinflection with multiple arguments: An extended annotation schema and a georgian case study. In *Annual Meeting of the Association for Computational Linguistics*.

Hendrycks D. and Gimpel K. (2016). Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415.

Jin H., Cai L., Peng Y., Xia C., McCarthy A. and Kann K. (2020). Unsupervised morphological paradigm completion. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 6696–6707. Online.

Kakolu Ramarao A., Zinova Y., Tang K. and van de Vijver R. (2022). HeiMorph at SIGMORPHON 2022 shared task on morphological acquisition trajectories. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, Washington. Association for Computational Linguistics, pp. 236–239.

Kann K., Bowman S.R. and Cho K. (2020). Learning to learn morphological inflection for resource-poor languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. **34**, pp. 8058–8065.

Kann K., Cotterell R. and Schütze H. (2017). One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics, pp. 1993–2003.

Kann K. and Schütze H. (2016). Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany. Association for Computational Linguistics, pp. 555–560.

Kingma D.P. and Ba J. (2015). Adam: A method for stochastic optimization. The 3rd International Conference for Learning Representations, San Diego, USA.

Kodner J. and Khalifa S. (2022). SIGMORPHON–UniMorph 2022 shared task 0: Modeling inflection in language acquisition. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, Washington. Association for Computational Linguistics, pp. 157–175.

Koskenniemi K., et al. (1983). Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production, vol. **11**. Finland: University of Helsinki, Department of General Linguistics Helsinki.

Leminen A., Smolka E., Dunabeitia J.A. and Pliatsikas C. (2019). Morphological processing in the brain: The good (inflection), the bad (derivation) and the ugly (compounding). *Cortex* **116**, 4–44.

Levenshtein V.I.,, et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, vol. **10**. Soviet Union, pp. 707–710.

Liu L. and Hulden M. (2021). Backtranslation in neural morphological inflection. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pp. 81–88.

Liu L., Ryan Z. and Hulden M. (2021). The usefulness of bibles in low-resource machine translation. In *Proceedings of the Workshop on Computational Methods for Endangered Languages*, vol. **1**, pp. 44–50.

Magueresse A., Carles V. and Heetderks E. (2020). Low-resource languages: A review of past work and future challenges. arXiv preprint arXiv:2006.07264.

Marchman V.A., Plunkett K. and Goodman J. (1997). Overregularization in english plural and past tense inflectional morphology: A response to marcus (1995). *Journal of Child Language* **24**(3), 767–779.

Mousa A.E.-D., Kuo H.-K.J., Mangu L. and Soltau H. (2013). Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, pp. 8435–8439.

Narasimhan K., Karakos D., Schwartz R., Tsakalidis S. and Barzilay R. (2014). Morphological segmentation for keyword spotting. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 880–885.

Nicolai G., Cherry C. and Kondrak G. (2015). Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 922–931.

Plunkett K. and Juola P. (1999). A connectionist model of english past tense and plural morphology. *Cognitive Science* **23**(4), 463–490.

Riesa J. and Yarowsky D. (2006). Minimally supervised morphological segmentation with applications to machine translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pp. 185–192.

Schütze H., Cotterell R. and Kann K. (2016). Neural multi-source morphological reinflection. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Shaalan K., Monem A.A. and Rafea A. (2006). Arabic morphological generation from interlingua. In *International Conference on Intelligent Information Processing*. Springer, pp. 441–451.

Simonyan K., Vedaldi A. and Zisserman A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Singh A.K. (2008). Natural language processing for less privileged languages: Where do we come from? where are we going?. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Tan C.M.J. and Motani M. (2020). Dropnet: Reducing neural network complexity via iterative pruning. In *International Conference on Machine Learning*. PMLR, pp. 9356–9366.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł. and Polosukhin I. (2017). Attention is all you need. In *30th Annual Conference on Neural Information Processing Systems*, Long Beach, CA, USA.

Vlachos A. (2011). Evaluating unsupervised learning for natural language processing tasks. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, pp. 35–42.

Wehrli S., Clematide S. and Makarov P. (2022). CLUZH at SIGMORPHON 2022 shared tasks on morpheme segmentation and inflection generation. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Seattle, Washington. Association for Computational Linguistics, pp. 212–219.

Wu S., Cotterell R. and Hulden M. (2020). Applying the transformer to character-level transduction. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Zollmann A., Venugopal A. and Vogel S. (2006). Bridging the inflection morphology gap for arabic statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 201–204.

Cite this article: Radman A, Atros M and Duwairi R (2024). Neural Arabic singular-to-plural conversion using a pretrained Character-BERT and a fused transformer. *Natural Language Engineering* **30**, 1130–1154. https://doi.org/10.1017/S1351324923000475