Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Arabic abstractive text summarization using RNN-based and transformer-based architectures

Mohammad Bani-Almarjeh[a,*], Mohamad-Bassam Kurdy[a,b,c]

[a] *Syrian Virtual University, Damascus, Syria*
[b] *ESC Rennes School of Business in Rennes, France*
[c] *Burgundy School of Business in Dijon, France*

A R T I C L E I N F O

Keywords:
Natural language processing
Deep learning
Transfer learning
Text summarization

A B S T R A C T

Recently, the Transformer model architecture and the pre-trained Transformer-based language models have shown impressive performance when used in solving both natural language understanding and text generation tasks. Nevertheless, there is little research done on using these models for text generation in Arabic. This research aims at leveraging and comparing the performance of different model architectures, including RNN-based and Transformer-based ones, and different pre-trained language models, including mBERT, AraBERT, AraGPT2, and AraT5 for Arabic abstractive summarization. We first built an Arabic summarization dataset of 84,764 high-quality text-summary pairs. To use mBERT and AraBERT in the context of text summarization, we employed a BERT2BERT-based encoder-decoder model where we initialized both the encoder and decoder with the respective model weights. The proposed models have been tested using ROUGE metrics and manual human evaluation. We also compared their performance on out-of-domain data. Our pre-trained Transformer-based models give a large improvement in performance with ~79% less data. We found that AraT5 scores ~3 ROUGE higher than a BERT2BERT-based model that is initialized with AraBERT, indicating that an encoder-decoder pre-trained Transformer is more suitable for summarizing Arabic text. Also, both of these two models perform better than AraGPT2 by a clear margin, which we found to produce summaries with high readability but with relatively lesser quality. On the other hand, we found that both AraT5 and AraGPT2 are better at summarizing out-of-domain text. We released our models and dataset publicly[1],[2].

1. Introduction

After the significant success achieved by employing neural networks for computer vision, many efforts have focused on utilizing these networks for natural language processing tasks, such as text summarization. Text summarization aims at extracting the core ideas expressed in a text. It can be categorized into two main categories: an extractive one, where the most informative sentences in the source text are being selected to be in the summary, and an abstractive one, where the salient information in the source text is being paraphrased into novel sentences. Most of the previous works that have been done for this task have used the extractive method, and

* Corresponding author.
E-mail addresses: banimarje@gmail.com (M. Bani-Almarjeh), t_bkurdy@svuonline.org (M.-B. Kurdy).

[1] https://huggingface.co/malmarjeh
[2] https://data.mendeley.com/datasets/7kr75c9h24/1

https://doi.org/10.1016/j.ipm.2022.103227
Received 13 July 2022; Received in revised form 14 November 2022; Accepted 5 December 2022
Available online 26 December 2022

this is especially the case for the works in Arabic (Kahla et al., 2021). Recently, several papers have employed RNN-based sequence-to-sequence models for Arabic abstractive summarization (Alahmadi et al., 2022; Al-Maleh & Desouki, 2020; Suleiman & Awajan, 2020, 2022; Wazery et al., 2022). However, compiling large labeled datasets to train these models is often impractical and very expensive (Han et al., 2021).

In the last few years, the Transformer model architecture has been introduced (Vaswani et al., 2017), which has offered performance improvement over RNN-based architectures. Also, a new approach called transfer learning has emerged and has quickly become the dominant method to train and use deep learning models. This approach consists of two phases: (1) the pre-training phase, where the model is pre-trained on a huge amount of data using a self-supervised training objective, and (2) the fine-tuning phase, where the model is fine-tuned on a downstream task using a supervised dataset (Ruder, 2019). Multiple recent studies have demonstrated that transfer learning achieves state-of-the-art results on almost all NLP tasks (Qiu et al., 2020). This proves that the knowledge learned by the neutral unsupervised learning can be successfully transferred to downstream tasks. Due to this success, a wave of large pre-trained models that have adopted this approach has stormed the NLP community in recent years (Antoun et al., 2021; Brown et al., 2020; Devlin et al., 2019; Lewis et al., 2020; Nagoudi et al., 2022).

Very few works have tried to leverage the Transformer architecture and the Transformer-based pre-trained models for Arabic. (Kahla et al., 2021) have fine-tuned mBERT (Devlin et al., 2019), mBART-50 (Tang et al., 2020), and AraBERT (Antoun et al., 2020) for Arabic abstractive text summarization following a cross-lingual approach and found that AraBERT gives the worst results among their other proposed models. (Nagoudi et al., 2022) have also fine-tuned AraT5 for Arabic abstractive multi-sentence summarization. However, the authors of (Kahla et al., 2021) did not give details about how they fine-tuned AraBERT and mBERT for text summarization considering that they are encoder-only Transformers and cannot be directly used for text generation tasks. Also, there is no previous work that leverages different RNN-based and Transformer-based architectures for Arabic summarization and compares the performance of different pre-trained models. In addition to that, there is a need for a standard Arabic abstractive summarization dataset of high-quality examples.

In this work, we built an Arabic text summarization dataset (SumArabic) (Bani Almarjeh, 2022) of high-quality content using the Common Crawl.[3] Then, we trained both an attentional RNN-based sequence-to-sequence model and a Transformer-based encoder-decoder model on our dataset. Other versions of these two models have also been trained with an additional dataset of 280,000 examples to investigate the impact of training using larger datasets. After that, we studied the efficiency resulting from leveraging different large pre-trained models for our task, where we fine-tuned the following models: mBERT, AraBERT, AraGPT2 (Antoun et al., 2021), and AraT5. To be able to use mBERT and AraBERT for text summarization, we built a BERT2BERT-based encoder-decoder model where we initialized both the encoder and decoder with the respective model pre-trained weights.

Our models have also been tested on an out-of-domain test set to assess their generalizability. We established performance benchmarks on our dataset and released it to the research community to enable further research.

Our contributions are:

1 We built and released a novel Arabic abstractive summarization dataset of high quality (SumArabic).
2 We leveraged different architectures for Arabic abstractive summarization including RNN-based sequence-to-sequence models and Transformer-based encoder-decoder models. We leveraged the following pre-trained models: mBERT, AraBERT, AraGPT2, and AraT5 by fine-tuning them on our summarization dataset. The BERT2BERT architecture has also been used when fine-tuning mBERT and AraBERT.
3 We evaluated the performance of the proposed models using ROUGE metrics and manual human evaluation. We also compared their performance on an out-of-domain test set.
4 We made our models publicly available to the research community.

2. Background and related work

In this section, we will review the relevant previous studies that have been done for abstractive text summarization in both English and Arabic. We will focus on both the studies that employed the RNN-based sequence-to-sequence model and the ones that employed the Transformer architecture.

2.1. English text summarization

(Rush et al., 2015) was the first to use a fully data-driven approach for the task of abstractive sentence summarization. The authors used a convolutional neural network at the encoder and an attentional feed-forward network at the decoder. (Chopra et al., 2016) modified this architecture by using a recurrent neural network at the decoder. (Nallapati et al., 2016) extended further this work by using a feature-rich encoder that uses embedding vectors for part of speech and named-entity tags, and a pointer network for copying words from the source text that are unseen during the training phase.

(Liu et al., 2018) used a Transformer-based language model to summarize very long input examples. The authors leveraged a two-stage approach where they first extracted the most important sentences from the text, then summarized these sentences by using

[3] https://commoncrawl.org

an abstractive method. Their work showed that using a decoder-only Transformer-based model outperforms using encoder-decoder Transformer-based architectures when summarizing long input sequences.

In a follow-up work, (Khandelwal et al., 2019) used a pre-trained decoder-only Transformer-based model for the task of abstractive text summarization, where they appended the summary of each example to the source text. This set-up achieved better performance than using a pre-trained encoder-decoder Transformer. Additionally, their pre-trained decoder-only model significantly outperformed a pre-trained encoder-decoder model when fine-tuned on only 1% of the used data, indicating that having all the model parameters being pre-trained, as in the former model, has a large impact when fine-tuning in low-resource settings.

To use BERT (Devlin et al., 2019) for text summarization, (Liu & Lapata, 2019) have developed a modified version of BERT that encodes a document and creates representations of its sentences. They proposed $\mathrm{B_{ERT}S_{UM}}$, which is an extractive summarizer that stacks multiple inter-sentence Transformer layers on top of BERT outputs and uses a final layer for sentence classification. For abstractive summarization, they used an encoder-decoder model where the encoder is $\mathrm{B_{ERT}S_{UM}}$ and the decoder is a Transformer that was initialized randomly. Both of the two proposed models achieved better performance than baseline models.

The BERT2BERT architecture has been first introduced by Rothe et al., (2020) who developed a Transformer-based sequence-to-sequence model and did a comprehensive study on warm-starting it with pre-trained BERT, GPT-2 (Radford et al., 2019), and RoBERTa (Liu et al., 2019) checkpoints. The authors evaluated their models on multiple natural language generation tasks such as text summarization and proved that initializing the encoder with BERT or RoBERTa outperforms randomly initializing it, indicating the importance of having good representations of the source text at the encoder. They also proved that using BERT or RoBERTa at the decoder outperforms using GPT-2.

Many recent works have developed very large pre-trained models that can be used in the context of abstractive text summarization (Lewis et al., 2020; Qi et al., 2021). (Lewis et al., 2020) introduced BART, which is an encoder-decoder model where the encoder is presented with a corrupted version of the text while training, and the decoder is requested to generate the original text. The model achieves significant improvement on both abstractive and extractive text summarization tasks compared to previous works. (Zhang et al., 2020) proposed a novel self-supervised pre-training objective called Gap Sentences Generation (GSG) which is tailored specifically for text summarization. In this pre-training objective, the most important sentences in the input text are masked and the model is requested to generate them based on the remaining sentences. The authors evaluated their approach using a set of 12 summarization datasets covering various domains and achieved state-of-the-art results on all of them.

2.2. Arabic text summarization

Most of the work that has been done for Arabic text summarization was based on the extractive method (Alami et al., 2018, 2019). There are few works published recently that applied an abstractive approach.

(Suleiman & Awajan, 2020) proposed an attentional RNN-based sequence-to-sequence model with a two-layer bidirectional LSTM encoder and a one-layer LSTM decoder. The first layer at the encoder encodes the word embeddings of the input text, whereas the second layer encodes the word embeddings of the named entities. Their evaluation results showed that using an additional layer at the encoder for representing named entities results in better representation of the input text, and therefore in better performance, compared to a baseline model that uses only one layer. They also investigated the use of two different pre-trained word embedding models and found that the quality of the used embeddings highly affects the quality of the summaries. In a follow-up work (Suleiman & Awajan, 2022), they investigated the use of a bidirectional LSTM encoder with three layers and a unidirectional LSTM decoder with one layer. The three layers at the encoder create representations of the input text, a set of keywords, and the named entities. This proposed model achieved better performance compared to different variations of the model that were used as a baseline.

In a similar approach, (Al-Maleh & Desouki, 2020) trained an attentional sequence-to-sequence model using a dataset of approximately 295k text-summary pairs extracted from an Arabic website. They employed the copy mechanism (Vinyals et al., 2015) in their model and applied a length penalty to penalize long summaries and a coverage penalty to prevent repetition. The authors found that using the copy mechanism leads to higher ROUGE-1 scores.

Recently, (Wazery et al., 2022) have compared the performance resulting from using different networks for Arabic abstractive summarization including GRU, LSTM, and bidirectional LSTM with different number of layers. They also studied the impact of pre-processing text and using different word embedding models. They found that using a sequence-to-sequence model with a bidirectional LSTM encoder of three layers and a unidirectional LSTM decoder gives better performance compared to the other model variations that were tested.

In another work, the authors of (Alahmadi et al., 2022) have followed the work of (Yang et al., 2020) and updated the RNN-based sequence-to-sequence architecture by employing a topic-aware module that consists of an RNN topic classifier and outputs a new vector to guide the decoder when generating the summary by placing more attention on the words that belong to the same topic. Their proposed model achieved 10.8% higher accuracy and produced better summaries according to human evaluation compared to several baselines.

On the other hand, the research introduced by Kahla et al., (2021) is one of the first works that fine-tuned large pre-trained models for the task of Arabic abstractive text summarization. The authors have fine-tuned mBERT, mBART-50, and AraBERT on a dataset of 21,508 news articles in Arabic. They also used a cross-lingual transfer-based approach to develop another three models: (1) an mBERT model that was first fine-tuned on a Hungarian dataset, then on Arabic, (2) an mBERT model that was first fine-tuned on an English dataset, then on Arabic, and (3) an mBART-50 model that was first fine-tuned on a Russian dataset, then on Arabic. When evaluating the proposed models, the authors found that the mBERT model that was fine-tuned on both the English and Arabic languages achieved the best performance.

The authors of AraT5 (Nagoudi et al., 2022) have evaluated their models on two tasks that are relevant to our work. The first is a news title generation task, where the authors fine-tuned their models on a dataset of 120k news articles, and the second is an abstractive text summarization task, where they fine-tuned the models using the WikiLingua dataset (Ladhak et al., 2020), which contains paragraph-summary pairs of 29,229 examples in Arabic. The proposed models achieved better performance on the two previous tasks compared to mT5 (Xue et al., 2021), which was used as a baseline.

Most of the previous works in this area of research in Arabic were focused on RNN-based models. In this work, we start by building a baseline model using the RNN-based architecture. Then, we leverage different Transformer-based models and compare the performance resulting from each architecture using a high-quality dataset. This includes building a BERT2BERT architecture and leveraging the following pre-trained language models: AraBERT, AraGPT2, AraT5, and mBERT.

3. Methodology

In this section, we first present the methodology we used for collecting and preparing our dataset. Then, we discuss the different models we have built and trained for the task of Arabic abstractive text summarization, including the model architectures and the training details.

3.1. Building the dataset

Since the quality of the deep learning models' outputs is highly correlated with the quality of the data used for training, we focused in this research on building a high-quality summarization dataset. A primary goal of our effort was to make it easy for researchers to rebuild our dataset. In this section, we will discuss the different stages we followed to build it.

3.1.1. Collecting the data

To build our dataset, we used Common Crawl to download the archived HTML files of the articles of the following two Arabic news websites: emaratalyoum.com and almamlakatv.com. We chose these two websites because we found that their articles have high-quality content and that the Almamlakatv website specifically has a style of writing that is distinct from other news websites. The rationales for using Common Crawl are to make it easy to rebuild our dataset by other researchers and to comply with the copyrights of the respective crawled websites. The data are downloaded using 83 Common Crawl index files that are between 2008 and July 2021. The following fields have been extracted for each record in every index file: url, file name, offset, and length. We filtered out the records with a duplicated url value and kept only the ones with a status value of 200, and a text/html MIME type. For Emaratalyoum website's records, we kept only the ones with a length value greater than 12,000 bytes, and a url value that contains one of the following section names: business, covid19, fashion-and-beauty, hotline, life, local-section, online, politics, sports, technology, and travel.

After downloading the data, we extracted the first paragraph of each article as the source text, the title as the summary, and the publishing date. We selected the first paragraph instead of the whole article because news articles are often written in the Inverted Pyramid style (Pöttker, 2003), where the lead of the article contains the most salient and important information and often represents the whole article. We filtered the data to remove the examples that have one of the following properties:

- An empty source text or summary
- A duplicated source text or summary
- A summary that contains a question mark "?", an exclamation mark "!", or a colon ":"
- A source text that is longer than 60 words, or a summary that is longer than 15 words
- A source text that is shorter than 10 words, or a summary that is shorter than 3 words
- A source text that is shorter than the summary

Table 1 shows the number of downloaded examples from the two websites before and after applying the filtering steps:

3.1.2. Preprocessing the data

The Arabic language is a morphologically rich language and therefore poses more challenges than some other languages like English. For example, the root word "طلب request" has over 260 word forms that can be written by adding certain affixes and clitics, like "سأطلبه I will request it" and "يطلبانهما they are asking for them", whereas there are only four word forms of the corresponding word in English. Another issue is the inconsistencies in using diacritics (Tashkil) and in omitting Hamza in Arabic writings. Tashkil is often omitted in Modern Standard Arabic (MSA), which is the language used in news, academia, and literature throughout the Arab World, and this makes the Arabic text more ambiguous since diacritics can change the meaning of the words. As an example, the word كتب means 'he wrote', whereas كتب means 'it has been written', and كتب means 'books'. Also, there are multiple dialects that are used in different regions of the Middle East, which in some cases have significant differences. All of these challenges need to be addressed when processing Arabic text. We focused on MSA in this work. To preprocess our data, we removed certain prefixes from the summaries such as "Image:" and "Video:", then we applied the preprocessing script that was introduced with the AraBERT model[4] and

[4] https://github.com/aub-mind/arabert/blob/master/preprocess.py

Table 1
Number of downloaded examples before and after applying filtering.

Source	Downloaded examples	Examples after filtering
emaratalyoum.com	161,040	67,826
almamlakatv.com	23,234	16,938
Total	184,274	84,764

Table 2
The average number of words per source text and per summary before and after preprocessing.

	Before preprocessing	After preprocessing
Avg # of words per source text	33.62	38.97
Avg # of words per summary	8.40	9.43

which consists of the following steps:

- Removing diacritics (Tashkil)
- Removing Tatweel (For example, the word "طـــالب" becomes "'طالب")
- Replacing links with the word: "[رابط]"
- Replacing email addresses with the word: "[بريد]"
- Removing HTML elements
- Adding a space before and after punctuation marks
- Adding a space between numbers and words

Table 2 shows the average number of words per both source text and summary before and after applying preprocessing. Fig. 1 displays the distribution of the number of words per both source text and summary.

3.1.3. *Tokenizing the data*

When preparing the data that will be used to train the RNN-based sequence-to-sequence models, we applied subword tokenization, which is an algorithm that segments words into subword units. Applying this algorithm has the benefits of reducing the vocabulary size and enabling the model to process unseen words. Doing subword tokenization is highly recommended and is more crucial when working with Arabic than English. (Alotaiby et al., 2009) did a preliminary analysis on both the Arabic and English versions of Gigaword Third Edition (Graff, 2007, 2007) and found that there are a total of 2.2 million unique words in the Arabic corpus compared to 1.26 million words in the corresponding English corpus, indicating that we need an Arabic lexicon with 76% more unique word types than an English one to cover a similar linguistic content. In a follow-up work, (Alotaiby et al., 2010) did a statistical study on clitics in Arabic using the same corpus and showed that doing clitics tokenization reduces the total vocabulary size by 24.54%. This highlights the importance of segmenting Arabic text before training. We used BERT's implementation of the WordPiece algorithm with a vocabulary of 20k tokens for both source texts and summaries when segmenting our data.

3.1.4. *Creating an out-of-domain set*

Before splitting the data into training, validation, and testing sets, we created an out-of-domain test set to test our models on examples that relate to a topic that was not seen during training, similarly to the work done by Straka et al., (2018). To create it, we started by processing the summaries by removing diacritics and applying Farasa's stemmer (Abdelali et al., 2016). Then, we tokenized the sentences and calculated their TF-IDF scores, before applying the KMeans algorithm to cluster the examples into 30 clusters. We chose the number of clusters to be 30 after testing several values. The PCA algorithm has then been used to visualize the examples with their clusters in a two-dimensional space. After examining the visualization, we chose a cluster with 652 examples to be our out-of-domain test set, as shown in Fig. 2. We found that the unifying theme of these examples is 'elections'.

3.1.5. *Splitting the data*

Splitting the data was done as follows: 90% of the examples were taken for training, 5% for validation, and 5% for testing. Table 3 shows the number of examples in each split of the dataset.

Fig. 3 lists the steps we took to prepare our dataset. We publicly published the dataset as a file that contains listings of the Common Crawl records and a script to recreate the data to the research community.

3.2. *Experiments*

In this work, we first developed an RNN-based sequence-to-sequence model (Seq2Seq-LSTM) and a Transformer-based sequence-to-

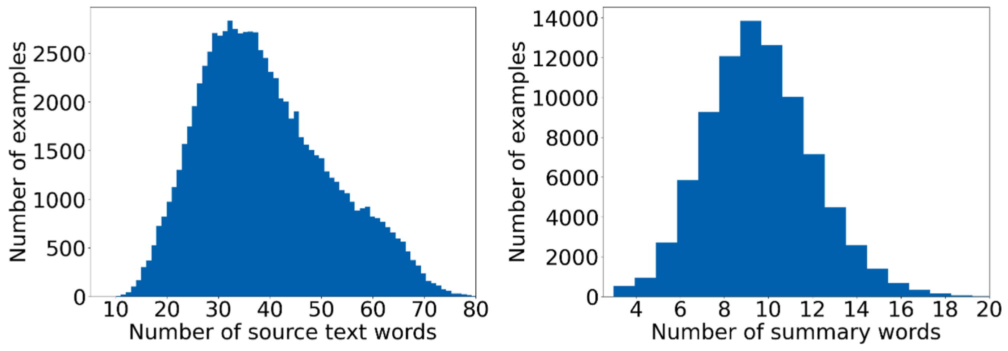

Fig. 1. The left diagram: the distribution of the number of words per source text after preprocessing. The right diagram: the distribution of the number of words per summary after preprocessing.

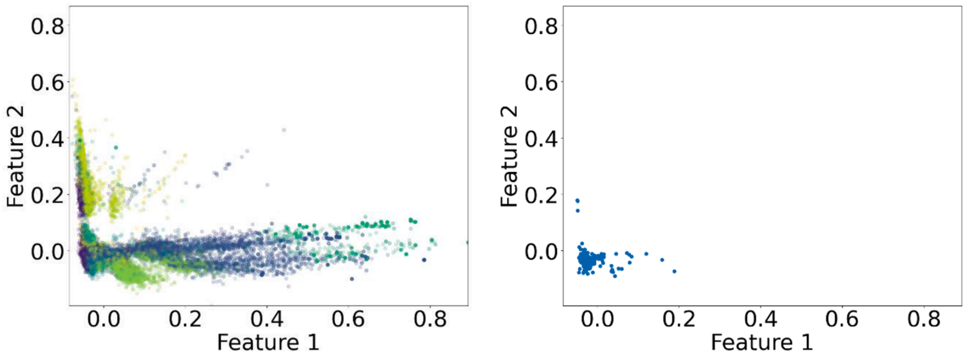

Fig. 2. The left diagram: the examples in our dataset clustered into 30 clusters and projected into a two-dimensional space, where each color corresponds to a cluster. The right diagram: the selected out-of-domain cluster which consists of 652 examples.

Table 3
The splitting of the dataset into training, validation, testing, and out-of-domain sets.

Dataset	Number of examples
Training	75,817
Validation	4121
Testing	4174
Out-of-domain	652
Total	84,764

sequence model (Transformer) and trained both of them on our dataset. Then, we collected an additional dataset of 280k examples from different news sources and trained new versions of the previous two models on a combination of the main and additional data, which we named Seq2Seq-LSTM+ and Transformer+. The process of collecting, preprocessing, and tokenizing the addition training data is the same as the one described in Section 3.1. We will discuss later the rationale for the re-training using the additional data.

Then, we leveraged multiple pre-trained large models. We started by fine-tuning the mBERT model, which is a multilingual model that is frequently used as a baseline in the literature. After that, we fine-tuned the Arabic pre-trained models: AraGPT2, AraBERT, and AraT5. To use mBERT and AraBERT in the context of text summarization, we utilized the BERT2BERT encoder-decoder architecture where we warm-started both the encoder and decoder with the respective model weights.

Fig. 4 shows a diagram of the models we trained and fine-tuned in this work.

In our experiments, the input and output tensors of the models have been padded to the length of the longest example, and when generating summaries, a limit on the length has been set to be between 5 and 40 tokens, unless specified otherwise. When fine-tuning the pre-trained models, we used the Adam optimizer with a learning rate of 5e-5.

Here is a detailed description of the architecture and the training details of each model:

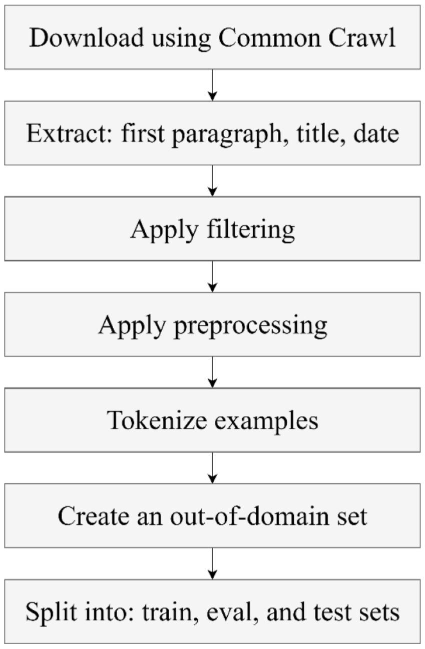
Download using Common Crawl
Extract: first paragraph, title, date
Apply filtering
Apply preprocessing
Tokenize examples
Create an out-of-domain set
Split into: train, eval, and test sets

Fig. 3. The data preparation steps.

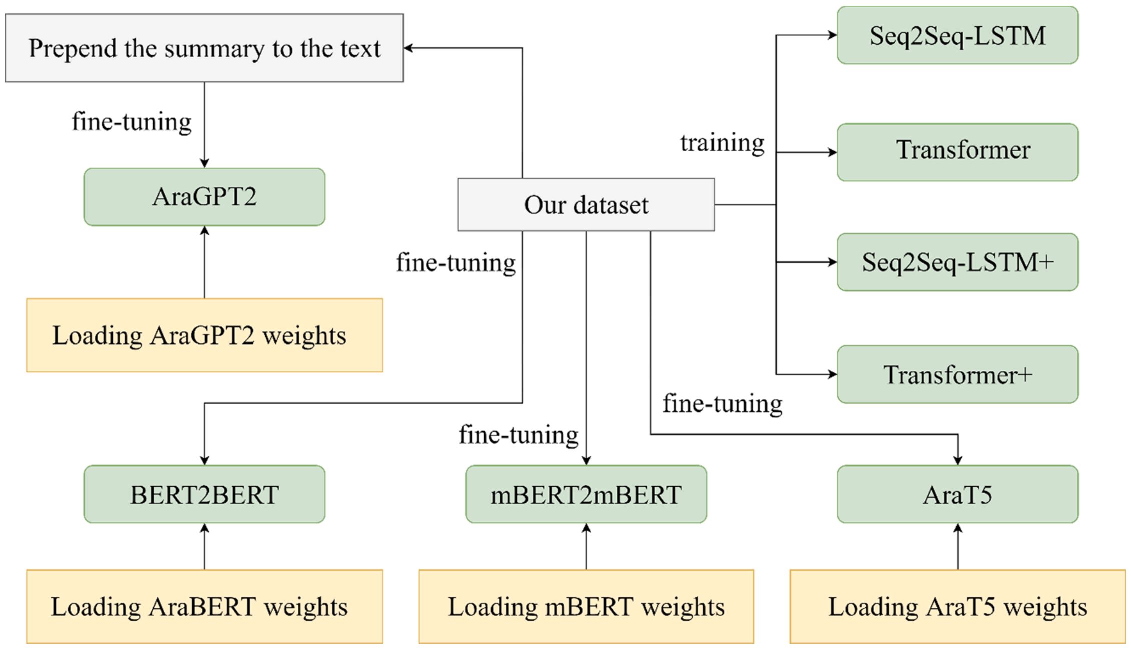

Fig. 4. A diagram of the proposed models.

3.2.1. *Seq2Seq-LSTM*

An attentional RNN-based sequence-to-sequence model that consists of a bidirectional LSTM (Bi-LSTM) encoder and a unidirectional LSTM decoder. Both the encoder and decoder start with an embeddings layer that processes the input tokens and creates context-independent representations of them. We chose the size of the embeddings at both the encoder and decoder to be 128 after trying the performance resulting from using multiple values. The resulting matrix from this layer, which is initialized randomly and is updated during training, has a shape of (20,000, 128) since our vocabulary consists of 20k tokens. The embeddings of the input are passed to a bidirectional LSTM network with 256 hidden units. We chose a bidirectional LSTM instead of a unidirectional one because the comprehension of the source text requires an understanding of both the left and right context of each word. The output of this layer is a context vector that encodes the meaning of the text. Since having a context vector of fixed size is not practical when encoding long documents, it became a common practice to use an attention mechanism (Chopra et al., 2016; Rush et al., 2015). We used this mechanism when generating the target tokens in our model. The decoder consists of a unidirectional LSTM, where the target tokens are generated in an auto-regressive way. It is first fed with the beginning-of-sentence (<bos>) token that signals the start of the summary. Then, the next token is generated with regard to the output representations of the encoder weighted by the attention weights. Eq. (1) describes the probability of a generated summary.

$$p(y) = \prod_{t=1}^{T} p\left(y_t \;\middle|\; \{y'_1 \ldots' y_{t-1}\}' c\right) \tag{1}$$

Where the probability of generating the summary tokens y is equal to the multiplication of the probabilities of generating each token conditioned on both previously generated tokens and the context vector c. The context vector is calculated according to Eq. (2).

$$c_t = \sum_{i=1}^{n} a_{t'i} h_i \tag{2}$$

Where n refers to the number of summary tokens, $a_{t'i}$ refers to the attention weight of both the generated token at step t and the ith token in the source text, and h_i refers to the hidden state of the ith token in the source text. The attention weight $a_{t'i}$ is calculated according to Eq. (3).

$$a_{t'i} = \frac{\exp\left(s_{t-1}^{T} W_a h_i\right)}{\sum_{k=1}^{n} \exp\left(s_{t-1}^{T} W_a h_k\right)} \tag{3}$$

Where W_a is a trainable attention matrix and s_{t-1} is the hidden state of the previously generated token. We used teacher forcing when generating the summary, which is a technique that allows passing the $t-1$ ground-truth token when generating the token at step t, instead of passing the previously generated token by the model. The decoder stops generating tokens when the end-of-sentence (<eos>) token is generated. Fig. 5 shows the architecture of the model and the summarization of a short example.

We trained this model for 16,500 steps using the Adam optimizer with a learning rate of 0.001 and a batch size of 64.

3.2.2. *Transformer*

The Transformer architecture consists of an encoder that processes the source text and a decoder that generates the summary one word at a time. Both the encoder and decoder consist of several sequential blocks where the output of one block is the input to the next. The Transformer has two main advantages over RNN-based sequence-to-sequence models: a) It processes the input sequences in parallel, which makes the computation much faster. b) It is better at learning long-range dependencies since it does not suffer from the vanishing gradients problem when dealing with long input sequences (Hochreiter et al., 2001). Similarly to Seq2Seq-LSTM, the first layer is an embedding layer that produces context-independent representations of the input. After that, a positional encoding is added to the embedding vectors to account for the order of the input tokens. The output is passed to a sequence of blocks that use a self-attention mechanism to learn rich context-dependent representations of the input. The self-attention output is calculated according to Eq. (4).

$$Attention(Q' K' V) = softmax_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

The calculation involves three matrices: Q (Query), K (Key), and V (Value). The K and V matrices represent the source text, whereas the Q matrix represents the summary. Every layer in each block is followed by a residual connection and a normalization layer, as described in Eq. (5).

$$\text{Output} = \text{LayerNorm}(x + \text{layer(x)}) \tag{5}$$

The output of the last block at the encoder contains the final representations that will be passed to the decoder. The decoder blocks differ in that they generate the output in an auto-regressive way and use a unidirectional self-attention to prevent attending to future tokens. Also, each block at the decoder has a cross-attention layer that is responsible for processing the output of the encoder. The decoder ends with a linear layer followed by a softmax function which generates a probability distribution for the next token in the summary. The model is illustrated in Fig. 6.

Similarly to the original Transformer's paper, we used 6 attention layers in both the encoder and decoder and 8 attention heads in each layer. The dimensionality of the encoder and decoder layers was set to 512, and the dimensionality of the feed-forward layers was set to 2048. We used AraBERT's tokenizer for tokenizing our data before passing them as inputs. When training the model, we accumulated gradients for two steps before back-propagation and trained for 21k steps with a batch size of 16.

3.2.3. *Seq2Seq-LSTM+*

Since the previous two models were trained from scratch on our training set that consists of only 75,702 examples, the models under-fitted the data. For this reason, we extracted an additional dataset of 280k examples and retrained both of the models on a combination of the additional and main data. The new models are named Seq2Seq-LSTM+ and Transformer+. When training Seq2Seq-

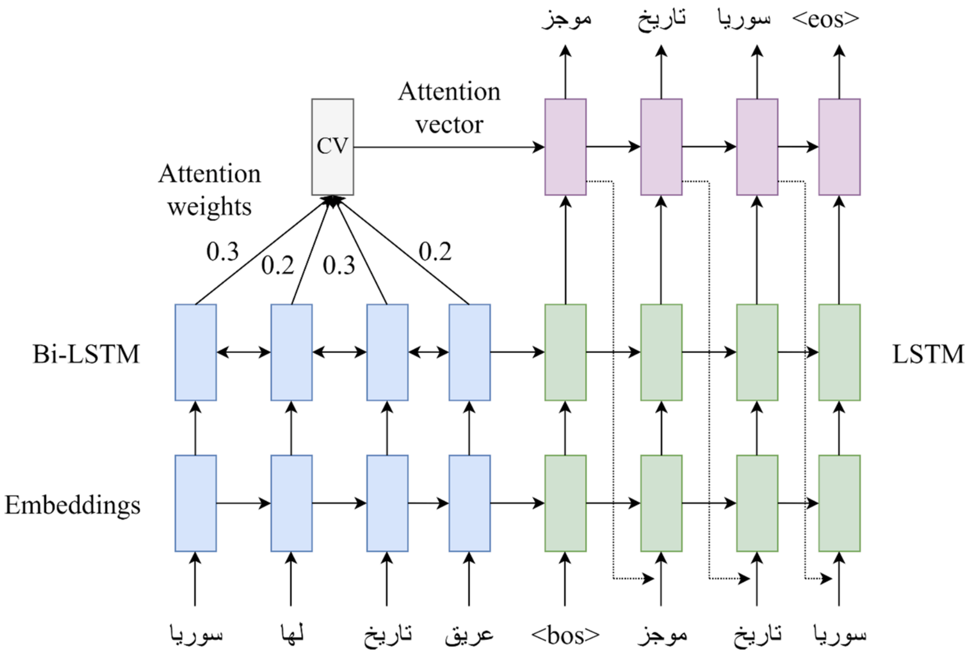

Fig. 5. An illustration of our attentional RNN-based sequence-to-sequence model when summarizing a short example.

LSTM+, we changed the dimensionality of the embeddings to 256 and the number of hidden units at both the encoder and decoder to 512. The model has been trained for ~39k steps using the Adam optimizer with a learning rate of 0.001 and a batch size of 64.

3.2.4. *Transformer+*

Similarly to the way we trained Seq2Seq-LSTM+, we trained Transformer+, which has an identical architecture to the first Transformer, on both the additional and main data. Training has been done for 68k steps with a batch size of 16.

3.2.5. *AraGPT2*

AraGPT2 (Antoun et al., 2021) has a similar architecture to GPT-2 and it was pre-trained as a causal language model on 77GB of Internet text that is mostly written in MSA. It generates a token y_i conditioned on the previous tokens $Y_{0:i-1}$ according to Eq. (6).

$$p_{AraGPT2}(y_i|Y_{0:i-1}) = Softmax(l_i) = Softmax(f_{AraGPT2}(Y_{0:i-1}) \tag{6}$$

AraGPT2 processes the input sequence and passes it to several blocks that use a unidirectional self-attention mechanism, as in the decoder part of the Transformer architecture. The outputs of these blocks are context-dependent representations of the input. The output of the final block is a logit vector l_i that is passed to a softmax function to create a probability distribution of the next token y_i.

When fine-tuning the model, we prepared our examples by appending each summary to the source text as follows:

'\n النص: [text] \n الملخص: \n [summary] <|endoftext|>'.

Where we replaced '[*text*]' with the source text, and '[*summary*]' with the summary.

The fine-tuning was for 100k steps using a batch size of 4. The loss has only been calculated on the summary tokens. The version we used is AraGPT2$_{\text{Base}}$.[5] Fig. 7 visualize the unidirectional self-attention representations of two attention heads (represented with gray and blue colors) of an input example at the 5th block of the fine-tuned model.

3.2.6. *BERT2BERT*

AraBERT is an Arabic version of BERT and is an encoder-only Transformer that takes an input $X_{1:n}$ with length n and returns a context-depended representation of this input with the same length $\bar{X}_{1:n}$. Because both the input and output of AraBERT must have the same length, it cannot be used for text summarization, where the summary has a different length than the source text. To be able to employ AraBERT for summarization, we used the BERT2BERT encoder-decoder architecture where we initialized both the encoder and decoder with AraBERT weights. To build this model, we reused AraBERT parameters in the corresponding layers of BERT2BERT. The encoder part of BERT2BERT is identical to AraBERT so we reused all of AraBERT parameters in the encoder without any change. For the decoder part, we added a cross-attention layer with random initial weights between the self-attention layer and the feed-forward layers at every block. Also, we changed the bidirectional self-attention layers to become unidirectional since the decoder works by

[5] https://huggingface.co/aubmindlab/aragpt2-base

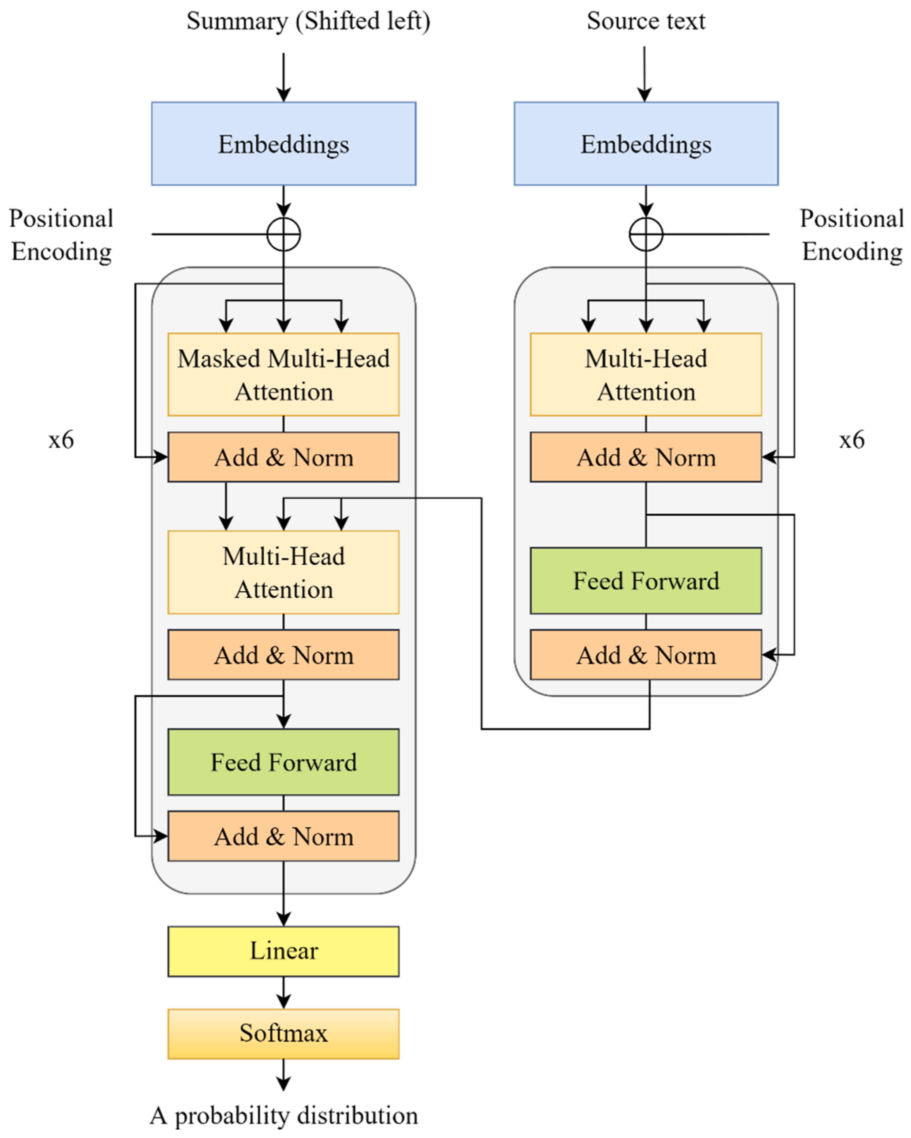

Fig. 6. Our Transformer-based model architecture.

processing only previously generated tokens at each step and cannot process future tokens. An LM head has also been added after the last block in the decoder to allow the generation of summary tokens. We initialized this final layer with the same parameters as the embeddings layer. Finally, we shared the encoder's weights with the decoder in order to reduce the total number of trainable parameters since there is a large similarity between the two. After initialization, only the cross-attention layers at decoder blocks were initialized randomly. We fine-tuned the model for 7000 steps with a batch size of 16. We accumulated gradients at fine-tuning for two steps before back-propagation. The version of AraBERT that has been used is 'bert-base-arabertv02'.[6] Fig. 8 shows how we reused AraBERT parameters (on the left of the arrow) in our model (on the right of the arrow) and the addition of cross attention layers at the decoder.

3.2.7. mBERT2mBERT

We used the BERT2BERT architecture and warm-started it with multilingual BERT,[7] which was pre-trained on 104 languages including Arabic using Wikipedia text. The size of the vocabulary at both the encoder and decoder is 120k tokens. We accumulated gradients for two steps before back-propagation and fine-tuned the model for 17k steps with a batch size of 16. At fine-tuning, the

[6] https://huggingface.co/aubmindlab/bert-base-arabertv02
[7] https://huggingface.co/bert-base-multilingual-cased

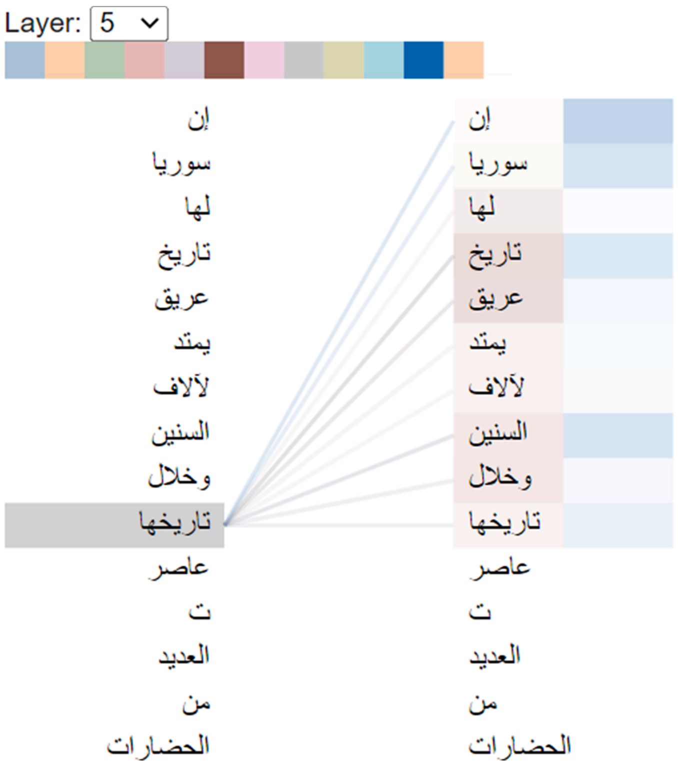

Fig. 7. An illustration of the self-attention representations of two attention heads of an input example at the 5th block of the fine-tuned AraGPT2.

$$\Theta_{\mathrm{AraBERT}} \rightarrow \Theta_{\mathrm{enc}}$$

$$\Theta_{\mathrm{AraBERT}}^{word-embed} \rightarrow \Theta_{\mathrm{dec}}^{lm-head}$$

$$\Theta_{\mathrm{AraBERT}}^{feed-forward} \rightarrow \Theta_{\mathrm{dec}}^{feed-forward}$$

$$\Theta_{\mathrm{AraBERT}}^{self-attention} \rightarrow \Theta_{\mathrm{dec}}^{self-attention}$$

$$+\Theta_{\mathrm{dec}}^{cross-attention}$$

Fig. 8. The initialization of BERT2BERT parameters (on the right of the arrow) with AraBERT parameters (on the left of the arrow).

length of the input was limited to 150 tokens, and the length of the summary to 45 tokens.

3.2.8. AraT5

AraT5, which is an Arabic version of T5 (Raffel et al., 2020), is an encoder-decoder model that reframes all NLP tasks into a unified text-to-text framework and enables using the same model for different tasks such as text summarization, machine translation, and classification. A prefix is added to the input to indicate the task at hand such as 'summarize:' for text summarization and 'translate English to Arabic' for machine translation. The training objective for this model is Masked Language Modeling (MLM) where 15% of the tokens are masked. More than one consecutive tokens are masked with one sentinel token. We used the AraT5$_{\mathrm{Base}}$ version[8] which is based on the T5$_{\mathrm{Base}}$ model architecture and is trained on both MSA and twitter data. The MSA data have a size of 70GB and are

[8] https://huggingface.co/UBC-NLP/AraT5-base

Table 4
The results of the evaluation on the main test set.

	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq-LSTM (baseline)	34.55	17.90	33.04
Transformer	34.11	15.79	32.12
Seq2Seq-LSTM+ (baseline)	39.24	21.52	37.57
Transformer+	41.49	23.50	39.61
mBERT2mBERT	44.88	27.25	42.96
AraGPT2	42.62	25.27	40.48
BERT2BERT	45.96	27.50	44.02
AraT5	**49.06**	**30.81**	**46.87**

extracted from multiple Arabic sources, whereas the twitter data are randomly sampled 1.5B tweets where only the tweets that contain at least three Arabic words are selected. Both the encoder and decoder have a similar architecture to $\text{BERT}_{\text{Base}}$, where they are composed of 12 layers with 12 attention heads.

The model was fine-tuned for 66k steps with a batch size of 8. At fine-tuning, the length of the input has been limited to 128 tokens, and the length of the summary to 40 tokens.

Our experiments have been done using Python v3.8, TensorFlow v2.8.2, and HuggingFace's Transformers v4.5.1.

4. Results and discussion

The proposed models have been evaluated on both the main and out-of-domain test sets. We did both an automatic evaluation using the ROUGE metrics, which are often used for evaluating text summarization models, and a manual evaluation since the ROUGE metrics do not give a complete and accurate assessment of the quality and readability of the generated summaries.

4.1. Automatic evaluation

We used ROUGE-1, ROUGE-2, and ROUGE-L for automatically evaluating our models, which refer to unigram, bigram, and longest common subsequence overlaps between the generated and reference summaries respectively. The precision, recall, and F-measure values are calculated for each ROUGE metric as follows:

$$\begin{aligned} precision &= \frac{\left|grams_{reference} \cap grams_{generated}\right|}{grams_{generated}} \\ recall &= \frac{\left|grams_{reference} \cap grams_{generated}\right|}{grams_{reference}} \\ F-measure &= 2.0 * \frac{recall * precision}{recall + precision} \end{aligned} \tag{7}$$

We reported the evaluation results using ROUGE F1 scores. The evaluation was done using the rouge Python library.[9] The beam search algorithm has been used when generating the summaries with a beam size of 3.

In our experiments, Seq2Seq-LSTM and Seq2Seq-LSTM+ are the baselines that were implemented by previous studies (Suleiman & Awajan, 2020; Wazery et al., 2022) and that we use to measure the improvement in performance obtained by our different models.

Table 4 shows the ROUGE F1 scores when evaluating our models on the main test set. Fig. 9 displays a plot of the scores.

We notice that the AraT5 model has achieved the best performance among all the other models according to ROUGE scores, outperforming BERT2BERT by ~3 ROUGE, which has also performed competitively. mBERT2mBERT has scored better than AraGPT2 by ~2 ROUGE even if the former is a multilingual model and the latter is a monolingual model that was trained specifically for Arabic. Also, all of AraT5, BERT2BERT, AraGPT2, and mBERT2mBERT have scored better than Transformer+ and Seq2Seq-LSTM+ even if they were fine-tuned with ~79% less data. Furthermore, we notice the significant increase in performance achieved by both Seq2Seq-LSTM+ and Transformer+ compared to Seq2Seq-LSTM and Transformer, agreeing with the common observation that deep learning models are data-hungry and require huge amounts of data to achieve competitive performance. The table also shows that Transformer+ has scored better than Seq2Seq-LSTM+ by ~2 ROUGE, while on the other hand, Transformer has scored less than Seq2Seq-LSTM.

We also tested our models on the out-of-domain test set that consists of 652 examples and relates to the topic of "elections". Table 5 and Fig. 10 show the ROUGE F1 scores when evaluating on this set.

Seq2Seq-LSTM+ and Transformer+ were not evaluated on the out-of-domain test set because they were trained using the additional data which were not included when sampling out the set. Generally, all the models scored less on the out-of-domain test set than on the main test set. The lowest drop in performance was by AraGPT2 and AraT5, which dropped by 7.39 ROUGE-1 and 7.85 ROUGE-1 respectively. Also, both mBERT2mBERT and BERT2BERT had a large drop in their scores, where they dropped by 10.78 ROUGE-1 and

[9] https://pypi.org/project/rouge

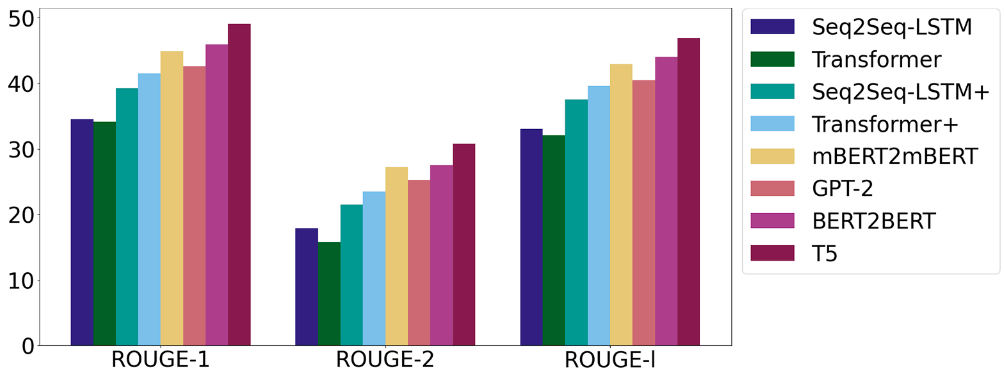

Fig. 9. A plot of the ROUGE F1 evaluation scores on the test set.

Table 5
The results of the evaluation on the out-of-domain test set.

	ROUGE-1	ROUGE-2	ROUGE-L
Seq2Seq-LSTM (baseline)	22.28	8.18	21.13
Transformer	22.45	7.48	20.90
mBERT2mBERT	34.1	17.69	32.63
AraGPT2	35.23	18.06	33.59
BERT2BERT	36.04	18.51	34.57
AraT5	**41.21**	**23.68**	**39.35**

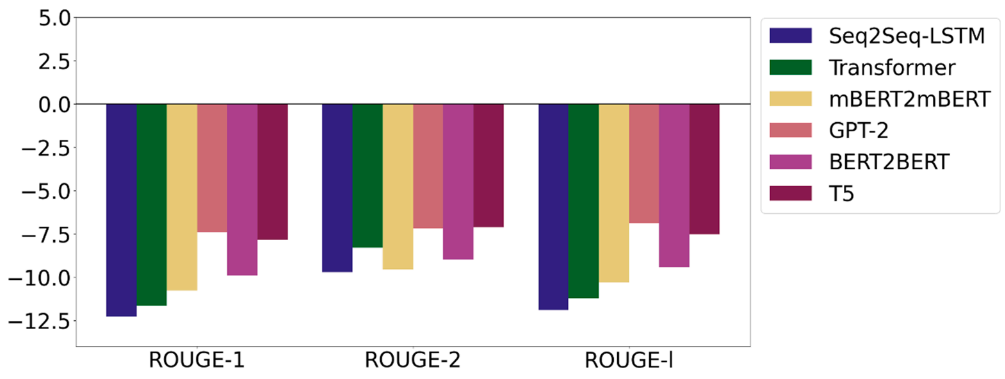

Fig. 10. A plot of the drop in performance measured by ROUGE F1 scores when evaluating on the out-of-domain set.

9.92 ROUGE-1 respectively. The Seq2Seq-LSTM and Transformer models had the largest drop in performance, where the score of the former dropped by 12.27 ROUGE-1 and the score of the latter dropped by 11.66 ROUGE-1. We conclude according to the results that AraGPT2 and AraT5 give a better output when dealing with out-of-domain data than the other tested models, whereas our Seq2Seq-LSTM and Transformer give generally worse output.

4.2. Manual evaluation

Because the ROUGE metrics do not usually give a complete and accurate assessment of the quality and readability of the generated summaries as we mentioned earlier, we did a manual human evaluation. We randomly sampled 20 examples from the test set and generated their summaries using our models, then we asked three native Arabic speakers to give a score between 1 and 5 for each generated summary on the following two aspects: (1) quality: a measure of the degree to which the generated summary is capturing the main ideas in the source text, and (2) readability: an assessment of the grammatical correctness and sentence structure of the summary (Table 6). We calculated the average of the reported scores.

Table 7 and Fig. 11 show the manual human evaluation scores.

We find that AraT5 and BERT2BERT had the best scores on the two aspects with a slight difference between them. AraGPT2 scored high on the readability aspect but had a relatively smaller score on the quality aspect. Also, mBERT2mBERT had relatively low scores on both the readability and quality criteria even if it had scored high on the ROUGE metrics. In addition, Transformer+ scored a

Table 6
Description of the Quality and Readability criteria for manual human evaluation.

Score	Quality	Readability
1	Output is out of topic	Inconsistent / Difficult to read
2	Main ideas are slightly captured	Slightly readable
3	Main ideas are moderately captured	Readable with weak Arabic
4	Main ideas are largely captured	Readable with acceptable Arabic
5	Main ideas are completely captured	Readable with fluent Arabic

Table 7
The manual human evaluation scores.

	Quality	Readability
Seq2Seq-LSTM+ (baseline)	2.95	3.5
Transformer+	3.35	4.05
mBERT2mBERT	3.2	3.8
AraGPT2	3.85	4.3
BERT2BERT	4.2	**4.5**
AraT5	**4.3**	4.45

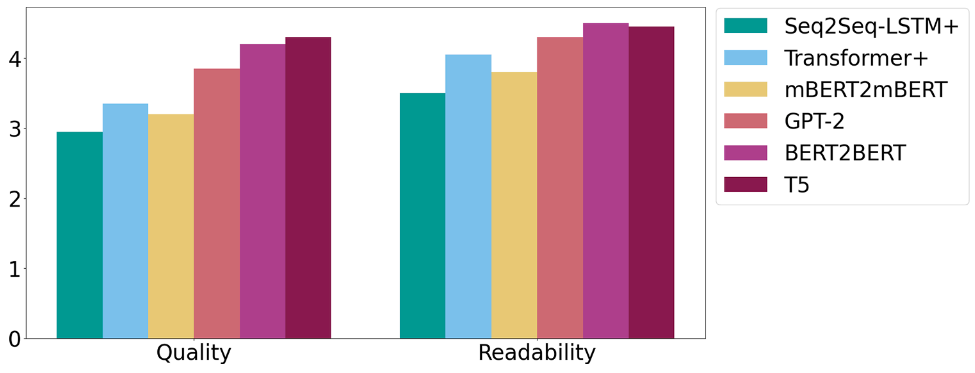

Fig. 11. A plot of the manual human evaluation scores.

relatively low value on the quality aspect but had an acceptable readability score. Seq2Seq-LSTM+ scored the worst on both criteria among all the other models.

We summarized two documents using our models and showed the produced summaries in Table 10 and Table 11.

4.3. Comparison with previous studies

Table 8 summarizes different abstractive text summarization datasets for both English and Arabic which are comparable to our dataset. We listed the number of examples in each split, document length (DL) which refers to the average number of words in the document, summary length (SL) which refers to the average number of words in the summary, and the language of the dataset.

The authors of some previous works such as (Kahla et al., 2021) and (Suleiman & Awajan, 2020) did not release their dataset publicly, thus a direct comparison with their results is not possible.

We did a comprehensive evaluation of the dataset that was published by Al-Maleh and Desouki, (2020) where we randomly selected 200 examples and evaluated them manually with regard to the quality of their summaries. We found that more than 50% of the examples had low or medium quality. In addition, the dataset suffers from a high degree of repetition where many examples are repeated several times with slightly different summaries and this could lead to biased results if the models were tested on examples that were already seen during training.

The closest dataset to ours with regard to the average number of words at the source text and summary is the Gigaword dataset. Although a direct comparison is not possible, in Table 9 we compare ROUGE F1 scores between our AraT5 model, the pointer-generator model of (Al-Maleh & Desouki, 2020), and the BART-RXF model that achieved state-of-the-art results on the Gigaword dataset (Aghajanyan et al., 2020).

We did not compare our models with the work of (Nagoudi et al., 2022) that fine-tuned AraT5 on the WikiLingua dataset. This is because our dataset consists of single-sentence summaries which is different from WikiLingua which consists of multi-sentence summaries.

Table 8
A comparison between different abstractive text summarization datasets for both English and Arabic.

Dataset	Number of examples			DL	SL	Language
	Train	Dev	Test			
DUC 2004	–	–	500	35.6	10.4	English
X-Sum	204,045	11,332	11,334	431	23	English
Gigaword	3.8M	189K	1951	31.4	8.3	English
CNN / DM	286,817	13,368	11,487	781	56	English
WikiLingua	23.4k	2.9K	2.9K	391	39	Arabic
Kahla et al.	19,807	–	1701	412	35.1	Arabic
Al-Maleh	235,870	38,968	20,000	80.6	3.3	Arabic
Our dataset	75,702	4205	4205	38.97	9.43	Arabic

Table 9
A comparison between the ROUGE F1 scores of different state-of-the-art models.

	ROUGE-1	ROUGE-2	ROUGE-L
BART-RXF	40.45	20.69	36.56
Al-Maleh and Desouki	44.23	–	–
AraT5 (Ours)	**49.06**	**30.81**	**46.87**

Table 10
A text example summarized using our models.

Source text	شهدت مدينة طرابلس، مساء أمس الأربعاء، احتجاجات شعبية وأعمال شغب لليوم الثالث ع لى التوالي، وذلك بسبب تردي الوضع المعيشي والاقتصادي. واندلعت مواجهات عنيفة وعم ليات كر وفر ما بين الجيش اللبناني والمحتجين استمرت لساعات، إثر محاولة فتح الطر ات المقطوعة، ما أدى إلى إصابة العشرات من الطرفين. وأقدم عدد من المحتجين على ر شق عناصر الجيش اللبناني بالحجارة والمفرقعات النارية، بالمقابل أطلق الجيش الق نابل المسيلة للدموع لتفريق المحتجين. **The city of Tripoli witnessed popular protests and riots yesterday, on Wednesday evening, for the third consecutive day, due to the deteriorating living and economic situation. Violent confrontations and hit-and-run operations erupted between the Lebanese army and the protesters that lasted for hours, after an attempt to open the blocked roads, which led to dozens of injuries on both sides. A number of protesters threw stones and firecrackers at the Lebanese army, while the army fired tear gas to disperse the protesters.**
AraT5	مواجهات عنيفة بين الجيش اللبناني ومحتجين في طرابلس Violent clashes between the Lebanese army and protesters in Tripoli
BERT2BERT	اشتباكات بين الجيش اللبناني والمحتجين في طرابلس لليوم الثالث على التوالي Clashes between the Lebanese army and protesters in Tripoli for the third consecutive day
AraGPT2	تجدد الاحتجاجات في طرابلس Renewed protests in Tripoli
mBERT2mBERT	احتجاجات في طرابلس بعد مواجهات مع الجيش اللبناني Protests in Tripoli after clashes with the Lebanese army
Transformer+	اشتباكات بين الجيش اللبناني ومحتجين في طرابلس Clashes between the Lebanese army and protesters in Tripoli
Seq2Seq-LSTM+	تجدد الاحتجاجات المناهضة للحكومة بطرابلس Renewed anti-government protests in Tripoli

4.4. Conclusions and recommendations

We present the following conclusions and recommendations based on the results of our experiments:

(1) Leveraging AraT5, which is a pre-trained encoder-decoder Transformer, gives better performance according to ROUGE metrics than fine-tuning a BERT2BERT-based model that is initialized with an Arabic BERT such as AraBERT (the improvement is ~3 ROUGE in our experiments).
(2) Both of AraT5 and the BERT2BERT-based model that is initialized with AraBERT weights perform better than AraGPT2 on text summarization by a clear margin. This may be due to the fact that AraGPT2 uses unidirectional self-attention layers which is suboptimal when encoding the meaning of a source text.
(3) AraT5 and AraGPT2 are better at summarizing out-of-domain data than a BERTBERT-based model that is initialized with AraBERT, whereas RNN-based and Transformer-based sequence-to-sequence models that were trained from scratch are noticeably worse.
(4) The BERT2BERT-based model that is initialized with multilingual BERT yields relatively poor scores when evaluated manually, even if it gives high scores on the ROUGE metrics.

Table 11
A second text example summarized using our models.

Source text	يجري علماء في بريطانيا تجربة لاختبار فعالية عقار إيبوبروفين لمساعدة المصابين بفيروس كورونا. وذكرت هيئة الإذاعة البريطانية "بي بي سي" أن فريق مشترك من أطباء مستشفيات "جاي" و"سانت توماس" و"كينغز كوليدج" في لندن يعتقد أن إيبوبروفين، هو مضاد للالتهابات ومسكن للألم، يمكن أن يعالج صعوبات التنفس. ويأمل العلماء أن يساعد هذا العلاج المنخفض التكلفة المرضى في الاستغناء عن أجهزة التنفس الصناعي. **Scientists in Britain are conducting an experiment to test the effectiveness of the drug ibuprofen to help people infected with the Corona virus. The British Broadcasting Corporation "BBC" reported that a joint team of doctors from "Jay", "St Thomas" and "King's College" hospitals in London believed that ibuprofen, an anti-inflammatory and pain reliever, could treat breathing difficulties. Scientists hope this low-cost treatment will help patients dispensing ventilators.**
AraT5	علماء يختبرون فعالية عقار " إيبوبروفين " لعلاج مرضى كورونا Scientists are testing the effectiveness of “ibuprofen” to treat coronavirus patients
BERT2BERT	عقار " إيبوبروفين " قد يساعد مرضى " كورونا " على التغلب على صعوبات التنفس Ibuprofen may help coronavirus patients overcome breathing difficulties
AraGPT2	" إيبوبروفين" ينجح في علاج مرضى "كورونا " “Ibuprofen” succeeds in treating “coronavirus” patients
mBERT2mBERT	"خبراء بريطانيون يحذرون من عقار إيبوبروفين لعلاج "كورونا British experts warn of ibuprofen to treat “coronavirus”
Transformer+	تجربة فعالية عقار إيبوبروفين لمساعدة المصابين بكورونا Experimenting with the effectiveness of ibuprofen to help people infected with coronavirus
Seq2Seq-LSTM+	إيبوبروفين يخفف صعوبات التنفس عند المرضى Ibuprofen relieves breathing difficulties in patients

(5) AraT5 and the BERT2BERT-based model that is initialized with AraBERT produce summaries with the highest readability and quality scores according to human evaluation among our other proposed models.
(6) AraGPT2 gives summaries with high readability but with relatively lesser quality when compared with the other proposed models.
(7) When trained from scratch on a large dataset, a Transformer-based model performs better than an RNN-based sequence-to-sequence model according to both ROUGE metrics and human evaluation.
(8) Fine-tuning pre-trained large models outperform training RNN-based and Transformer-based sequence-to-sequence models from scratch with significantly less data (in our case we obtained better scores with 79% less data).

4.5. Limitations

Our work focused on generating single-sentence summaries from news sources and more research could be conducted on generating multi-sentence summaries and working with different text sources. Also, our models where trained on Modern Standard Arabic (MSA) and it will perform badly on dialectic Arabic. For summarizing texts in Arabic dialects, we suggest leveraging MARBERT (Abdul-Mageed et al., 2021) that was pre-trained on 1B tweets in both MSA and Arabic dialects.

Lastly, it is known that summarization models often generate unreliable summaries (Pagnoni et al., 2021). In this research, we did not investigate which model architecture leads to more factual and faithful summaries, which we believe to be an important area of research that we leave for a future work.

5. Conclusion

In this paper, we leveraged multiple architectures and pre-trained language models for Arabic abstractive text summarization including RNN-based and Transformer-based ones. We also used a BERT2BERT-based encoder-decoder architecture for fine-tuning encoder-only Transformer models. A novel dataset of text-summary pairs has also been built to train our models. We evaluated the models using both ROUGE metrics and manual evaluation and tested their performance on an out-of-domain data set. Our work shows that the Transformer architecture and the pre-trained Transformer-based language models yield a remarkable performance boost on Arabic text summarization. We showed that the fine-tuned AraT5 achieves the best performance. Our future direction is to summarize Arabic text using a two-stage approach, where the first is extractive and the second is abstractive, which we think is a promising next step that will help us to summarize much longer documents.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Mohammad Bani-Almarjeh: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Mohamad-Bassam Kurdy:** Conceptualization, Validation, Writing – review & editing, Supervision, Project administration.

Declaration of Competing Interest

None.

Data availability

We have added a link to the data in the manuscript

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ipm.2022.103227.

References

Abdelali, A., Darwish, K., Durrani, N., & Mubarak, H. (2016). Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations* (pp. 11–16). https://doi.org/10.18653/v1/n16-3003

Abdul-Mageed, M., Elmadany, A. R., & Nagoudi, E. M. B. (2021). ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* (pp. 7088–7105). https://doi.org/10.18653/v1/2021.acl-long.551

Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., & Gupta, S. (2020). Better fine-tuning by reducing representational collapse. arXiv:2008.03156.

Alahmadi, D., Wali, A., & Alzahrani, S. (2022). TAAM: Topic-aware abstractive arabic text summarisation using deep recurrent neural networks. *Journal of King Saud University - Computer and Information Sciences, 34*, 2651–2665. https://doi.org/10.1016/j.jksuci.2022.03.026

Alami, N., En-nahnahi, N., Ouatik, S. A., & Meknassi, M. (2018). Using Unsupervised Deep Learning for Automatic Summarization of Arabic Documents. *Arabian Journal for Science and Engineering, 43*, 7803–7815. https://doi.org/10.1007/s13369-018-3198-y

Alami, N., Meknassi, M., & En-nahnahi, N. (2019). Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. *Expert Systems with Applications, 123*, 195–211. https://doi.org/10.1016/j.eswa.2019.01.037

Al-Maleh, M., & Desouki, S. (2020). Arabic text summarization using deep learning approach. *Journal of Big Data, 7*. https://doi.org/10.1186/s40537-020-00386-7

Alotaiby, F., Alkharashi, I., & Foda, S. (2009). Processing large Arabic text corpora: Preliminary analysis and results. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools* (pp. 78–82).

Alotaiby, F., Foda, S., & Alkharashi, I. (2010). Clitics in Arabic language: A statistical study. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation* (pp. 595–601).

Antoun, W., Baly, F., & Hajj, H. (2020). AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection* (pp. 9–15).

Antoun, W., Baly, F., & Hajj, H. (2021). AraGPT2: Pre-trained transformer for Arabic language generation. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop* (pp. 196–207).

Bani Almarjeh, M. (2022). "SumArabic". *Mendeley Data, V1*. https://doi.org/10.17632/7kr75c9h24.1 [Dataset].

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & Amodei, D. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (pp. 1877–1901).

Chopra, S., Auli, M., & Rush, A. M. (2016). Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 93–98).

Devlin, J., Chang, M. W., Lee, K., & 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* (pp. 4171–4186).

Graff, D. (2007). Arabic Gigaword Third Edition - Linguistic Data Consortium [WWW Document]. URL https://catalog.ldc.upenn.edu/LDC2007T40 (accessed 6.25.22).

Graff, D., Kong, J., Chen, K., & Maeda, K. (2007). English Gigaword Third Edition - Linguistic Data Consortium [WWW Document]. URL https://catalog.ldc.upenn.edu/LDC2007T07 (accessed 6.25.22).

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., & Zhu, J. (2021). Pre-trained models: Past, present and future. *AI Open, 2*, 225–250. https://doi.org/10.1016/j.aiopen.2021.08.002

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in: A field guide to dynamical recurrent networks. 10.1109/9780470544037.ch14.

Kahla, M., Yang, Z. G., & Novák, A. (2021). Cross-lingual fine-tuning for abstractive Arabic text summarization. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (pp. 655–663). https://doi.org/10.26615/978-954-452-072-4_074

Khandelwal, U., Clark, K., Jurafsky, D., & Kaiser, L. (2019). *Sample efficient text summarization using a single pre-trained transformer.* CoRR. abs/1905.08836.

Ladhak, F., Durmus, E., Cardie, C., & McKeown, K. (2020). WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 4034–4048). https://doi.org/10.18653/v1/2020.findings-emnlp.360

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 7871–7880). https://doi.org/10.18653/v1/2020.acl-main.703

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, Ł., et al. (2018). Generating wikipedia by summarizing long sequences. In *Proceedings of the 6th International Conference on Learning Representations* (pp. 1–18).

Liu, Y., & Lapata, M. (2019). Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (pp. 3730–3740). https://doi.org/10.18653/v1/d19-1387

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., & Stoyanov, V. (2019). *RoBERTa: A robustly optimized BERT pretraining approach.* CoRR. abs/1907.11692.

Nagoudi, E. M. B., Elmadany, A., & Abdul-Mageed, M. (2022). AraT5: Text-to-text transformers for Arabic language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics* (pp. 628–647). https://doi.org/10.18653/v1/2022.acl-long.47

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning* (pp. 280–290). https://doi.org/10.18653/v1/k16-1028

Pagnoni, A., Balachandran, V., & Tsvetkov, Y. (2021). Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4812–4829). https://doi.org/10.18653/v1/2021.naacl-main.383

Pöttker, H. (2003). News and its communicative quality: The inverted pyramid—When and why did it appear? *Journal of Studies, 4*, 501–511. https://doi.org/10.1080/1461670032000136596

Qi, W., Gong, Y., Yan, Y., Xu, C., Yao, B., Zhou, B., & Duan, N. (2021). ProphetNet-X: Large-scale pre-training models for English, Chinese, multi-lingual, dialog, and code generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations* (pp. 232–239). https://doi.org/10.18653/v1/2021.acl-demo.28

Qiu, X. P., Sun, T. X., Xu, Y. G., Shao, Y. F., Dai, N., & Huang, X. J. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Studies, 63*, 1872–1897. https://doi.org/10.1007/S11431-020-1647-3

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog, 1*, 9.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research, 21*, 1–67.

Rothe, S., Narayan, S., & Severyn, A. (2020). Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics, 8*, 264–280. https://doi.org/10.1162/tacl_a_00313

Ruder, S. (2019). *Neural transfer learning for natural language processing.* NUI Galway.

Rush, A. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 379–389).

Straka, M., Mediankin, N., Kocmi, T., Žabokrtský, Z., Hudeček, V., & Hajič, J. (2018). Sumeczech: Large Czech news-based summarization dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (pp. 3488–3495).

Suleiman, D., & Awajan, A. (2020). Deep learning based abstractive arabic text summarization using two layers encoder and one layer decoder. *Journal of Theoretical and Applied Information Technology, 98*, 3233–3244.

Suleiman, D., & Awajan, A. (2022). Multilayer encoder and single-layer decoder for abstractive Arabic text summarization. *Knowledge-Based System, 237*, Article 107791. https://doi.org/10.1016/J.KNOSYS.2021.107791

Tang, Y., Tran, C., Li, X., Chen, P.-J., Goyal, N., Chaudhary, V., & Fan, A. (2020). *Multilingual translation with extensible multilingual pretraining and finetuning.* CoRR. abs/2008.00401.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N.…P.olosukhin, I. (2017). Attention is all you need, in: Advances in neural information processing systems 30. 10.1109/2943.974352.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks, in: Advances in neural information processing systems. pp. 1–9.

Wazery, Y. M., Saleh, M. E., Alharbi, A., & Ali, A. A. (2022). Abstractive Arabic text summarization based on deep learning. *Computational Intelligence and Neuroscience, 2022*. https://doi.org/10.1155/2022/1566890

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., & Raffel, C. (2021). mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 483–498). https://doi.org/10.18653/v1/2021.naacl-main.41

Yang, M., Wang, X., Lu, Y., Lv, J., Shen, Y., & Li, C. (2020). Plausibility-promoting generative adversarial network for abstractive text summarization with multi-task constraint. *Information Sciences (Ny)., 521*, 46–61. https://doi.org/10.1016/j.ins.2020.02.040

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). PEGASUS: Pre-Training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*.