Journal of Big Data

RESEARCH Open Access

Arabic text summarization using deep learning approach

Molham Al-Maleh[1*] and Said Desouki[2]

*Correspondence: molhamaleh@gmail.com
[1] Faculty of Information Technology, Higher Institute for Applied Sciences and Technology, Damascus, Syria
Full list of author information is available at the end of the article

Abstract

Natural language processing has witnessed remarkable progress with the advent of deep learning techniques. Text summarization, along other tasks like text translation and sentiment analysis, used deep neural network models to enhance results. The new methods of text summarization are subject to a sequence-to-sequence framework of encoder–decoder model, which is composed of neural networks trained jointly on both input and output. Deep neural networks take advantage of big datasets to improve their results. These networks are supported by the attention mechanism, which can deal with long texts more efficiently by identifying focus points in the text. They are also supported by the copy mechanism that allows the model to copy words from the source to the summary directly. In this research, we are re-implementing the basic summarization model that applies the sequence-to-sequence framework on the Arabic language, which has not witnessed the employment of this model in the text summarization before. Initially, we build an Arabic data set of summarized article headlines. This data set consists of approximately 300 thousand entries, each consisting of an article introduction and the headline corresponding to this introduction. We then apply baseline summarization models to the previous data set and compare the results using the ROUGE scale.

Keywords: Natural language processing, Text summarization, Deep learning, Big data, Sequence-to-sequence framework

Introduction

The task of text summarization is one of the most important challenges that faces computer capabilities with all its new advances. This task is based on generating short text from longer text so that the short text contains the most important info of the original text. There are two basic methodologies used to summarize the texts, which are extractive summarization—from which most systems with good results came out—and abstractive summarization that simulate human summarization. The first methodology is based on determining the important parts of the text in a statistical approach like the work of Belkebir et al. [1], or in a semantic approach like the work of Imam et al. [2] on the Arabic language; then, it represents the summary by truncating these parts and linking them like what was done by Knight et al. [3] on the English language. The second methodology is based on simulating human work in summarizing, which is based on

Springer Open

expressing the basic meaning of the text in a new linguistic style and in different words; it involves more sophisticated processes, such as paraphrasing, generalization, and reordering [4]. Previous studies have begun to generate abstract summaries either using linguistically inspired constraints [5, 6] or with syntactic transformation of the input text [7, 8].

In this work, we use a data-driven model to generate headlines for Arabic articles in a manner similar to the successful approach achieved by machine translation based on neural networks, which was also adopted by the new studies in generating headlines for English articles [9]. Recently, deep learning methods have made clear progress in the area of English text summarization, if not ideal, based on neural network models using a sequence-to-sequence framework. These models consist of two complementary units which are trained jointly through a gradient descent or reinforcement learning. The first unit is an encoder that generates a hidden representation of the original text, while the second unit is a decoder that generates the summarized text. Summarized text words are generated word-by-word until a special stopping character is generated that ends the summary. In addition, in the late days of its conclusion, abstractive text summarization models, based on neural networks, worked on merging the abstractive and extractive approaches using the pointer-generator approach. That approach added the capability of copying words from the source file directly to the summary [10, 11]. The word 'pointer' from the approach name indicates the extraction methodology. Whereas, the word 'generator' indicates the possibility of generating a new word in the summary, according to the abstraction methodology. Our study generates summarizations using the abstractive neural model as a baseline model. Our baseline model depends on the attention mechanism presented by Bahdanau et al. [12], in order to determine the parts of the original text that must be focused on while generating each new word from the summary. The decoder uses the beam search algorithm to truncate the size of probabilities when generating summarization tokens from the abstractive section. We improved the baseline model by adding the copy mechanism presented by See et al. [11], to end up with a pointer-generator model.

We explain the steps for processing in more detail in "Proposed methodology" section. This approach of summarization, known as Attention Based Summarization (ABS), incorporates less linguistic structure than comparable abstractive summarization approaches, but can expand easily to train on a huge amount of data; it is capable of training on any article-headline pairs. Based on this availability, we have trained our system to generate headlines for articles after building a new data set in Arabic consisting of about 300 thousand pairs. The original text in each pair is the introduction to the article while the corresponding summary is the headline of the article. The method for building the data set is explained in more details in "Data set" section under "Experiments". An example of generating summary is presented in Table 1, and we mentioned "Training details" under "Experiments" section. To examine the efficiency of this approach in the Arabic language, we calculated the ROUGE scale shown in Table 3. The results of this study and the set of data relied upon it, are the first of its kind in the Arabic language in the field of summarizing articles' headlines based on deep learning techniques. The contributions in our research specifically (1) the newly created Arabic dataset with 300 thousand pairs of (article, headline) and (2) the comparisons between Arabic and

Table 1 Comparison between the results of the baseline model and the pointer-generator model

	Origin	Translation
Article	أيد سبحانه و تعالى نبيه **محمد** عليه الصلاة و السلام بعدد من المعجزات ؛ حتى تكون دليلا على صدقه و من معجزاته ما يأتي : (...)	God Almighty gave a number of miracles to his prophet Muhammad, peace and blessings be upon him, so that it will be a proof of his sincerity, and part of his miracles are the following (…)
Baseline model summarization	معجزات **الرسل**	Prophets' miracles
Pointer-generator summarization	معجزات **الرسول** عليه الصلاة و السلام.	Prophet's miracles peace and blessings be upon him
Reference summarization	معجزات محمد عليه الصلاة و السلام.	Muhammad's miracles peace and blessings be upon him

Words in bold are new generated words, while underlined words are copied from the source

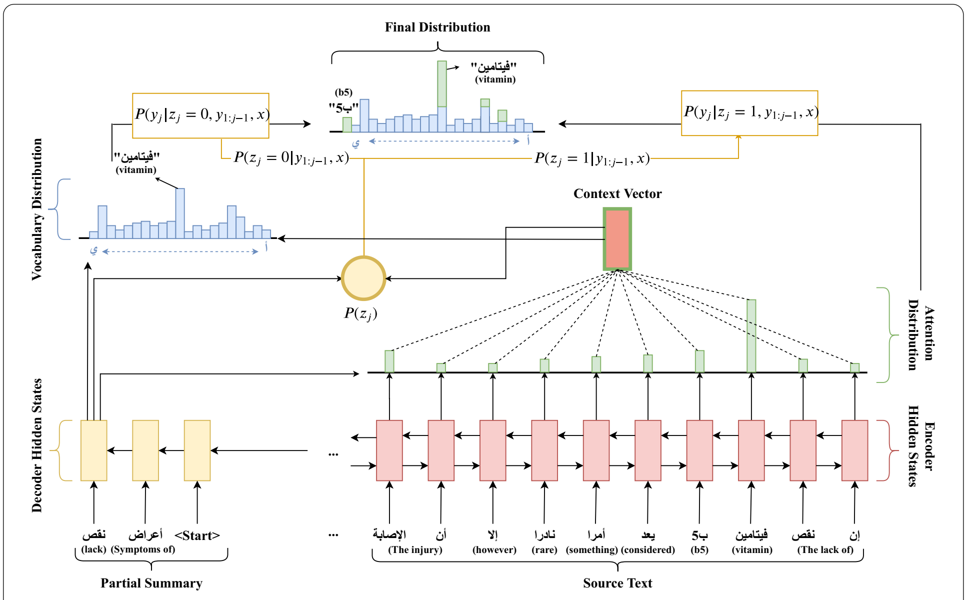

Fig. 1 Sequence-to-sequence model

English results, may contribute to the emergence of future comparisons as in the Document Understanding Conferences (DUC) for the English language.

Related work

A vast majority of past work in summarization has been extractive, which consists of identifying key sentences or passages in the source document and reproducing them as summary [13–17]. Humans on the other hand, tend to paraphrase the original story in their own words. As such, human summaries are abstractive in nature and seldom consist of reproduction of original sentences from the document. Some of the abstractive summarization researches used machine learning methodologies which Sarker et al. [18] made a brief summarization over them. One of the important contributions in Arabic text summarization based on machine learning is by Sobh et al. [19]. On the other hand, with the emergence of deep learning as a viabe alternative for many NLP tasks, researchers have started considering this framework as an attractive, fully data-driven alternative to abstractive summarization. Consequently a new approach emerged which is the neural abstractive summarization with sequence-to-sequence models [12, 20]. This approach has been applied to tasks such as headline generation [9] and article summarization [21]. Chopra et al. [22] show that attention approaches that are more specific to summarization can further improve the performance of models. Gu et al. [10] were the first to show that a copy mechanism, introduced by Vinyals et al. [23], can combine the advantages of both extractive and abstractive summarization by copying words from the source. See et al. [11] refine this pointer-generator approach and use an additional coverage mechanism [24] that makes a model aware of its attention history to prevent repeated attention. While contributions to the Arabic language using sequence-to-sequence deep learning framework are still limited, Elmadani et al. [25] showcased how the fine-tuned pretrained BERT model [26] can be applied to the Arabic language to both construct the first documented model for abstractive Arabic text summarization, and showed its performance in Arabic extractive summarization. In similar research domains, Helmy et al. [27] proposed a deep learning based approach for Arabic keyphrase extraction. It achieves better performance compared to the related competitive approaches. It also introduce the community with an annotated large-scale dataset of about 6000 scientific abstracts which can be used for training, validating and evaluating deep learning approaches for Arabic keyphrase extraction. Our work used the same framework as See et al. [11], where we use pointer-generator approach, but we go beyond the standard architecture and use coverage and length penalties too (Fig. 1). We also propose a novel dataset for Arabic headline summarization on which we establish benchmark numbers too.

Background

We describe the standard approach for supervised abstractive summarization learning based on the attentive sequence-to-sequence framework, and the challenges it faces in text representation and generation. The goal of a model under this framework is to maximize the probability of generating correct target sequences.

Sequence-to-sequence framework

The sequence-to-sequence framework consists of two parts: a neural network for the encoder and another network for the decoder. The source text, reference summary data is tokenized and fed to the encoder and decoder networks respectively during training. The encoder network reads the source text and transforms it into a potentially useful vector representation, which then passes to the decoder network to help in the prediction of the summary sequence on a token per token basis. Figure 2 illustrates how the encoder and decoder networks work together.

Encoder mechanism

The encoder mechanism uses a deep neural network to convert a sequence of source words into a sequence of vectors representing its contextual meaning. This encoding is done using recurrent, convolutional or transformer neural networks.

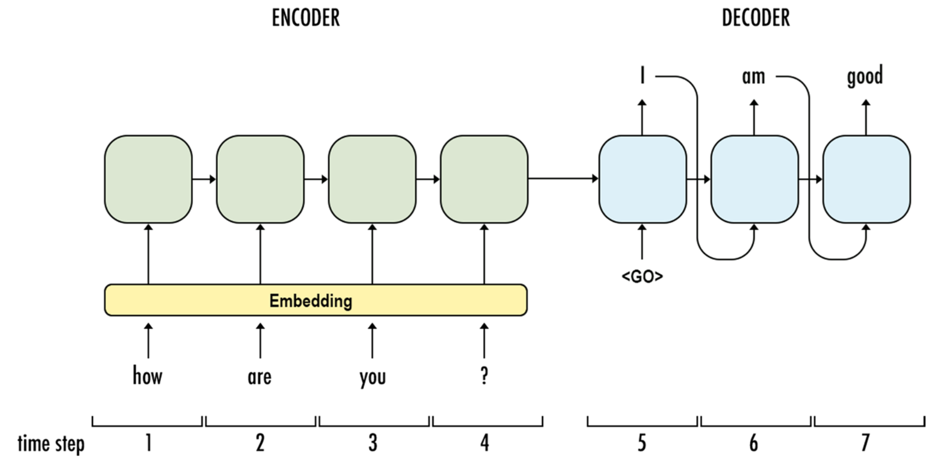

Fig. 2 Beam search decoding

Decoder mechanism

The decoder network uses the vector representation coming out of the encoder network and its own internal state information to represent the state of the sequence generated so far. Essentially, the decoder mechanism combines specific vectorial knowledge about the relevant context with general knowledge about language generation in order to produce the output sequence.

Attention mechanism

A mapping of the decoder state at each time step with all the encoder states into an attention vector, helps produce a context vector which is a weighted sum of the encoder states. Incorporating this context vector at each decoding time step helps improve text generation [12].

Necessity for attention

From a cognitive science perspective, attention, defined as the ability to focus on one thing and ignore others, allows for picking out salient information from noisy data and to remember one event rather than all events. Thus, attention is selective and appears to be as useful for deep learning as it is for people. From a sequence-to-sequence standpoint, attention is the action of focusing on specific parts of the input sequence. It can be stochastic and trained with reinforcement learning (hard attention) or differentiable and trained with back-propagation (soft attention). We note that attention changes over time.

As the model generates each word, its attention changes to reflect the relevant parts of the input.

Self-attention

When a sequence-to-sequence model is trying to generate the next word in the summary, this word is usually describing only a part of the input text. Using the whole representation of the input text (h) to condition the generation of each word cannot efficiently produce different words for different parts of the input. But, if we first divide the input into n parts, we can compute representations of each part $(h_1 \ldots h_n)$. Then, when the model is generating a new word, its attention mechanism can focus on the relevant part of the input sequence, so that the model can only use specific parts of the input.

Text generation

Greedy decoding

When using greedy decoding, the model at any time step has only one single hypothesis. Since a text sequence can be the most probable despite including tokens that are not the most probable at each time step, greedy decoding is seldom used in practice.

Beam decoding

When using beam search decoding the model iteratively expands each hypothesis one token at a time and in the end of each iteration, it only keeps the beam-size best ones as shown in Fig. 3. Small beam sizes are able to yield good results in terms of ROUGE score while larger beam sizes can yield worse results. To make decoding efficient the decoder expands only hypotheses that look promising. Bad hypotheses should be pruned early to avoid wasting time on them, but pruning compromises optimality.

Proposed methodology

We used the encoder–decoder framework to generate an abstractive headline, based on the introduction of an article as the original text. Using the previous framework expansions, we calculate the context vector, the copy vector to generate words outside of the model dictionary, and the coverage penalty to prevent repetition in generated summaries. Table 1 shows an example of the model output that contains article's introduction and the resulting headline. The general form of the model is shown in Fig. 1, while the flowchart of the model is shown in Fig. 4.

Attention mechanism

We take advantage of the attention mechanism presented by Bahdanau et al. [12], which gives the decoder the ability to identify important portions of the text, which are required in the generation of the next word of the summary. The mechanism input is (1) the tokens of the article w_i which are fed one-by-one into the encoder producing a sequence of encoder hidden states h_i. This hidden state represents the original text. (2) The hidden state of the decoder s_t. On each time step t, the decoder receives the word embedding of the previous

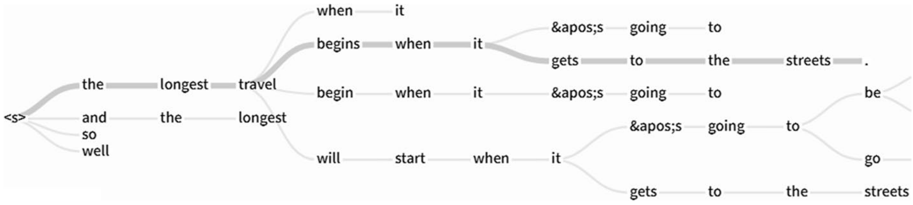

Fig. 3 Pointer-generator model (Derived from See et al. [11])

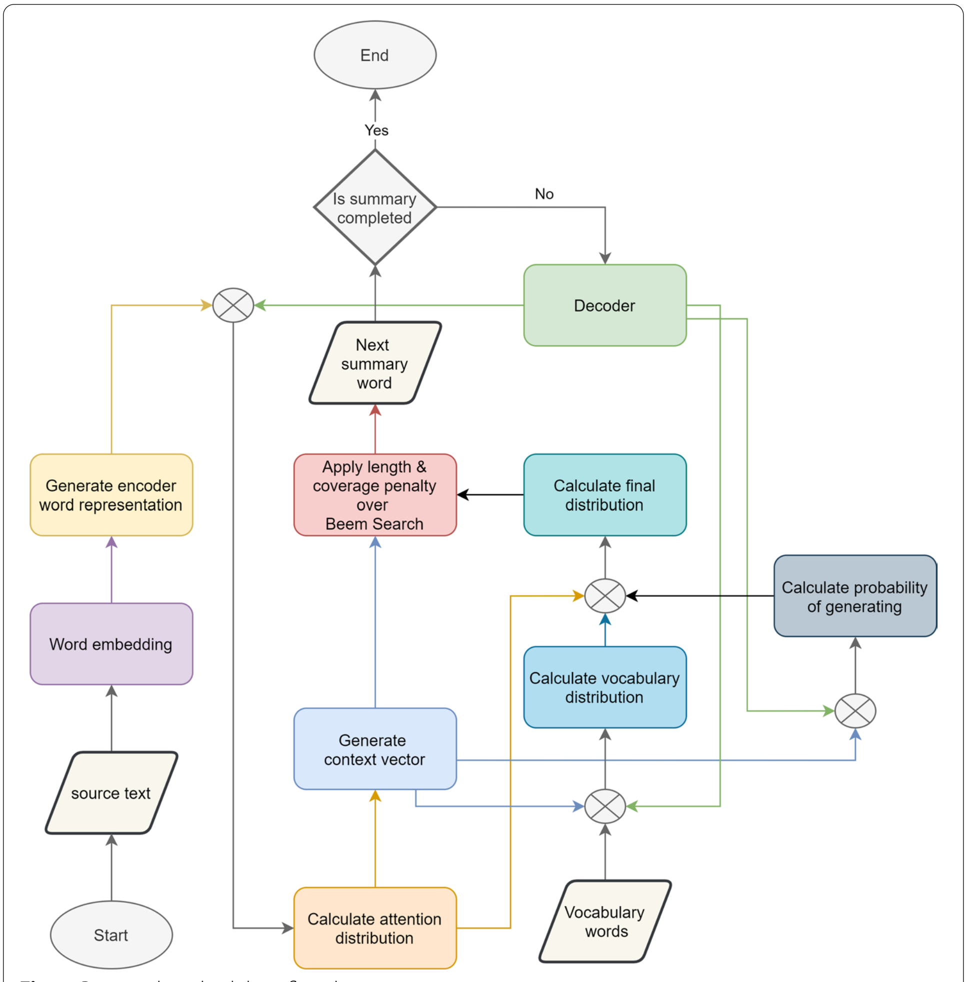
End
Yes
Is summary completed
No
Decoder
Next summary word
Generate encoder word representation
Apply length & coverage penalty over Beem Search
Calculate final distribution
Calculate probability of generating
Word embedding
Calculate vocabulary distribution
Generate context vector
source text
Vocabulary words
Start
Calculate attention distribution

Fig. 4 Proposed methodology flowchart

word, and has decoder state s_t which includes generated words from summary up to this point. Based on these inputs, the mechanism gives a degree of focus to each word of the original text called attention distribution a^t, and uses all of them to generate a context vector from which the decoder takes advantage of the generation process as shown in Fig. 1. The attention distribution a^t is calculated using Eqs. (1) and (2):

$$e_i^t = v^t tanh(W_h h_i + W_s s_t + b_{attn}), \tag{1}$$

$$a^t = softmax\left(e_i^t\right), \tag{2}$$

where each of $v, W_h,\ W_s, b_{attn}$ are learning parameters. a^t given by Eq. (2) can be seen as a probability distribution over the words of the original text, which tells the decoder where to focus while generating the next word. The next step in this model is to generate a weighted sum of the hidden states of the decoder, called the context vector h_t^*:

$$h_t^* = \sum_i a_i^t h_i. \tag{3}$$

The context vector can be viewed as a fixed-size representation of everything read from the original text up to this moment. The context vector combined with the hidden state of the decoder are passed to two linear layers to generate the words' dictionary distribution P_{vocab}.

$$P_{vocab} = softmax\left(V'\left(V\left[s_t, h_t^*\right] + b\right) + b'\right), \tag{4}$$

where each of V, V', b, b' are learning parameters.P_{vocab} is a probability distribution over all of dictionary words.

$$P(y) = P_{vocab}(y). \tag{5}$$

Copy mechanism

Since there are a number of tokens appearing in the original text that are outside the dictionary of the model, there must be a mechanism for generating these words. We can use the copy mechanism provided by Vinyals et al. [23], which was first introduced in the field of text summarization by Gu et al. [10] to demonstrate the possibility of merging the merits of the two abstractive and extractive approaches. This mechanism provides the ability to copy words from the original text, thus giving the model the ability to generate words out of vocabulary (OOV) so that it is not restricted to a preset fixed dictionary. Copy models expand their decoder by predicting a binary soft switch, called Z_j, which determines whether the model will copy or generate. The copy distribution is a probabilistic distribution over the original text, and the joint distribution is calculated as a convex combination for both parts of the model.

$$p(y_j|y_{1:j-1},x) = p(z_j = 1|y_{1:j-1},x)*p(y_j|z_j = 1,y_{1:j-1},x)+p(z_j = 0|y_{1:j-1},x)*p(y_j|z_j = 0,y_{1:j-1},x), \tag{6}$$

where the two parts represent the copy part and the generation part, respectively. We reused the attention distribution $p(a_j|x, y_{1:j-1})$ as a copy distribution, following the pointer-generator model of See et al. [11]. For example, we calculate—using the copy attention—the possibility of copying a 't' token from the original text as the sum of attentions for all places where 't' appeared. In Fig. 1, for each decoder time step, the probability of P(z_j = 0) and P(z_j = 1) is calculated, which determines whether the model will copy a word from the source or generate it from the vocabulary. The vocabulary distribution and the attention distribution are weighted and summed together to obtain the final distribution, from which prediction is made. Note that out-of-vocabulary original text words like "ب5" are included in the final distribution.

Coverage and length penalty

We used a weighting function that included a penalty on length *lp* and a penalty on coverage *cp*, which is defined as:

$$s(x,y) = \frac{\log p(y|x)}{lp(x) + cp(x;y)}. \tag{7}$$

Length

We normalized length during beam search phase, using the length penalty defined by Wu et al. [28] which is defined as follows:

$$lp(y) = \frac{(5 + |y|)^{\alpha}}{(5 + 1)^{\alpha}}. \tag{8}$$

With a tunable parameter α, where increasing α leads to longer summaries. We set its value to 0.5 to achieve balanced length of addresses. In addition, we have set a minimum length for the output sequence based on the training data.

Repeats

Copy models usually tend to refer to the same tokens in the original text, resulting in the same phrases being generated in the output multiple times. We followed the same method introduced by Gehrmann et al. [29] in calculating the summary-based coverage penalty.

$$cp(x; y) = \beta\left(-n + \sum_{i=1}^{n} \max\left(1.0, \sum_{j=1}^{m} a_i^j\right)\right). \tag{9}$$

This penalty is increased if the decoder reaches more than 1.0 total attention towards a specific encoded token. By selecting a high enough value for β in Eq. (9), this penalty prevents summaries that would cause a repetition in the output.

Experiments

Data set

We are dealing with a data set that we built by gathering Arabic articles published by the Arabic website, mawdoo3 [30], in a wide variety of topics. We considered the introduction paragraph of the article as the original text, and in return, we considered the title of the article as the correct summary of this introductory paragraph. We named this newly created dataset, Arabic Headline Summary (AHS). Each entry in the dataset went through several steps of cleaning and enhancing.

This dataset is divided into three sections: training, validation and test, as shown in Table 2. By simulating the work of See et al. [11], we truncate source texts to 900 tokens and target summaries to 100 tokens in both of the training and validation sections. We also limit both input and output vocabulary to the 100,000 most frequent words, and replace the rest with the UNK tokens.

Table 2 The new Arabic dataset, AHS, statistics

Dataset	Train	Valid	Test	DL	SL
AHS	235,870	38,968	20,000	80.6	3.3

DL and SL denote average number of tokens in source document and summary, respectively

Data preprocessing

Since we created a newly dataset in Arabic language, we did specific preprocessing steps to normalize the dataset entries. The steps we followed are:

a. Add a space between (conjunction letters/commas/special characters) and the following word, like (الموضوع ، و بالتالي → الموضوع، وبالتالي).
b. Remove all the diacritics, like (يتمتع → يَتَمَتَّعْ).
c. Remove repeated character, like (الموضوع → الموضـــــوع)
d. Remove unwanted extra spaces (نقص أضرار نقص→أضرار)
e. Remove unusual entries like poems (قِفَا نَبْكِ مِنْ ذِكْرَى حَبِيبٍ وَمَنْزِلِ بِسِقْطِ اللِّوَى بَيْنَ الدَّخُولِ فَحَوْمَلِ).

Justification for a, b, c: ensure that the same word does not have more than one form.

Justification for d: remove noise from data.

Justification for e: remove unusual form of text that also could have rare words.

Training details

We re-applied the pointer-generator model as defined by See et al. [11]. The model architecture consists of two-way Long Short Term-Memory (LSTM) encoder that has 256 hidden states in both directions and 128-dimensional word embedding, while the decoder consists of one layer that has 512 hidden states. The model was trained using Adagrad [31], with an initial training rate of 0.15 and an initial assembly rate of 0.1. The training rate decreased to 10^{-5} after the nineteenth epoch with a perplexity rate of 12.7354. We do not use dropout and use gradient-clipping with a maximum norm of 2. We implemented our model using pyTorch on the open source tool OpenNMT-py [32]. We ran the experiment on Google Colab with a free Tesla K80 GPU and 12 GB RAM.

Model

We tried our dataset on two models; the first, which is the baseline model, uses sequence-to-sequence framework consisting of an encoder and a decoder using a recurrent neural network with both the attention and coverage mechanism, while the second adds the copy mechanism to the baseline model.

Evaluation metrics

We evaluated the accuracy of the summary on the 20,000 test samples using the ROUGE-1 scale. This scale calculates the precision in the summary, which shows the percentage of tokens from the generated summary that is relevant to the reference summary.

$$Precision = \frac{number\ of\ overlapping\ words\ between\ both\ summaries}{total\ words\ in\ reference\ summary}. \tag{10}$$

This scale also uses the recall measure, which shows how far the generated summary covers the reference summary.

$$Recall = \frac{number\ of\ overlapping\ words\ between\ both\ summaries}{total\ words\ in\ generated\ summary}. \tag{11}$$

whereas the F-measure gives the harmonic mean between precision and recall as follows:

$$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{12}$$

These scales are widely used in the text summarization task, where Precision given by Eq. (10) shows the ratio of intersections between the generated summary statements and the reference summary. Whereas the Recall given by Eq. (11) shows how the generated summary fulfills the reference summary.

Results and discussion

The results of the two models that we evaluated in our study are shown in Table 3 and illustrated in Figs. 5 and 6. In Fig. 5, we found that the baseline model achieved better results without the length and coverage penalty, while the pointer-generator model, which uses the copy mechanism, achieved better results than the baseline model, and that reflects the improvement made by the copy mechanism, as illustrated in Fig. 7.

We found that using length penalty with pointer-generator model could lead to slightly improved results, while the results become worse when adding the coverage penalty as illustrated in Fig. 6. The slight improvement in the results could be related to the short length of reference summaries in our dataset, average length 3.3 as shown in Table 2. Consequently, the combined effect of the length penalty, which restricts the length of the summaries (headlines) to match the length limit, alongside the copy mechanism, which copies words not included in the model dictionary, could led to this improvement. Whereas the poor results when using the coverage penalty is related to the fact that this mechanism targets relatively long texts to examine the coverage of these texts for various topics, and its effect was inversed with short texts.

Table 3 Results of the ROUGE scale for the two models applied to the Arabic dataset, AHS

Model	Inference penalty	ROUGE-1 average precision	ROUGE-1 average recall	ROUGE-1 average F-measure
Baseline model	No penalty	57.02	21.05	30.67
	Coverage	55.93	19.60	28.95
	Length	55.58	21.05	30.46
	Length + coverage	55.16	19.60	28.85
Baseline model + copy mechanism	No penalty	62.01	33.53	43.42
	Coverage	60.43	32.12	41.84
	Length	62.13	34.46	44.23
	Length + coverage	61.36	32.59	42.47

All our ROUGE scores have a 95% confidence interval of at most ± 0.25

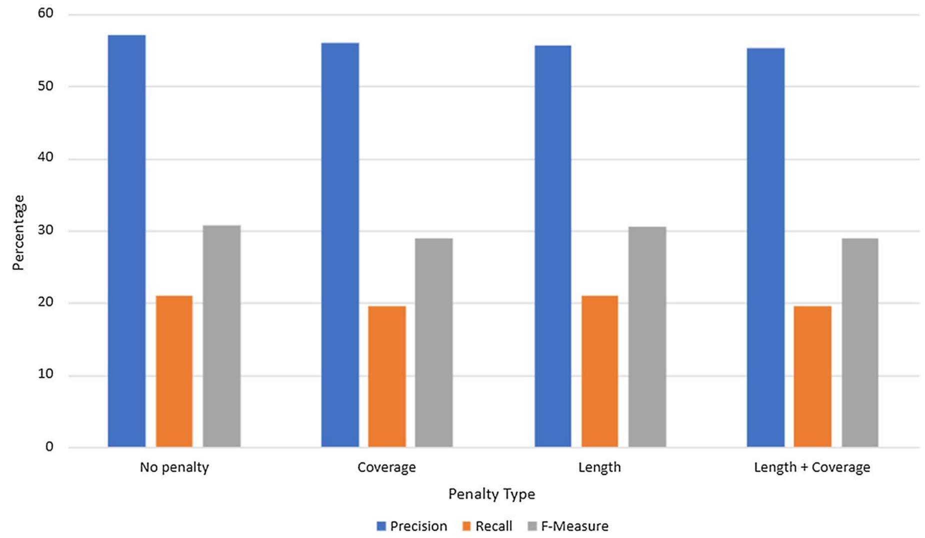

Fig. 5 Results comparison of baseline model

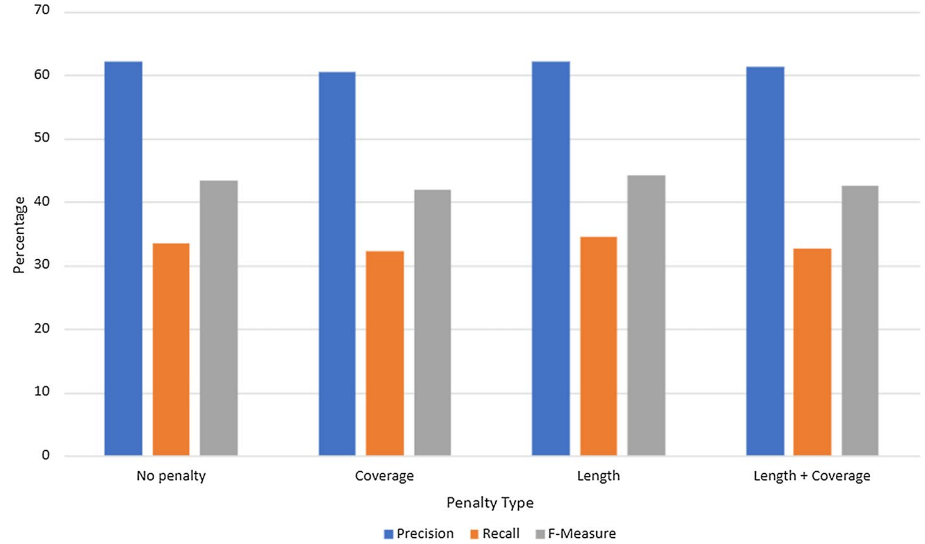

Fig. 6 Results comparison of baseline model with copy mechanism

We did not use the ROUGE-2 scale because the average reference summaries length was less than four, therefore, this metric is not a good choice for this case as it depends on pairs of words. Table 4 shows examples of pointer-generator model's results. Example (1) shows how model generated word “أضرار” (harms) in place of “مضار” (harmful), while Example (2) shows how model generated word “تحسين” (improve) in place of "حفاظ" (maintain).

By comparing the results of the models that we applied to the Arabic dataset, AHS, with other abstractive models applied to Gigaword and CNN Daily Mail datasets, we note that the pointer-generator model with length penalty, has achieved better results

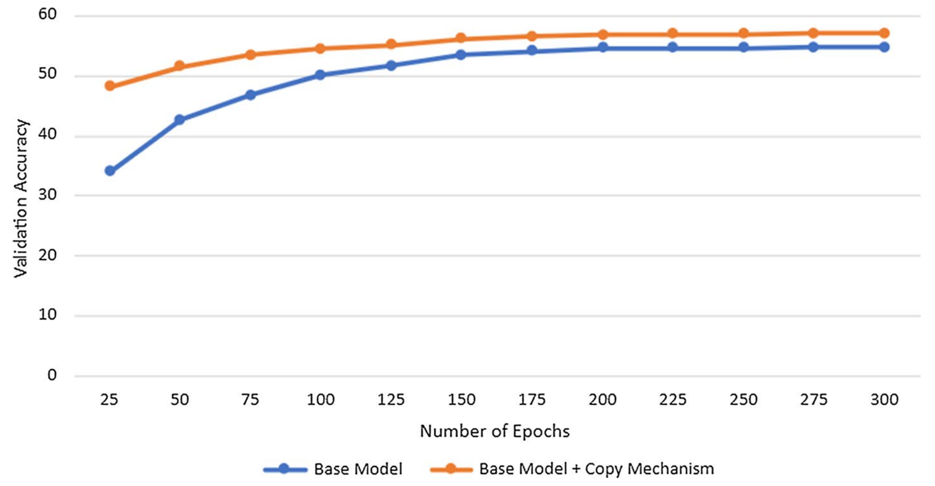

Fig. 7 Results comparison between baseline model and baseline model with copy mechanism for different number of epochs

Table 4 Two examples of headline generation by the pointer-generator model with penalties

	Origin	Translation
Example (1)		
Article	يسبب نقص الحديد العديد من المضار لجسم الإنسان بعضها قد يكون خطيرا و بعضها قد يكون طبيعيا و يعالج بسهولة ، و من هذه المضار (...)	Iron deficiency causes many harmful effects to the human body, some of which may be dangerous, some of which may be natural and easily treated, and of these harmful effects (…)
Pointer-generator summarization	أضرار نقص الحديد.	Iron deficiency harms
Reference summarization	مضار نقص الحديد.	The harmful effects of iron deficiency
Example (2)		
Article	يمكن أن تساهم العديد من الأطعمة في الحفاظ على صحة القلب ، و تقليل خطر إصابته بالعديد من الأمراض ، و منها ما يلي (...)	Many foods can contribute to maintaining a healthy heart, and reduce the risk of many diseases, including the following (…)
Pointer-generator summarization	تحسين صحة القلب.	Improve heart health
Reference summarization	الوقاية من الأمراض.	Disease prevention

The word underlined is a word generated by the model

Table 5 Results of abstractive summarizers on different datasets

References	Method	ROUGE F1	Dataset
Rush et al. [9]	ABS	30.88	Gigaword
	ABS+	31.00	
See et al. [11]	Pointer-generator	36.44	CNN/daily mail
	Pointer-generator + coverage model	39.53	
Gehrmann et al. [29]	Pointer-generator + coverage penalty	39.12	
	Bottom-up summarization	41.22	
(Ours)	Pointer-generator + length penalty	44.23	AHS

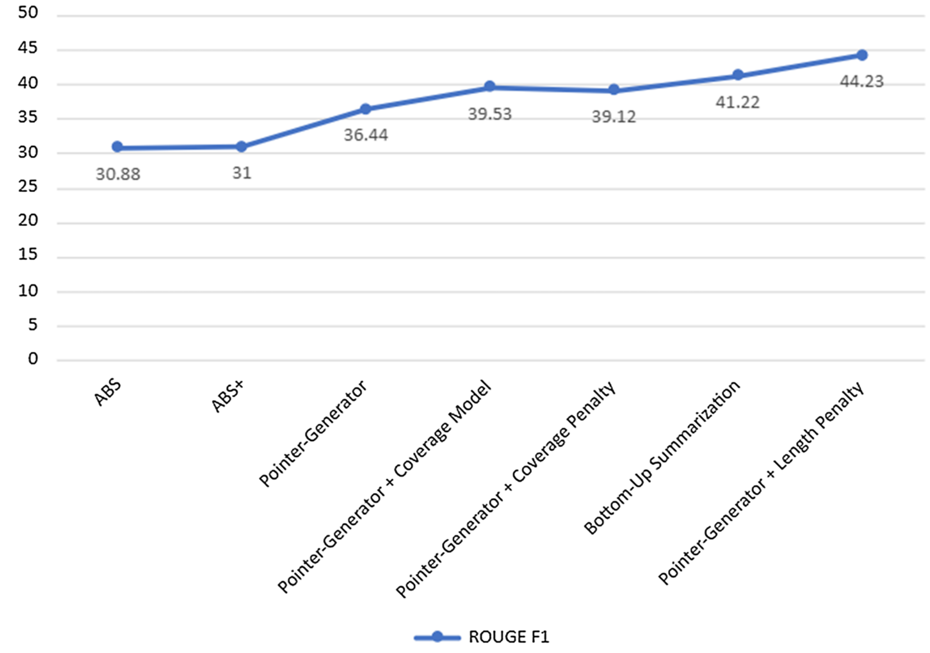

Fig. 8 ROUGE F1 results comparison between abstractive summarizers on different datasets

for the Arabic dataset as shown in Table 5 and illustrated is Fig. 8. We assume that this improvement is due to two main reasons:

1. The nature of the dataset, which consists of short reference summaries (headlines) comparing to other datasets.
2. The nature of the Arabic language in terms of more language grammar consistency within written text, which contributed to this improvement.

Arabic research results comparison

For comparison with Arabic research that uses deep learning to summarize texts, we relied on the latest Arabic research based on deep learning that approximates the task of summarizing texts, which is extracting keyphrases from text [27]. The length of the extracted key phrases, two to three words, will correspond to the length of the titles generated by the model we worked on.

Since the idea of the research in [27] is close to our research and uses deep learning, it was appropriate to compare the two researches. We applied our model (pointer-generator with a length penalty) to the test dataset of [27], which has 940 entries. The results of the summary contained many (UNK) tokens, which means the word is outside the dictionary of the model, thus the comparison was not possible. We managed to explain what happened with the following reasons:

1. The dataset was pre-processed in [27] in a way that effected the original form of the words, such as:

   a. Writing letters "أ، إ، آ، ا" in the normal form "ا".
   b. Writing letter "ة" in the normal form "ه".

2. The compound Arabic word was divided into its primary parts, such as "ركبت" became "ركب ت" using Stanford Core NLP [33]. Table 6 shows the results of the modification.

These modifications to the dataset have resulted in the words between the two datasets is no longer being compatible, and the first have a set of vocabulary words that differ from their original forms. This resulted in a large number of words unknown to our model, consequently, irrational results for the ROUGE scale. Given the foregoing, we can consider the comparison between the results of the two researches is only feasible by applying the same modifications to our data set and retraining the model. However, modifying the word structure in the data set contradicts the idea of summarizing texts using deep learning and may generate incorrectly spelling headlines.

Conclusion

Sequence-to-sequence framework that has the encoder-decoder form has gained an increased interest in the field of text summarization. Recurrent neural networks have been improved to be used in the field of text summarization. Many studies were taken place in this field regarding the English language. In this study, we re-implemented the latest approaches and mechanisms, that have been followed in English, on the Arabic language. We started by building a new dataset for Arabic language that is convenient for this task, then applied the abstractive neural model with the attention mechanism,

Table 6 Data entries modifications of the research we compared with [27]

Type	Text
Original	رَكِبْتُ سَيَّارَتِي إِلَى مَدِينَةِ الْقَاهِرَةِ
Trans	I drove my car to Cairo city
No Diac	ركبت سيارتي إلى مدينة القاهرة
Normal	ركبت سيارتي الي مدينه القاهره
Segmented	ركب ت سياره ي الي مدينه القاهره

which we called the baseline model, and examined its results. By adding the copy mechanism, we expanded the baseline model to match the pointer-generator model defined by See et al., then showed the improved results after taking advantage of both abstractive and extractive approaches. We also tried both of the models with coverage and length penalties and found that pointer-generator model with length penalty achieved the best results. There are other hypotheses that could be tested to improve results in the future. We wish that this study with its new dataset, and the two models it worked on and compared between their results, would be a starting point for future research in this field that can achieve new enhancements for the Arabic language.

Future work should consider expanding the data set to cover more articles. The dataset size that we created, AHS, is close to the size of the CNN/Daily Mail dataset, but can still be expanded to the size of the Gigaword collection. It also could apply other mechanisms that may be useful in the domain of sequence-to-sequence framework, such as implementing a coverage mechanism using a separate coverage vector as what has done by See et al. However, the attempting to infer new models that are beneficial with the Arabic language, in particular, will be the best part to work on since it is a unique grammatical language written from right to left.

Abbreviations

ABS: Attention Based Summarization; DUC: Document Understanding Conferences; RST: Rhetorical Structure Theory; GP: Genetic Programming; OOV: Out of Vocabulary; AHS: Arabic Headline Summary; LSTM: Long Short Term-Memory.

Acknowledgements

Not applicable.

Authors' contributions

MA-M took on the main role so he performed the literature review, implemented the proposed model, conducted the experiments and wrote the manuscript. SD took on a supervisory role and oversaw the completion of the work. Both authors read and approved the final manuscript.

Funding

The authors declare that they have no funding.

Availability of data and materials

The dataset supporting the conclusions of this article is available in the Open Science Framework repository, https://osf.io/btcnd/.

Competing interests

The authors declare that they have no competing interests.

Author details

[1] Faculty of Information Technology, Higher Institute for Applied Sciences and Technology, Damascus, Syria. [2] Faculty of Informatics and Communication Engineering, Arab International University, Damascus, Syria.

Received: 11 February 2020 Accepted: 26 November 2020
Published online: 11 December 2020

References

1. Belkebir R, Guessoum A. A supervised approach to Arabic text summarization using adaboost. In: New contributions in information systems and technologies. New York: Springer; 2015. p. 227–36.
2. Imam I, Nounou N, Hamouda A, Abdul Khalek HA. An ontology-based summarization system for Arabic documents (ossad). IJCA. 2013;74(17):38–43.
3. Knight K, Marcu D. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. Artif Intell. 2002;139(1):91–107.
4. Jing H. Using hidden markov modeling to decompose human-written summaries. Comput Linguist. 2002;28(4):527–43.
5. Dorr B, Zajic D, Schwartz R. Hedge trimmer: a parse-and-trim approach to headline generation. In: Proceedings of the HLTNAACL03 on text summarization workshop, vol. 5. Edmonton: Association for Computational Linguistics; 2003. p. 1–8.

6. Zajic D, Dorr B, Schwartz R. Bbn/umd at duc-2004: Topiary. In: Proceedings of the HLT-NAACL 2004 document understanding workshop. Boston: Association for Computational Linguistics; 2004. p. 112–9.
7. Cohn T, Lapata M. Sentence compression beyond word deletion. In: Proceedings of the 22nd international conference on computational linguistics, vol. 1. Manchester: Association for Computational Linguistics; 2008. p. 137–44.
8. Woodsend K, Feng Y, Lapata M. Generation with quasi-synchronous grammar. In: Proceedings of the 2010 conference on empirical methods in natural language processing. Cambridge: Association for Computational Linguistics; 2010. p. 513–23.
9. Rush AM, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. arXiv Prepr arXiv:1509.00685. 2015.
10. Gu J, Lu Z, Li H, Li VO. Incorporating copying mechanism in sequence-to-sequence learning. arXiv Prepr arXiv:1603.06393. 2016.
11. See A, Liu PJ, Manning CD. Get to the point: summarization with pointer-generator networks. arXiv Prepr arXiv:1704.04368. 2017.
12. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv Prepr arXiv:1409.0473. 2014.
13. El-Shishtawy T, El-Ghannam F. Keyphrase based Arabic summarizer (KPAS). arXiv Prepr arXiv:1206.5384. 2012.
14. Azmi A. Al-thanyyan S. Ikhtasir—a user selected compression ratio Arabic text summarization system. In: 2009 international conference on natural language processing and knowledge engineering. Dalian: IEEE; 2009. p. 1–7.
15. AlSanie W. Towards an infrastructure for Arabic text summarization using rhetorical structure theory. M.Sc. Thesis, Dept. of Computer Science, King Saud University, Riyadh, Saudi Arabia. 2005.
16. Douzidia FS, Lapalme G. Lakhas, an Arabic summarization system. In: Proceedings of 2004 document understanding conference (DUC2004). Boston: NIST; 2004. p. 260–73.
17. Haboush A, Al-Zoubi M. Arabic text summarization model using clustering techniques. World Comput Sci Inf Technol J. 2012;2:62–7.
18. Sarker IH, Kayes ASM, Watters P. Effectiveness analysis of machine learning classification models for predicting personalized context-aware smartphone usage. J Big Data. 2019;6:57.
19. Sobh I, Darwish N, Fayek M. An optimized dual classification system for arabic extractive generic text summarization. M.Sc. Thesis, Faculty of engineering, Cairo University, Giza, Egypt. 2009.
20. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Adv Neural Inf Process Syst. 2014;4:3104–12.
21. Nallapati R, Zhou B, Ma M. Classify or select: neural architectures for extractive document summarization. arXiv Prepr arXiv:1611.04244. 2016.
22. Chopra S, Auli M, Rush AM. 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. San Diego: Association for Computational Linguistics; 2016. p. 93–8.
23. Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: Proceedings of the 28th international conference on neural information processing systems, vol. 2. Montreal: MIT Press; 2015. p. 2692–700.
24. Tu Z, Lu Z, Liu Y, Liu X, Li H. Modeling coverage for neural machine translation. arXiv Prepr arXiv:1601.04811. 2016.
25. Elmadani K N, Elgezouli M, Showk A. BERT fine-tuning for Arabic text summarization. arXiv Prepr arXiv:2004.14135. 2020.
26. Devlin J, Chang MW, Lee K, Toutanova K. Bert: pre-training of deep bidirectional transformers for language understanding. arXiv Prepr arXiv:1810.04805. 2018.
27. Helmy M, Vigneshram RM, Serra G, Tasso C. Applying deep learning for Arabic keyphrase extraction. Procedia Comput Sci. 2018. https://doi.org/10.1016/j.procs.2018.10.486.
28. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, et al. Googles neural machine translation system: bridging the gap between human and machine translation. arXiv Prepr arXiv:1609.08144. 2016.
29. Gehrmann S, Deng Y, Rush A. Bottom-up abstractive summarization. In: Proceedings of the 2018 conference on empirical methods in natural language processing. Brussels: Association for Computational Linguistics; 2018. p. 4098–109.
30. Mawdoo3. https://mawdoo3.com/. Accessed 02 Feb 2019.
31. Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res. 2011;12:2121–59.
32. Klein G, Kim Y, Deng Y, Senellart J, Rush AM. Opennmt: open-source toolkit for neural machine translation. arXiv Prepr arXiv:1701.02810. 2017.
33. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D. The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations. Baltimore: Association for Computational Linguistics; 2014. p. 55–60.

Publisher's Note