ARMAN: Pre-training with Semantically Selecting and Reordering of Sentences for Persian Abstractive Summarization

Alireza Salemi[1], Emad Kebriaei[1], Ghazal Neisi Minaei[1], Azadeh Shakery[1,2]
[1]School of Electrical and Computer Engineering
College of Engineering, University of Tehran, Tehran, Iran
[2]School of Computer Science
Institute for Research in Fundamental Sciences (IPM), Iran
{alireza.salemi,emad.kebriaei,ghazal.minaei,shakery}@ut.ac.ir

Abstract

Abstractive text summarization is one of the areas influenced by the emergence of pre-trained language models. Current pre-training works in abstractive summarization give more points to the summaries with more words in common with the main text and pay less attention to the semantic similarity between generated sentences and the original document. We propose ARMAN, a Transformer-based encoder-decoder model pre-trained with three novel objectives to address this issue. In ARMAN, salient sentences from a document are selected according to a modified semantic score to be masked and form a pseudo summary. To summarize more accurately and similar to human writing patterns, we applied modified sentence reordering. We evaluated our proposed models on six downstream Persian summarization tasks. Experimental results show that our proposed model achieves state-of-the-art performance on all six summarization tasks measured by ROUGE and BERTScore. Our models also outperform prior works in textual entailment, question paraphrasing, and multiple choice question answering. Finally, we established a human evaluation and show that using the semantic score significantly improves summarization results.

1 Introduction

Abstractive text summarization is the task of generating a short, fluent, and concise text that contains novel words and phrases other than the original document, preserving the primary subjects in the document. In contrast with extractive summarization, which aims to select the most important parts of the text to generate a summary, in abstractive summarization, the main goal is to generate a new persuasive piece of text as the summary of a document.

Earlier abstractive summarization works (Hermann et al., 2015; See et al., 2017; Rush et al., 2015) focused on training with large datasets containing pairs of documents and summaries in a supervised manner. By introducing Transformer (Vaswani et al., 2017) architecture and pre-training objectives and their positive impact on most NLP tasks, most current state-of-the-art (SOTA) methods focused on self-supervised objectives for pre-training Transformer architecture in abstractive summarization tasks (Liu and Lapata, 2019; Zhang et al., 2020a; Qi et al., 2020). However, current pre-training works give more points to the summary with more words in common with the main text and pay less attention to the semantic similarity between generated sentences and the original document.

According to Simons (2017), the Persian language is one of the top 25 spoken languages in the world. However, there are limited research studies in Persian document summarization, and most of the prior works were mainly focused on extractive summarization. The main focus of this work is on Persian abstractive summarization. Nevertheless, our proposed method is language-independent.

In this work, we first bring semantic similarity scores into a sentence selection schema to create a document's pseudo summary. Briefly, we prepare a summary corresponding to each document in a dataset by selecting important sentences based on semantic scores in a self-supervised manner. Next, we propose three novel objectives for pre-training a seq2seq Transformer. Our model, ARMAN, uses Transformer encoder-decoder structure and introduces a new combination of masking sentences with sentence shuffling and reordering objectives. We fine-tuned the models on six downstream tasks. According to an experiment, we found that letting the training model to copy pieces of the input text into the output summary does not lead to better results in downstream tasks. Experiment results showed that our proposed models obtained SOTA performance in all Persian abstractive sum-

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9391–9407
November 7–11, 2021.

marization datasets on both ROUGE (Lin, 2004) and BERTScore(Zhang et al., 2020b). Our models generated even better summaries than previous SOTA in zero and few shot settings when fine-tuned with a small number of document-summary pairs. We achieved SOTA results on two datasets with only 1K examples. Moreover, our proposed models performed well in other NLU tasks, including textual entailment, question paraphrasing, and multiple choice question answering. Finally, to ensure the significant improvement in summarization, we held a human evaluation, and we performed a student t-test on its results.

The main contributions of this paper are threefold:

- We introduce a top-sentence selection algorithm based on a semantic score to make document-summary pairs in a self-supervised manner.
- We propose three novel objectives to pre-train a Transformer encoder-decoder architecture for Persian abstractive text summarization that outperforms previous state-of-the-art models on six downstream tasks.
- We created an abstractive summarization dataset called Tebyan.

2 Related Work

Automatic text summarization was mainly performed based on statistical methods (Nenkova, 2005); most of them were striving to rank sentences by extracting their features(Svore et al., 2007; Erkan and Radev, 2004; Filippova and Altun, 2013). By rising of sequence to sequence learning with neural networks (Hochreiter and Schmidhuber, 1997; Sutskever et al., 2014) and attention mechanism (Bahdanau et al., 2015) usage in abstractive summarization tasks (Nallapati et al., 2016), a new era in abstractive summarization began.

By introducing Transformer (Vaswani et al., 2017) and Masked Language Modeling (MLM) methods of BERT (Devlin et al., 2019), most NLP tasks achieved a vast improvement gain using these pre-training methods and architectures. Following BERT's approach, many other Language Models were trained (Liu et al., 2019; Joshi et al., 2020) with differences in the amount of data used for pre-training and some optimizations on BERT's pre-training method; most of them were only Encoders. Furthermore, Encoder-Decoder models were trained with a mixture of pre-training tasks; T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) are two of them.

Since the pre-training of Transformer was successful on most NLP tasks, some models were pre-trained for specific duties; PEGASUS (Zhang et al., 2020a) is a pre-trained model that was trained specifically for summarization on C4 and HugeNews corpora. PEGASUS trained with Gap Sentence Generation (GSG) that masks the most important sentences based on syntactic similarity of sentences of a document. ARMAN is different from PEGASUS in that we mask the most important sentences based on the semantic similarity of sentences. Furthermore, we use only a single mask token for any consecutive sentences that should be masked. This approach helps the model learn how many sentences should be generated for each masked token in the input sequence. STEP (Zou et al., 2020) is another pre-trained summarization model trained with MLM, Next Sentence Generation (NSG), and Sentence Reordering (SR) objectives. ARMAN uses SR as one of the pre-training methods in a modified form; we change the order of sentences in the input document. The model should select the most important sentences using semantic similarity of sentences to the document, then reorder them in the actual order that they appeared in the original document.

In the Persian language, some extractive summarization methods exist (Khademi et al., 2018; Rezaei et al., 2019; Kermani and Ghanbari, 2019; Khademi and Fakhredanesh, 2020), but to the best of our knowledge, we know just one model on abstractive summarization. Farahani et al. (2020b) have used ParsBERT (Farahani et al., 2020a) checkpoint with Rothe et al. (2020)'s method to train a new sequence to sequence model with pre-trained weights for the encoder and decoder. In this regard, ARMAN is one of the first works on abstractive summarization for the Persian language. Also, ARMAN was able to achieve SOTA results on all available datasets.

3 Methodology

This section introduces a sentence selection method based on semantic similarity scores to make a pseudo summary. Then, we propose three novel objectives for pre-training a seq2seq model for the abstractive summarization tasks.

3.1 Top Sentence Selection (TSS)

We introduce a new semantic-based approach for selecting important document sentences to make a pseudo summary in this work. The pseudo summary consists of important sentences of a given document, and the models are supposed to generate an output similar to the pseudo summary corresponding to the document. For comparison, we also use a syntactic-based metric to select sentences from the original document. Inspired by recent work in generating pseudo summaries (Zhang et al., 2020a), we select sentences from a document based on two strategies and concatenate them to create a pseudo summary. For each document in a data collection, we make a summary as described in Algorithm 1. At first, we calculate a score function for each pair of $(sentence, document \setminus sentence)$. Then we calculate the top m sentences and merge them to make the pseudo summary. The parameter m is calculated based on the number of sentences.

Algorithm 1: Top Sentence Selection

Input : Document
Output : Text, Summary
  for s_i in $Document$ **do**
    $r_i := score\_func(s_i, Document \setminus s_i)$
  end for
  $Summary := \emptyset$
  $Text := Document$
  for $j \leftarrow 1$ to m **do**
    $k := argmax\{r_i\}_{\forall\, s_i \notin Summary}$
    $Summary := Summary \cup \{s_k\}$
    $Text := Text \setminus \{s_k\}$
  end for

Syntactic-based approach: In this strategy, we create a pseudo summary by selecting and merging sentences from a document using a syntactic-based approach. ROUGE is a mainly used metric that calculates the similarity between a candidate sentence and a collection of reference sentences based on the overlap of N-grams (Lin, 2004). The higher the ROUGE score between two pieces of text, the more similar they are. The *score-func* in Algorithm 1 calculates the ROUGE1-F1 score between the sentence and remaining sentences of the document. PEGASUS (Zhang et al., 2020a) has used such a method as Gap Sentence Generation.

Semantic-based approach: Although selecting sentences based on the ROUGE metric is simple, cost-effective, and usable in low-resource languages, ROUGE comes with some drawbacks (Kryscinski et al., 2019). In particular, ROUGE does not account for different words with the same meaning since it only calculates syntactical matches. Thus, if we have two sentences with the same meaning but expressed with different words, they will be assigned a low ROUGE score. To the best of our knowledge, this paper is the first to study semantic similarity in creating pseudo summaries and its effect on the quality of generated summaries.

To consider the semantic score in calculating the similarity of two sentences, we used a recent BERTScore metric. BERTScore computes a similarity score for each token in the candidate sentence with each in the reference sentence using contextual embeddings (Zhang et al., 2020b). Due to the high computational cost of calculating this metric for each pair of $(sentence, document \setminus sentence)$, we used FastText (Bojanowski et al., 2017) pre-trained embeddings instead of BERT contextual embeddings. According to BERTScore, for a reference x, and a candidate $\hat{x}$, the recall, precision, and F1 scores are:

$$R_{FT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x_i}^\top \hat{\mathbf{x}}_j,$$

$$P_{FT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x_i}^\top \hat{\mathbf{x}}_j,$$

$$F1_{FT} = 2\frac{P_{FT}.R_{FT}}{P_{FT} + R_{FT}}.$$

For applying semantic score, the score function in Algorithm 1 calculates $F1_{FT}$[1].

3.2 Pre-training Objectives

In this work, we propose new pre-training objectives and compare our models with the closely similar work of PEGASUS (Zhang et al., 2020a). We use Transformer encoder-decoder structure and introduce a new combination of masking sentences plus shuffling and reordering objectives. The general procedure of pre-training with the proposed objectives is shown in Figure 1.

TSS-ROUGE

In this objective, we implemented PEGASUS for the Persian language to compare with our proposed models. The base architecture of this model is a Transformer encoder-decoder. Instead of masking words, we mask sentences with <*mask*> tokens. In

[1]FT stands for FastText.

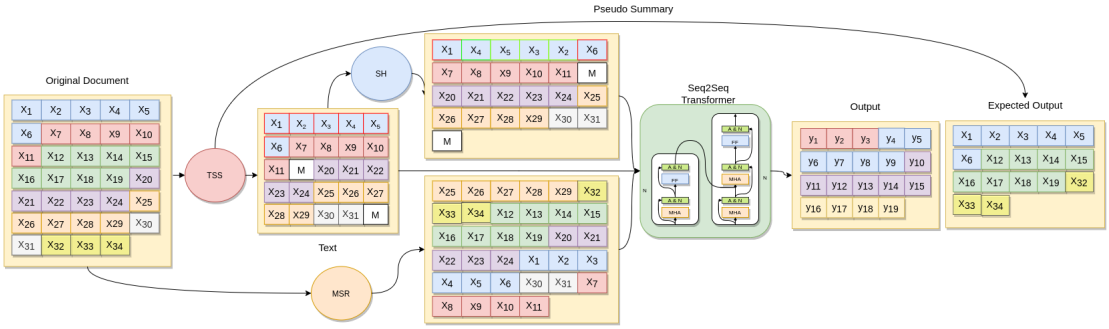

Figure 1: The procedure of making input and output for pre-training Seq2Seq Transformer. TTS selects the salient sentences and divides the original document into text and summary parts. The summary part is the desired output that the Transformer should generate.

order to generate pseudo summaries as input to this structure, the syntactic-based approach using the ROUGE metric is applied.

TSS-Semantic Similarity (SS)

This objective takes semantically created pseudo summaries into account. This method is the same as the previous TSS-ROUGE. The semantic-based approach using the modified BERTScore is applied to generate pseudo summaries as input to the structure. The masking criterion is a bit different from TSS-ROUGE. We put only one <*mask*> token for any number of consecutive sentences that should be masked. In this way, the model learns to guess the number of sentences as well. In 20% of the cases, instead of masking a sentence, we keep it in place; this will make the model learn to bring some pieces of the document into the summary. We call the trained model with this objective ARMAN(SS-80).

TSS-Shuffling (SH)

In addition to considering a semantic-based approach for creating a pseudo summary, we apply span shuffling in a sentence and the masking objective together in this objective. In particular, instead of masking sentences 20% of the cases, we shuffle a span of them. The intuition is that the model will learn not to just copy sentences in the final summary and be sensitive to precedence and latency at the span level. We call the trained model with this objective ARMAN(SH).

TSS-Modified Sentence Reordering (MSR)

In this objective, we do masking as the TSS-Semantic Similarity objective does in 90% of documents, and in 10% of other documents, we shuffle all sentences. In the latter, the model should reorder sentences and keep the top 30% of important sentences of the original document according to the semantic scores. The idea behind this method is that the model will learn to arrange the sentences in the correct order in the final summary. Moreover, the model will learn to care about important pieces of the document. In addition to enriching the summary semantically, this work also considers its brevity. We call the trained model with this objective ARMAN(MSR).

4 Data Collection

This section introduces the datasets used for pre-training and fine-tuning models and the procedure of cleaning corpora.

4.1 Pre-training Datasets

We merged four large Persian corpora from different sources for pre-training models, which contained formal and informal texts.

irBlogs (AleAhmad et al., 2016) is a collection of 5M+ posts from 600K+ Persian weblogs. Some blogs use informal language for their posts, so this dataset has an enormous amount of informal texts, which could help our models become familiar with this type of Persian speech.

MirasText (Sabeti et al., 2018) is an automatically produced text corpus for the Persian language by crawling over 250 Persian websites. This corpus contains around 2.8M articles and 1.4B words in all of the articles.

CC100 (Conneau et al., 2020; Wenzek et al., 2020) is a monolingual dataset for 100+ languages constructed from Commoncrawl snapshots. This dataset contains about 111GB of Persian raw text with 13.3B different tokens.

YJC News[2] is a collection of articles gathered from the Young Journalist Club website[3]. This dataset contains news from various subjects, including 1M+ articles.

4.2 Downstream Datasets

For the Summarization task, five datasets were used. All datasets are publicly available and could be used to reproduce our results. Following Grusky et al. (2018), extractive density and coverage for each summarization dataset has been reported in Appendix A. Moreover, we used a Natural Language Understanding (NLU) dataset to test our models' performances on language modeling tasks.

PN-Summary (Farahani et al., 2020b) is an abstractive summarization dataset consisting of 93,207 articles of various news categories crawled from six news agency websites.

Wiki-Summary (Farahani, 2020b) is a dataset that was extracted from Wikipedia dump files. The main task of this dataset is to generate highlights for each article. There are two versions of this dataset; we used the first version in our experiments, consisting of 56,363 articles and highlight pairs.

VOA Dataset (Farahani, 2020a) is a medium-sized corpus of 7.9 million words consisting of 38,952 articles of the VOA website[4] from 2003 to 2008. The main task that was performed on this dataset was generating a headline for each article.

PerKey (Doostmohammadi et al., 2018) is a key phrase extraction dataset for the Persian language crawled from six Persian news agencies. There are 553k articles available in this dataset. Some of these articles have summaries, and all of them have titles.

Tebyan Dataset accumulates 92,289 document-summary pairs that we have collected from the Tebyan website[5]. These articles consist of various subjects and are not limited to news articles. More information about this dataset is provided in Appendix A.

ParsiNLU (Khashabi et al., 2020) is a collection of NLU tasks for the Persian language including Textual Entailment, Sentiment Analysis, Question Paraphrasing, Multiple Choice Question Answering, and Reading Comprehension tasks. We have fine-tuned our models on most of them to test their performances on NLU tasks.

4.3 Preprocessing

Due to the necessity of a massive amount of data for pre-training of language models, we needed to collect large datasets, but those datasets need to be cleaned. We adopted a heuristic function to produce an automatic pipeline for cleaning our pre-training datasets. First of all, for each document in each dataset, we separated the sentences and removed those which have the following characteristics; 1) sentences with less than five words 2) sentences that do not end with valid Persian end of sentence marks 3) sentences that contain some specific keywords from Persian webpages and javascript codes.

Furthermore, we omitted documents with less than three sentences after the above cleaning. Next, we used the *langdetect*[6] package to filter out any document which is not in Persian with the probability of 0.99. Lastly, we removed duplicate paragraphs of documents. More information about the size of each corpus after cleaning is reported in Appendix A. Our heuristic was inspired by methods from Raffel et al. (2020)'s work. This preprocessing procedure only has been used for the pre-training datasets.

5 Experiments

In this section, we compare ARMAN with previous works and conduct several experiments to assess the performance of the proposed methods. The codes for pre-training and fine-tuning of all models are publicly available on GitHub[7].

5.1 Pre-training and Implementation

Our model is based on Transformer (Vaswani et al., 2017) encoder-decoder structure. We pre-trained ARMAN, which contained a 12 layer encoder and a 12 layer decoder with 768 embedding/hidden size, 3072 feed-forward filter size, and 12 self-attention heads. ARMAN and PEGASUS were trained on

[2] https://github.com/mohammadiahmad/persian-dataset
[3] https://www.yjc.ir/
[4] https://www.voanews.com/
[5] https://www.tebyan.net/
[6] https://pypi.org/project/langdetect/
[7] https://github.com/alirezasalemi7/ARMAN

Model	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
$\text{Transformer}_{\text{base}}$	34.49/16.03/28.91	23.96/6.14/17.66	31.53/11.71/27.41	55.86/43.49/52.22	45.33/29.88/42.85	23.82/6.79/18.55
$\text{PEGASUS}_{\text{base}}$	45.67/27.81/39.71	31.98/11.63/23.79	47.55/28.68/43.57	62.82/51.96/59.48	53.99/39.3/51.72	37.2/21.23/31.47
$\text{ParsBERT}_{\text{base}}$	44.01/25.07/37.76	27.34/7.1/**25.5**	43.54/24.24/40.76	-	-	-
$\text{mT5}_{\text{small}}$	42.25/24.36/35.94	15.2/4.73/12.64	42.32/25.57/38.99	33.88/19.17/28.75	28.5/12.55/25.91	27.16/12.08/21.27
$\text{ARMAN(SS)}_{\text{base}}$	45.98/28.2/40.09	32.27/11.72/23.91	47.91/28.9/43.75	62.97/52.11/59.64	54.18/39.39/51.84	37.53/21.73/31.77
$\text{ARMAN(SH)}_{\text{base}}$	45.89/28.03/39.89	32.04/11.78/23.83	46.96/27.88/42.93	63.47/52.71/60.16	54.5/39.9/52.19	37.6/21.77/31.82
$\text{ARMAN(MSR)}_{\text{base}}$	**46.19/28.41/40.27**	**32.48/11.86**/24.08	**48.23/29.52/44.27**	**63.59/52.87/60.3**	**54.81/40.17/52.51**	**37.79/21.85/31.98**

Table 1: A comparison of results for ARMAN(SS), ARMAN(SH), and ARMAN(MSR) with other pre-trained models on downstream tasks. These results are reported using ROUGE metrics.

Model	PN-Summary P/R/F1	Wiki-Summary P/R/F1	VOA P/R/F1	Perkey(summary) P/R/F1	Perkey(title) P/R/F1	Tebyan P/R/F1
$\text{PEGASUS}_{\text{base}}$	79.86/79.67/79.7	74.29/71.31/72.64	80.84/81.13/80.92	86.13/86.01/86.01	83.68/83.31/83.45	75.26/75.17/75.14
$\text{ARMAN(SS)}_{\text{base}}$	80.08/79.74/79.85	74.24/71.48/72.71	81.02/81.13/81	86.27/86.01/86.09	83.65/83.36/83.46	75.48/75.32/75.32
$\text{ARMAN(SH)}_{\text{base}}$	79.95/79.69/79.76	74.25/71.43/72.68	80.64/80.91/80.71	86.46/86.22/86.29	83.85/83.49/83.62	75.48/75.28/75.29
$\text{ARMAN(MSR)}_{\text{base}}$	**80.14/79.84/79.93**	**74.67/71.55/72.95**	**81.1/81.35/81.16**	**86.54/86.24/86.33**	**83.93/83.59/83.71**	**75.49/75.46/75.4**

Table 2: A comparison of results for ARMAN(SS), ARMAN(SH), and ARMAN(MSR) with other pre-trained models on downstream tasks. These results are reported using the original BERTScore metric.

the mentioned pre-training corpora in section 4.1. The batch size and the training steps of pre-training were set to 128 and 1M, respectively. Adafactor (Shazeer and Stern, 2018) with square root learning rate decay and a dropout rate of 0.1 was used in pre-training and fine-tuning. Pre-training experiments were carried out on the Google Colab platform with TPU v2-8. It took almost 11 days for 1M steps to train ARMAN. Also, we sampled 1M documents from the CC100 dataset and used the SentencePiece Unigram algorithm (Kudo, 2018) to generate the vocabulary for our models. The size of the vocabulary was 96K in all experiments.

5.2 Fine-tuning on Text Summarization

Abstractive summarization aims to produce a short, fluent, and concise text using advanced natural language techniques to extract essential information from the original document. We fine-tuned our pre-trained models on six downstream tasks. In all experiments, we set the input length (L_{input}) to 512 and output length to 256. Also, we used beam-search as Wu et al. (2016)'s approach with a beam-size of 8 and a length penalty of 0.8. More information about the experiments' setup is reported in Appendix B.

Table 1 shows results based on standard ROUGE metrics. To compare summaries generated by our models with the state-of-the-art $\text{PEGASUS}_{\text{base}}$ with a text generation evaluation metric, we reported results based on original BERTScore (Zhang et al., 2020b) (using bert-base-multilingual-cased as pre-trained contextual embeddings) in Table 2. Both tables show the performance improvements of $\text{ARMAN(MSR)}_{\text{base}}$ on all downstream datasets. According to tables 1 and 2, even $\text{ARMAN(SS)}_{\text{base}}$, our basic proposed method, outperforms $\text{PEGASUS}_{\text{base}}$ in all datasets. These results show that considering the semantic similarity in pre-training objectives is critical in improving the final summary.

In $\text{ARMAN(MSR)}_{\text{base}}$, we encouraged the model to learn the correct relative orders between sentences by reordering at the sentence level. Results of this model show that the reordering objective gives an improvement in summarization. Our second model, $\text{ARMAN(SH)}_{\text{base}}$, does not help in improving the quality of summaries. So, we conclude that shuffling at the span level leads to a sub-optimal response, as reported in Raffel et al. (2020).

5.3 To copy or not to copy!

We observed that $\text{PEGASUS}_{\text{large}}$[8] tries to copy sentences from the document into a generated summary when it is not fine-tuned on any summarization datasets.The intuition is that when the task is to copy a sentence, and in return for that copying the model gets an extra score, the model becomes biased towards copying the sentences to increase the probability of catching a significant match. In other words, it always copies some sentences from

[8] https://huggingface.co/google/pegasus-large

Model	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
ARMAN(SS-80)$_{base}$	45.98/28.2/40.09	32.27/11.72/23.91	**47.91**/28.9/43.75	**62.97/52.11/59.64**	54.18/39.39/51.84	37.53/21.73/31.77
ARMAN(SS-100)$_{base}$	**46.33/28.57/40.38**	**32.36/11.78/24.1**	47.73/**28.95/43.89**	62.83/51.92/59.53	**54.25/39.51/51.92**	**37.64/21.78/31.94**

Table 3: Comparison of ARMAN(SS-80) and ARMAN(SS-100) results on tasks using ROUGE metrics.

Model	PN-Summary Dens/Cov	Wiki-Summary Dens/Cov	VOA Dens/Cov	Perkey(summary) Dens/Cov	Perkey(title) Dens/Cov	Tebyan Dens/Cov
ARMAN(MSR)$_{base}$	**8.29486/0.87188**	**2.55229/0.68437**	4.59273/0.89648	**13.08480/0.84591**	**2.50826/0.81320**	18.56819/**0.86931**
PEGASUS$_{base}$	8.73796/0.87553	2.60724/0.68463	**4.35264/0.88661**	13.48538/0.84700	2.51945/0.81221	**18.23422**/0.87605

Table 4: Comparison of ARMAN(MSR) and PEGASUS results on tasks using Density(Dens) and Coverage(Cov) (Grusky et al., 2018) metrics. ARMAN has less Density and Coverage in 4 out of 6 datasets.

Model	PN-Summary F1-T/P-S	Wiki-Summary F1-T/P-S	VOA F1-T/P-S	Perkey(summary) F1-T/P-S	Perkey(title) F1-T/P-S	Tebyan F1-T/P-S
ARMAN(MSR)$_{base}$	**0.71184/0.82143**	**0.29325/0.62210**	**0.58415/0.95609**	0.58555/0.85361	**0.61337/0.90528**	**0.33835/0.95138**
PEGASUS$_{base}$	0.62983/0.81188	0.28139/0.60744	0.57440/0.94425	**0.64304/0.85691**	0.54963/0.89825	0.30000/0.86349

Table 5: Comparison of ARMAN(MSR) and PEGASUS results on tasks using F1-Target(F1-T) and Precision-Source(P-S) (Nan et al., 2021). ARMAN has a higher F1-Target and Precision-Source in 5 out of 6 datasets.

the input to the output with the hope that it will match the output because this yields a decrease in the loss function value.

We set up an experiment to observe the behavior of our models when they are not encouraged to copy sentences of the input into the output. According to semantic score, all proposed methods selected 30% of the top-ranked sentences. In this experiment, we pre-trained ARMAN(SS)$_{base}$ with two different values for masking rate in TSS objective; 1) ARMAN(SS-80)$_{base}$ masked only 80% of important sentences and left the other 20% unchanged in the input text, 2) ARMAN(SS-100)$_{base}$ masked all of the important sentences without copying any sentences from input text into the pseudo summary.

Results in Figure 2 show that in a zero-shot setting, ARMAN(SS-100)$_{base}$ produces a higher ROUGE score when we do not consider copying in the pre-training objective. Additionally, we fine-tuned ARMAN(SS-100) and ARMAN(SS-80) on downstream tasks. Results in Table 3 and Figure 2 show that ARMAN(SS-100)$_{base}$ performs better than ARMAN(SS-80) before and after fine-tuning. Given these results, we used this more effective criteria in our best model, ARMAN(MSR)$_{base}$.

5.4 Factual Consistency and Abstractiveness

From another perspective, we compared the abstractiveness and factual consistency of our best model, ARMAN(MSR), with PEGASUS on downstream summarization tasks because they are important factors for assessing the quality of summaries.

To compare the abstractiveness of models, we calculated the *coverage* and *density* (Grusky et al., 2018) of summaries generated by each model. A higher value for *coverage* indicates that the summary uses fewer novel words, and a higher value for *density* is an indicator of a more extractive summary. The average *density* and *coverage* of ARMAN(MSR) and PEGASUS on each dataset are reported in table 4. The results show that ARMAN has a lower *density* and *coverage* compared to PEGASUS in 4 out of 6 tasks. Also, in the Tebyan dataset, ARMAN has a higher *density* but lower *coverage*, which means ARMAN uses more novel words compared to PEGASUS. Therefore we conclude that ARMAN's summaries are more abstractive than PEGASUS.

To compare the factual consistency of models, we calculated *precision-source* and *F1-target* (Nan et al., 2021) metrics. While the mentioned metrics evaluate entity-level factual consistency, they still gives considerable information about the factual consistency of models. In order to extract named entities, we used the ParsBERT (Farahani et al., 2020a) model, which was trained on the PAYMA (Shahshahani et al., 2019) dataset[9]. The average *precision-source* and *F1-target* of ARMAN(MSR) and PEGASUS on each dataset are reported in Table 5. The results show that ARMAN has a higher *F1-target* and *precision-source* score than PEGA-

[9] https://huggingface.co/HooshvareLab/bert-fa-base-uncased-ner-peyma

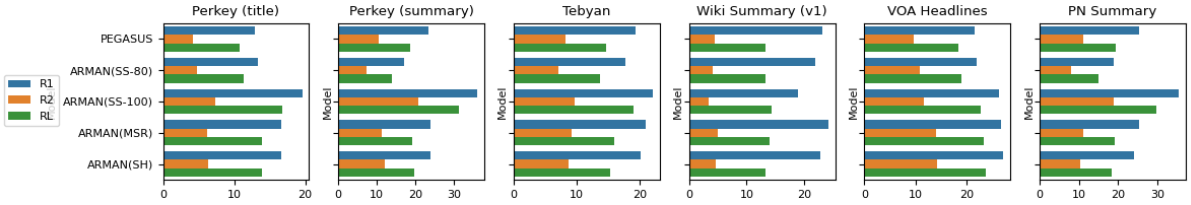

Figure 2: A comparison of results for ARMAN(SS-80), ARMAN(SS-100), ARMAN(SH), ARMAN(MSR), and PEGASUS on zero-shot learning using ROUGE metrics. ARMAN(SS-100) got remarkably better results in most downstream tasks in zero-shot experiments. More details are reported in Appendix C.

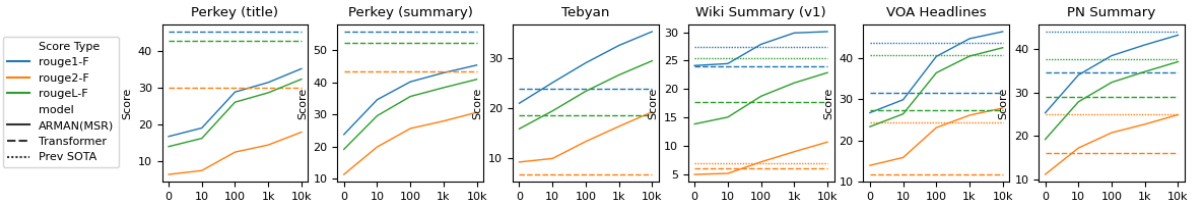

Figure 3: Results of fine-tuning ARMAN(MSR) trained with 0, 10, 100, 1K, 10K examples of each downstream dataset for 2K steps. Also, results of Transformer$_{\text{base}}$, which trained on the whole dataset for 150K steps, and previous SOTA (if available) are shown. The results for other models are reported in Appendix C.

SUS in 5 out of 6 tasks. Therefore, it seems ARMAN is more factually consistent than PEGASUS.

5.5 Zero and Few Shot Summarization

We studied our models in zero and few shot settings to make abstractive summarization a practical solution for real-world tasks where providing a large supervised collection of training and testing data is laborious. In a zero-shot setting, we pre-trained models on pre-training datasets and examined them on downstream tasks without fine-tuning. Results in Figure 2 show that our models outperformed PEGASUS. In a few-shot setting, we fed our best model with 10^k ($k = 1, 2, 3, 4$) examples to study the model's results on low resource scenarios. In this experiment, Transformer$_{\text{base}}$ and ARMAN(MSR)$_{\text{base}}$ were trained for 150K and 2K steps, respectively. According to Figure 3, we observed that in Wiki Summary and VOA datasets, our model has beaten the state-of-the-art model with only seeing 1K samples. In a larger dataset, Perkey, our model did not get a better result than Transformer$_{\text{base}}$ because it was fine-tuned on the whole dataset with more steps. We conclude that our model gets an acceptable outcome in lower amounts of data and computational resources.

5.6 NLU Results

In order to study if ARMAN works well as a language model, we tested our models in Natural Language Understanding (NLU) tasks. According to Khashabi et al. (2020), we selected multiple-choice question-answering, textual entailment, sentiment analysis, and question paraphrasing tasks to examine our models' performance on them. For more information about these tasks and datasets, see Appendix A and Khashabi et al. (2020).

According to the results in Table 6, ARMAN(SH)$_{\text{base}}$ has beaten other models in the natural part of Textual Entailment and Question Paraphrasing. This model learned how to arrange a disordered sentence. Thus, it makes sense why it is powerful in recognizing the same sentences with different written forms. In Multiple-Choice QA, our best-performing model achieves the highest accuracy in math and logic questions. Our proposed model, with semantic similarity and mask-only approach, surpasses others in literature questions. In the common knowledge task, WikiBERT$_{\text{base}}$ (Pyysalo et al., 2021) outperformed other models because it has been trained over a large Wikipedia dataset. In the Sentiment Analysis task, the proposed models could not achieve acceptable results compared to other models. A more detailed study about the behavior of models on NLU tasks is outside the scope of this work.

5.7 Human Evaluation

According to Kryscinski et al. (2019)'s work, we held a human evaluation experiment by considering ROUGE's drawbacks. Our purpose was to de-

	Texual Entailment		Question Paraphrasing		Sentiment (sentence sent.)		Multiple-Choice Question Answering		
Model	**natural (accuracy)**	**translated (accuracy)**	**natural (accuracy)**	**translated (accuracy)**	**food (F1)**	**movie (F1)**	**literature (accuracy)**	**com-know (accuracy)**	**math & logic (accuracy)**
$\text{mBERT}_{\text{base}}$	48.7*	51.6*	80.4	75.3	55.2	48.6	31.1	28.6	33.8*
$\text{WikiBERT}_{\text{base}}$	52.8*	52.6*	80	75.5	52	**58.5**	34.0	**31.4**	32.1
$\text{ParsBERT}_{\text{base}}$	51.8*	**53.9***	79.4	72	**59.1**	56.8	35.4	29.5	32.5*
$\text{mT5}_{\text{small}}$	51.9	51	75.2	72	54.6	49.4	33.7*	24.9	39.1*
$\text{PEGASUS}_{\text{base}}$	54.5	52.6	80	**76.1**	51.9	56	40	27.7	45.1
$\text{ARMAN(SS-80)}_{\text{base}}$	54.5	50.6	82.5	74.8	51.4	47	37.7	25.7	47.7
$\text{ARMAN(SS-100)}_{\text{base}}$	54.2	53	79.9	72.8	50	52.9	**41.4**	27.4	43.1
$\text{ARMAN(SH)}_{\text{base}}$	**55.5**	52.9	**82.6**	75.1	56.7	42	34.6	28.6	45.4
$\text{ARMAN(MSR)}_{\text{base}}$	54.8	51.8	79.9	75.9	52	46	36.57	21.7	**49.14**

Table 6: A comparison of results on ParsiNLU tasks. Some of the reported results (marked with *) in Khashabi et al. (2020)'s work could not be reproduced according to their policies. So we reported the numbers that we ourselves got using their trained models in our experiments.

termine whether semantic similarity makes better summaries than PEGASUS' GSG in the experiment. Also, we wanted to discover which model is the best from the human's viewpoint. We selected 30 documents from the PN-Summary dataset and the corresponding generated summaries from PEGASUS, ARMAN(SS-80), and ARMAN(MSR) models. We gave them to 10 participants and asked them to rank the generated summaries from the best to worst similar to Zou et al. (2020)'s work according to fluency, informativeness, and succinctness of the generated summaries. In order to perform statistical tests, we converted rankings into scores ($score = 4 - rank$). The experiment result is reported in Table 7. Moreover, we have performed some student t-test between models, and results are reported in Table 8. Those results show that ARMAN(MSR) is significantly better than other models ($p < 0.05$). Furthermore, results show that ARMAN(SS-80) is not significantly better than PEGASUS but has an extremely small p-value ($0.0507 > 0.05$).

Model	Rank 1	Rank 2	Rank 3	Score
PEGASUS	31%	35%	34%	1.97
ARMAN(SS-80)	38.33%	34.67%	27%	2.11
ARMAN(MSR)	50.33%	29%	20.67%	2.29

Table 7: Human evaluation results, proportions of model rankings, and average scores. Different models could have the same rankings in tests if they produced the same summary.

p-value	PEGASUS	ARMAN(SS)	ARMAN(MSR)
PEGASUS	-	0.0507	2×10^{-5}
ARMAN(SS)	0.0507	-	0.014
ARMAN(MSR)	2×10^{-5}	0.014	-

Table 8: The p-values for models in comparison. ARMAN(MSR) significantly improves results in comparison with ARMAN(SS-80) and PEGASUS ($p < 0.05$).

6 Conclusion

There are few models for generating abstractive summaries in the Persian language. This work introduces ARMAN, a Transformer encoder-decoder-based model pre-trained with a new combination of masking sentences with sentence shuffling and reordering objectives. We considered semantic similarities for important sentence selection to make document-summary input data in a self-supervised manner. The results show that the modified sentence selection and reordering model outperforms the most recent SOTA models in all six downstream tasks. Our model achieved a higher score than the previous SOTA with only 1K examples in the case of low supervised sample sizes. Finally, the human evaluation results show significant improvement over the dataset used for this experiment.

In future work, investigating the effect of using contextual embeddings for selecting salient sentences for producing text and summary pairs might prove necessary. Furthermore, the ability of models on extractive summarization is worth scrutinizing since our objectives select salient sentences, which is similar to extractive summarization.

Acknowledgements

We would like to thank the anonymous reviewers for their thoughtful and constructive comments. This research was supported in part by a grant from the Institute for Research in Fundamental Sciences (no. CS 1399-4-286).

References

Abolfazl AleAhmad, MohammadSadegh Zahedi, Maseud Rahgozar, and Behzad Moshiri. 2016. irblogs: A standard collection for studying persian bloggers. *Computers in Human Behavior*, 57:195–207.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ehsan Doostmohammadi, Mohammad Hadi Bokaei, and Hossein Sameti. 2018. Perkey: A persian news corpus for keyphrase extraction and generation. *2018 9th International Symposium on Telecommunications (IST)*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

Mehrdad Farahani. 2020a. News headline generation using bert2bert model. https://github.com/m3hrdadfi/news-headline-generation.

Mehrdad Farahani. 2020b. Summarization using bert2bert model on wikisummary dataset. https://github.com/m3hrdadfi/wiki-summary.

Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and Mohammad Manthouri. 2020a. Parsbert: Transformer-based model for persian language understanding.

Mehrdad Farahani, Mohammad Gharachorloo, and Mohammad Manthouri. 2020b. Leveraging parsbert and pretrained mt5 for persian abstractive text summarization.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Max Grusky, M. Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *NAACL*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Fatemeh Hojati Kermani and Shirin Ghanbari. 2019. Extractive persian summarizer for news websites. In *2019 5th International Conference on Web Research (ICWR)*, pages 85–89.

Mohammad Ebrahim Khademi and Mohammad Fakhredanesh. 2020. Persian automatic text summarization based on named entity recognition. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*.

Mohammad Ebrahim Khademi, Mohammad Fakhredanesh, and Seyed Mojtaba Hoseini. 2018. Conceptual text summarizer: A new model in continuous vector space.

Daniel Khashabi, Arman Cohan, Siamak Shakeri, Pedram Hosseini, Pouya Pezeshkpour, Malihe Alikhani, Moin Aminnaseri, Marzieh Bitaab, Faeze Brahman, Sarik Ghazarian, Mozhdeh Gheini, Arman Kabiri, Rabeeh Karimi Mahabadi, Omid Memarrast, Ahmadreza Mosallanezhad, Erfan Noury, Shahab Raji, Mohammad Sadegh Rasooli, Sepideh Sadeghi, Erfan Sadeqi Azer, Niloofar Safi Samghabadi, Mahsa Shafaei, Saber Sheybani, Ali Tazarv, and Yadollah Yaghoobzadeh. 2020. Parsinlu: A suite of language understanding challenges for persian.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *ACL*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Yang Liu and Mirella Lapata. 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 (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar GuÌ‡lçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero Nogueira dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen McKeown, and Bing Xiang. 2021. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, Online. Association for Computational Linguistics.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1436–1441. AAAI Press.

Sampo Pyysalo, Jenna Kanerva, Antti Virtanen, and Filip Ginter. 2021. WikiBERT models: Deep transfer learning for many languages. In *Proceedings of the 23rd Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 1–10, Reykjavik, Iceland (Online). Linköping University Electronic Press, Sweden.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Hosein Rezaei, Seyed Amid Moeinzadeh, A. Shahgholian, and M. Saraee. 2019. Features in extractive supervised single-document summarization: Case of persian news. *ArXiv*, abs/1909.02776.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Behnam Sabeti, Hossein Abedi Firouzjaee, A. J. Choobbasti, S. J. Najafabadi, and Amir Vaheb. 2018. Mirastext: An automatically generated text corpus for persian. In *LREC*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Mahsa Sadat Shahshahani, Mahdi Mohseni, Azadeh Shakery, and Heshaam and Faili. 2019. Payma: A tagged corpus of persian named entities. *Signal and Data Processing*, 16(1).

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Gary Simons. 2017. *Ethnologue*. SIL International, Dallas.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Krysta Svore, Lucy Vanderwende, and Christopher Burges. 2007. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 448–457, Prague, Czech Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Y. Wu, M. Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, M. Krikun, Yuan Cao, Q. Gao, Klaus Macherey, J. Klingner, Apurva Shah, M. Johnson, X. Liu, Lukasz Kaiser, Stephan Gouws, Y. Kato, Taku Kudo, H. Kazawa, K. Stevens, George Kurian, Nishant Patil, W. Wang, C. Young, J. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, G. Corrado, Macduff Hughes, and J. Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mt5: A massively multilingual pre-trained text-to-text transformer. In *NAACL*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. 2020. Pre-training for abstractive document summarization by reinstating source text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3646–3660, Online. Association for Computational Linguistics.

A Datasets Statistics

In this section, extra information about downstream datasets and pre-training text corpora is reported. Some of the datasets did not provide any validation split; however, the number of examples in the train/validation/test split and the average length of articles and summaries for each dataset is reported in Table 11. Additionally, the size of pre-training texts corpora before and after preprocessing is reported in Table 9.

Following Grusky et al. (2018) and Zhang et al. (2020a), we have plotted the extractive fragment density/coverage plot for each downstream dataset in Figure 4. Grusky et al. (2018) defined them as

$$\text{COVERAGE}(\text{A}, \text{S}) = \frac{1}{|S|} \sum_{f \in \text{F}(\text{A},\text{S})} |\text{f}|$$

$$\text{DENSITY}(\text{A}, \text{S}) = \frac{1}{|S|} \sum_{f \in \text{F}(\text{A},\text{S})} |\text{f}|^2$$

where A is an article and S is the corresponding summary, and $F(A, S)$ is the set of shared sequences of tokens in A and S. The density for extractive summaries is higher than more abstractive summaries. Lower coverage shows the novelty of text fragments in summary. Figure 4 shows that our downstream datasets range from more extractive summaries to more abstractive ones.

Tebyan dataset contains articles and summaries from a well-known Persian lifestyle website that includes various articles from different categories. In order to produce the Tebyan dataset, we have crawled 100K pages of their site. We removed all HTML tags using *beautifulsoup4*[10] for each page, and each page's primary content was stored with the author's provided summary, and paragraphs were separated with a newline character. Lastly, we have used Langdetect[11] to remove articles that were not in Persian. After to this procedure, 92,289 articles and summaries were collected, so we separated them into three parts, 85% for train, 7.5% for validation, and 7.5% for test split.

We have tested ARMAN on NLU tasks with the ParsiNLU (Khashabi et al., 2020), which is a Persian NLU dataset. This dataset consists of 5 main tasks and translation as an extra task. In Table 10, the number of examples for train/validation/test of ParsiNLU is reported. It should be noted that we

[10] https://pypi.org/project/beautifulsoup4/

[11] https://pypi.org/project/langdetect/

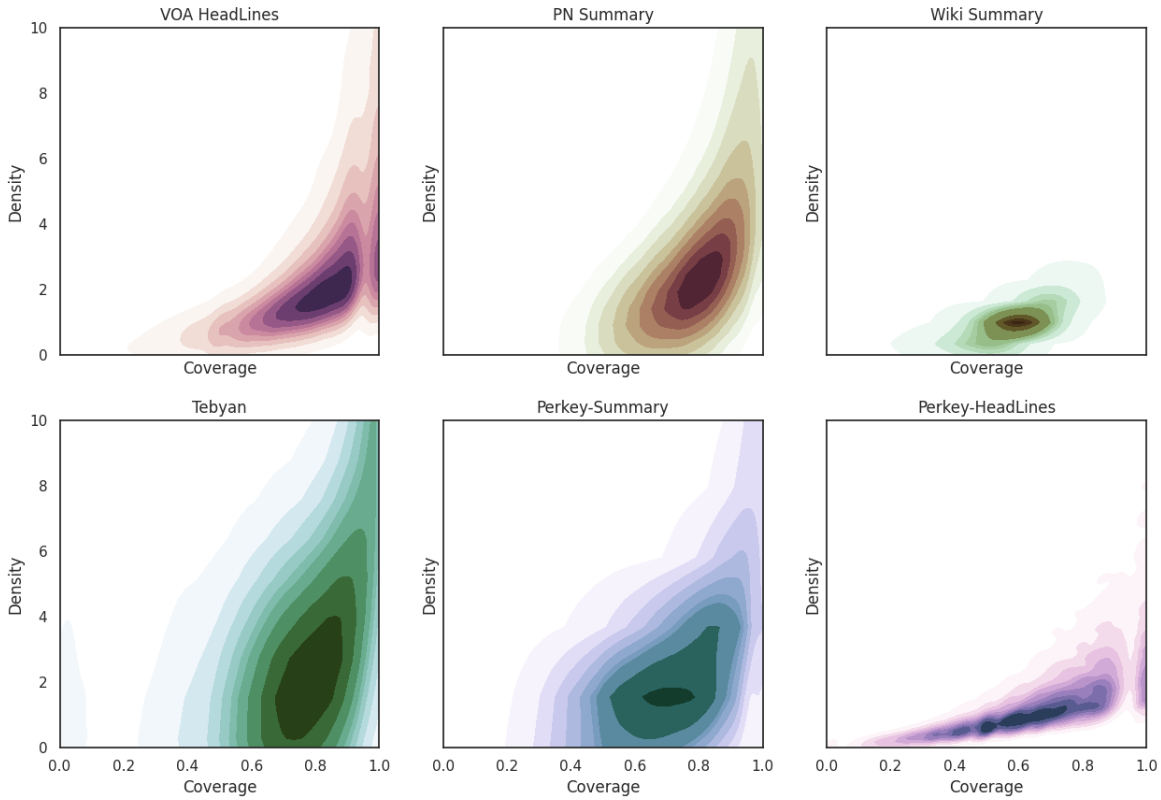

Figure 4: Density and coverage distributions across downstream datasets.

did not test ARMAN on the Reading Comprehension of this dataset due to resource leakage. The Sentiment Analysis task of this dataset has two subtasks; sentence-level sentiment and aspect-based sentiment of a sentence. We have tested ARMAN on the sentence-level sentiment task. The Sentiment Analysis task of this dataset has two subtasks; sentence-level sentiment and aspect-based sentiment of a sentence. We have tested ARMAN on the sentence-level sentiment task. For Question Paraphrasing and Textual Entailment, this dataset contains two subtasks; sentences written by humans and sentences translated from English datasets into Persian, so we have reported the accuracy of models for each subtask separately.

B ARMAN Hyper Parameters and Training Settings

In this section, we have described pre-training and fine-tuning parameters and settings. In Table 12, pre-training settings for $\text{ARMAN}_{\text{base}}$ are reported. Tables 13 and 14 contain information about settings used in fine-tuning $\text{ARMAN}_{\text{base}}$ and $\text{Transformer}_{\text{base}}$ on summarization tasks. Also, the fine-tuning settings for NLU tasks are reported in Table 15. Finally, we have reported each model's parameter counts used in summarization tasks in Table 16.

C Low Resource Numbers and Settings

Table 17 contains information about fine-tuning settings for low resource experiments in Section 5.5. The numbers in Figures 2 and 3 are reported in Table 18. We did not report the results of low resource experiments for ARMAN(SH), ARAMN(SS-80), ARMAN(SS-100), and PEGASUS in the main part of the paper, but they are reported in Tables 19, 20, 21, and 22.

D Samples

Two samples of ARMAN(SS), ARMAN(MSR), and PEGASUS generated summaries that were used in the human evaluation test are shown in Figures 5 and 6. More than 50% of participants believed that ARMAN(MSR)'s summaries were the best among all models in the human evaluation test, which shows that its summaries have high quality.

Pre-train Corpus/Dataset	Original Corpus/Dataset Size	Cleaned Corpus/Dataset Size
irBlogs	7.1GB	2.6GB
MirasText	15.7GB	6.8GB
YJC News	3GB	2.3GB
CC100	111GB	53GB
Total	136.8GB	64.7GB

Table 9: Size of pre-training text corpora in GB for each corpus before and after cleaning.

Task	Number of Train Examples	Number of Validation Examples	Number of Test Examples
Reading Comprehension	600	125	575
Multiple-Choice	1271	139	1050
Sentiment Analysis	1894	235	294
Textual Entailment	756	271	1751
Question Paraphrasing	1830	898	1916

Table 10: Task name and the number of examples for ParsiNLU dataset.

Dataset	Train Count	Validation Count	Test Count	Article Average Length	Summary Average Length
PN-Summary	82022	5592	5593	335	31
Wiki-Summary	45653	5073	5637	425	82
VOA	31550	3506	3896	179	11
Perkey (summary)	42077	-	19796	218	28
Perkey (title)	526445	-	24930	224	11
Tebyan	78445	6922	6922	819	37

Table 11: The number of articles and summaries and the average length of them for each downstream dataset (lengths are reported in words count).

Model	Learning rate	Label Smoothing	Steps	Batch Size	Objective	Max Input Length	Max Output Length
$\text{PEGASUS}_{\text{base}}$	0.01	0.0	1M	128	Ind-Orig	512	128
$\text{ARMAN(SS)}_{\text{base}}$	0.01	0.0	1M	128	TSS	512	128
$\text{ARMAN(SH)}_{\text{base}}$	0.01	0.0	1M	128	TSS+Shuffling	512	128
$\text{ARMAN(MSR)}_{\text{base}}$	0.01	0.0	1M	128	TTS+MSR	512	128

Table 12: Pre-training settings for $\text{ARMAN}_{\text{base}}$ models. We have used $\text{PEGASUS}_{\text{large}}$ (Zhang et al., 2020a) settings for maximum input and output length since they have searched for the best setting.

Dataset	Learning rate	Label Smoothing	Steps	Batch Size	Beam Size	Beam alpha	Max Input	Max Output
Perkey (summary)	5×10^{-4}	0.1	50K	128	8	0.8	512	256
Perkey (title)	5×10^{-4}	0.1	50K	128	8	0.8	512	256
PN-Summary	5×10^{-4}	0.1	50K	128	8	0.8	512	256
Tebyan	5×10^{-4}	0.1	50K	128	8	0.8	512	256
VOA	5×10^{-4}	0.1	20K	64	8	0.8	512	256
Wiki-Summary (v1)	5×10^{-4}	0.1	50K	64	8	0.8	512	256

Table 13: Fine-tuning settings for $\text{ARMAN}_{\text{base}}$ models on downstream summarization tasks and datasets.

Dataset	Learning rate	Label Smoothing	Steps	Batch Size	Beam Size	Beam alpha	Max Input	Max Output
Perkey (summary)	5×10^{-4}	0.1	150K	128	8	0.8	512	256
Perkey (title)	5×10^{-4}	0.1	150K	128	8	0.8	512	256
PN-Summary	5×10^{-4}	0.1	150K	128	8	0.8	512	256
Tebyan	5×10^{-4}	0.1	150K	128	8	0.8	512	256
VOA	5×10^{-4}	0.1	150K	64	8	0.8	512	256
Wiki-Summary (v1)	5×10^{-4}	0.1	150K	64	8	0.8	512	256

Table 14: Fine-tuning settings for $\text{Transformer}_{\text{base}}$ models on downstream summarization tasks and datasets.

Dataset	Learning rate	Label Smoothing	Steps	Batch Size	Beam Size	Beam alpha	Max Input	Max Output
Multiple-Choice	5×10^{-4}	0.1	20K	48	8	0.8	512	256
Sentiment Analysis	5×10^{-4}	0.1	20K	48	8	0.8	512	256
Textual Entailment	5×10^{-4}	0.1	20K	48	8	0.8	512	256
Question Paraphrasing	5×10^{-4}	0.1	20K	48	8	0.8	512	256

Table 15: Fine-tuning settings for $\text{ARMAN}_{\text{base}}$ models on NLU tasks. Batch size 48 was chosen to be the same as other models that were trained on those tasks. We have converted the classification problem into the text to text problems.

Model	Parameters	Transformer Type
$\text{ARMAN}_{\text{base}}$	223M	Vaswani et al. (2017)'s Encoder-Decoder
$\text{Transformer}_{\text{base}}$	223M	Vaswani et al. (2017)'s Encoder-Decoder
$\text{ParsBERT}_{\text{base}}$ (Farahani et al., 2020b)	221M	Rothe et al. (2020)'s Encoder-Decoder
$\text{PEGASUS}_{\text{base}}$ (Zhang et al., 2020a)	223M	Vaswani et al. (2017)'s Encoder-Decoder
$\text{mT5}_{\text{small}}$ (Xue et al., 2021)	300M	Vaswani et al. (2017)'s Encoder-Decoder

Table 16: Parameters count for each tested model that was used for summarization. Reported numbers are in millions.

Dataset	Learning rate	Label Smoothing	Steps	Batch Size	Beam Size	Beam alpha	Max Input	Max Output
Perkey (summary)	5×10^{-4}	0.1	2K	128	8	0.8	512	256
Perkey (title)	5×10^{-4}	0.1	2K	128	8	0.8	512	256
PN-Summary	5×10^{-4}	0.1	2K	128	8	0.8	512	256
Tebyan	5×10^{-4}	0.1	2K	128	8	0.8	512	256
VOA	5×10^{-4}	0.1	2K	64	8	0.8	512	256
Wiki-Summary (v1)	5×10^{-4}	0.1	2K	64	8	0.8	512	256

Table 17: Fine-tuning settings for $\text{ARMAN}_{\text{base}}$ and $\text{PEGASUS}_{\text{base}}$ models on downstream summarization tasks and datasets for low resource experiments.

examples	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
0	25.28/11.09/19.18	24.14/5.05/13.89	26.69/13.89/23.3	23.83/11.39/19.22	16.65/6.26/13.88	20.93/9.25/15.85
10	34.01/17.19/27.83	24.48/5.22/15.09	29.85/15.82/26.39	34.66/19.94/29.65	18.96/7.31/16.13	25.03/9.95/19.4
100	38.47/20.71/32.32	27.87/7.24/18.76	40.35/23.07/36.38	40.24/25.7/35.67	28.81/12.37/26.03	28.95/13.33/23.32
1K	40.96/22.66/34.78	29.86/9.03/21.1	44.67/26.08/40.42	43.04/28.01/38.42	31.43/14.32/28.61	32.42/16.33/26.55
10K	43.21/24.85/37.07	30.09/10.73/22.89	46.38/27.82/42.48	45.43/30.71/41	35.18/17.83/32.32	35.17/19.2/29.36

Table 18: Low resource results of ARMAN(MSR) from Figures 2 and 3. By less than 1000 examples, ARMAN(MSR) has beaten the previous SOTA on VOA and Wiki-Summary datasets. Also, 10K examples and 2K fine-tuning steps got comparable results with previous SOTA in the Pn-Summary dataset.

examples	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
0	24.08/10.34/18.37	22.73/4.58/13.24	27.13/14.14/23.67	23.91/12.07/19.6	16.69/6.36/13.9	20.1/8.74/15.23
10	34.71/17.63/28.51	24.6/5.43/15.27	33.88/18.5/30.06	38.27/23.88/33.61	22.93/8.97/20.15	25.91/11.4/20.27
100	38.67/20.67/32.49	27.41/7.42/19.03	41.05/23.39/37.03	41.34/26.31/36.58	26.83/11.28/24.09	29.13/13.49/23.52
1K	40.95/22.78/34.7	30/8.68/20.75	44.22/25.11/39.77	43.11/28.06/38.45	31.2/14.17/28.28	33.1/17.24/27.31
10K	43.07/24.84/37.05	29.83/10.43/22.58	46.8/27.87/42.86	45.19/30.43/40.76	34.79/17.53/31.83	34.71/18.83/28.97

Table 19: Low resource results of ARMAN(SH). By less than 1000 examples, ARMAN(SH) has beaten the previous SOTA on VOA and Wiki-Summary datasets. Also, 10K examples and 2K fine-tuning steps got comparable results with previous SOTA in the Pn-Summary dataset.

examples	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
0	18.92/7.96/15.04	21.87/4.14/13.22	21.8/10.81/19.02	17.19/7.36/14.02	13.33/4.8/11.31	17.67/7.05/13.7
10	37.1/19.18/30.54	24.84/5.99/16.45	33.84/17.6/30.09	35.36/21.14/30.58	25.17/10.47/22.21	27.74/12.97/22.39
100	39.26/21.2/33.21	27.54/7.28/18.89	41.17/23.15/37.31	40.6/25.89/36.22	28.54/12.23/25.72	30.83/15.42/25.33
1K	40.51/22.38/34.43	29.75/8.66/20.72	44.09/25.51/39.9	42.64/27.58/38.05	30.73/13.87/27.89	32.27/16.31/26.49
10K	43.03/24.82/36.91	29.36/10.2/22.33	46.88/27.96/42.91	44.94/30.18/40.51	34.53/17.31/31.65	34.78/18.74/28.94

Table 20: Low resource results of ARMAN(SS-80). By less than 1000 examples, ARMAN(SS-80) has beaten the previous SOTA on VOA and Wiki-Summary datasets. Also, 10K examples and 2K fine-tuning steps got comparable results with previous SOTA in the Pn-Summary dataset.

examples	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
0	35.53/18.91/29.78	18.86/3.42/14.4	26.18/11.51/22.75	36.15/20.89/31.37	19.58/7.34/16.73	22.04/9.67/19.04
10	38.49/20.79/32.49	24.05/6.47/17.59	33.45/16.42/29.66	38.55/23.05/33.72	24.85/10.18/21.74	28.53/14.13/23.84
100	39.26/21.19/33.17	27.76/7.46/19.31	41.52/22.97/37.52	40.71/25.27/35.84	28.63/12.27/25.7	30.32/14.29/24.73
1K	41.25/22.89/35.16	30.15/8.88/21.01	44.88/25.58/40.67	42.97/27.75/38.36	31.38/14.32/28.52	32.85/17.02/27.26
10K	43.51/25.28/37.42	29.48/10.16/22.31	46.98/28.33/43.07	45/30.08/40.52	35.29/17.95/32.42	34.95/19/29.22

Table 21: Low resource results of ARMAN(SS-100). By less than 1000 examples, ARMAN(SS-100) has beaten the previous SOTA on VOA and Wiki-Summary datasets. Also, 10K examples and 2K fine-tuning steps got comparable results with previous SOTA in the Pn-Summary dataset.

examples	PN-Summary R1/R2/RL	Wiki-Summary R1/R2/RL	VOA R1/R2/RL	Perkey(summary) R1/R2/RL	Perkey(title) R1/R2/RL	Tebyan R1/R2/RL
0	25.32/11.25/19.45	23.11/4.49/13.24	21.51/9.56/18.36	23.34/10.41/18.68	12.93/4.2/10.81	19.27/8.16/14.64
10	35.18/17.65/28.99	24.06/5.49/16.36	30.14/14.83/26.5	34.49/19.2/29.36	16.75/6.14/14.29	26.81/11.39/20.98
100	37.94/20/31.71	27.27/6.81/18.32	40.23/22.68/36.46	39.97/25.06/35.4	26.47/11.08/23.75	28.86/13.12/23.27
1K	39.91/21.88/33.82	29.46/8.61/20.61	42.67/23.8/38.53	42.11/27.09/37.48	29.97/13.46/27.19	31.87/16.01/26.2
10K	42.27/24.06/36.2	29.3/10.13/22.22	46.04/27.15/41.91	44.72/29.97/40.37	33.98/16.93/31.11	34.59/18.73/28.91

Table 22: Low resource results of PEGASUS.

Document	به گزارش ایرنا، ترامپ ساعت پیش در پیام توییتری نوشت: این افتخار بزرگ من است که اعلام کنم به ژنرال مایکل فلین عفو کامل اعطا شده است، تبریک به فلین و خانواده شگفت انگیز او، من می‌دانم که شما اکنون یک مراسم شکرگزاری واقعا خارق العاده خواهید داشت! فلین به عنوان اولین مشاور امنیت ملی ترامپ به حساب می‌آید که طبق اعتراف‌های خود درباره انجام تماس‌هایی با روسیه در سال ۲۰۱۷ به اف بی آی دروغ گفت. مذاکرات مخفیانه فلین با سفیر روسیه در واشنگتن در سال ۲۰۱۶ یعنی قبل از سوگند یاد کردن ترامپ در منصب رئیس جمهوری، اساس تحقیقات رابرت مولر بازپرس ویژه پرونده اتهام مداخله مسکو در انتخابات ریاست جمهوری ۲۰۱۶ آمریکا بود. ترامپ پس از اطلاع از این موضوع بلافاصله فلین را اخراج کرد. با این حال، همواره مدعی شد که این پرونده تحقیقاتی، یک حقه سیاسی است و فلین که ژنرال سابق ارتش و رئیس آژانس اطلاعات دفاعی آمریکا بوده، فرد خوبی است. در همین حال، وزارت دادگستری آمریکا در اقدامی غیرمعمول، پرونده علیه فلین را مختومه اعلام کرد و یادآور شد که دروغ‌های فلین به اف بی آی، قابل توجه نبوده است، اقدامی که برای ترامپ یک پیروزی بزرگ سیاسی به ارمغان آورد. اما یک قاضی فدرال خواستار بررسی قضایی بیشتر درمورد این مساله شد. مایکل فلین سپهبد بازنشسته نیروی زمینی ایالات متحده آمریکا، فقط به مدت ۲۴ روز به عنوان مشاور امنیت ملی ترامپ فعالیت کرد. فلین در فوریه ۲۰۱۷ (بهمن ۹۵) به دلیل اتهام‌های مطرح شده درخصوص حواشی تحریم‌های آمریکا علیه مسکو و ارتباطات او با سفیر روسیه، از سمت خود استعفا کرد.
ARMAN(SS)	دونالد ترامپ رئیس جمهوری آمریکا چهارشنبه شب اعلام کرد که به ژنرال مایکل فلین مشاور امنیت ملی خود عفو کامل داده است.
ARMAN(MSR)	دونالد ترامپ رئیس جمهوری آمریکا یکشنبه شب اعلام کرد که به مایکل فلین مشاور سابق امنیت ملی خود که طبق اعتراف های خود درباره انجام تماس هایی با روسیه در سال ۲۰۱۷ به اف بی آی دروغ گفت، عفو کامل اعطا کرده است.
PEGASUS	دونالد ترامپ رئیس جمهوری آمریکا یکشنبه شب با تبریک به ژنرال مایکل فلین مشاور سابق امنیت ملی خود و خانواده اش، اعلام کرد که اکنون به افتخار او و خانواده اش یک مراسم شکرگزاری فوق العاده برگزار خواهد شد.

Figure 5: The First sample of models' generated summaries in human evaluation tests.

Document	به گزارش شانا به نقل از شرکت ملی گاز ایران، روح‌الله نوریان، مدیر فناوری اطلاعات و ارتباطات این شرکت درباره راه‌اندازی پرتال سامگ گفت: راه‌اندازی کامل سامانه سامگ در ۳۱ شرکت گاز استانی، گام بلندی در جهت تحقق دولت الکترونیک و ارتباطات G2G (ارتباط بین دولت‌ها) و G2C (ارتباط دولت با شهروندان) بوده و موجب تسهیل ارائه خدمات به مشترکان گاز می‌شود. وی افزود: با راه‌اندازی این سامانه و یکپارچه کردن فرآیندهای مربوط به حوزه مشترکان گاز در سطح شرکت‌های گاز استانی، امکان مدیریت و پایش کیفیت ارائه خدمات و فرآیندهای وابسته به آن نیز فراهم می‌شود. بر اساس این گزارش، جلسه آموزش راه‌اندازی این سامانه در روزهای ۲۸ و ۲۹ مهر ماه امسال در این شرکت برگزار شد. سیدجلال نورموسوی، رئیس امور تعرفه‌ها و قراردادهای مدیریت گازرسانی شرکت ملی گاز ایران که در این جلسه حضور داشت، درباره اجرای این سامانه اظهار کرد: با اجرای این طرح، همه خدمات مورد نیاز مردم و مشترکان گاز از طریق سامانه‌های الکترونیکی و دفاتر پیشخوان به صورت شفاف، یکسان، قانون‌مند و بر اساس تعرفه‌های مقرر انجام می‌شود. وی ابراز امیدواری کرد با راه‌اندازی کامل سامانه سامگ، توسعه خدمات قابل ارائه در دفاتر پیشخوان دولت به بهترین وضع ممکن برسد.
ARMAN(SS)	پرتال سامگ (سامانه الکترونیکی خدمات گازرسانی) با هدف ارائه خدمات الکترونیکی به مشترکان گاز در شرکت های گاز استانی راه اندازی شد.
ARMAN(MSR)	سامانه «سامگ» با هدف یکپارچه سازی فرآیندهای مربوط به حوزه مشترکان گاز در سطح شرکت های گاز استانی، در شرکت ملی گاز ایران راه اندازی شد.
PEGASUS	راه اندازی کامل سامانه سامگ در دفاتر پیشخوان دولت و دفاتر ict روستایی، امکان مدیریت و پایش کیفیت ارائه خدمات و فرآیندهای وابسته به آن را فراهم می کند.

Figure 6: The Second sample of models' generated summaries in human evaluation tests.