IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 11, 2022, accepted February 26, 2022, date of publication March 8, 2022, date of current version March 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3157289

PersianQuAD: The Native Question Answering Dataset for the Persian Language

AREFEH KAZEMI[1], JAMSHID MOZAFARI[2], AND MOHAMMAD ALI NEMATBAKHSH[2]
[1]Department of Linguistics, Faculty of Foreign Languages, University of Isfahan, Isfahan 81746-73441, Iran
[2]Big Data Research Group, Faculty of Computer Engineering, University of Isfahan, Isfahan 81746-73441, Iran

Corresponding author: Arefeh Kazemi (akazemi@fgn.ui.ac.ir)

ABSTRACT Developing Question Answering systems (QA) is one of the main goals in Artificial Intelligence. With the advent of Deep Learning (DL) techniques, QA systems have witnessed significant advances. Although DL performs very well on QA, it requires a considerable amount of annotated data for training. Many annotated datasets have been built for the QA task; most of them are exclusively in English. In order to address the need for a high-quality QA dataset in the Persian language, we present PersianQuAD, the native QA dataset for the Persian language. We create PersianQuAD in four steps: 1) Wikipedia article selection, 2) question-answer collection, 3) three-candidates test set preparation, and 4) Data Quality Monitoring. PersianQuAD consists of approximately 20,000 questions and answers made by native annotators on a set of Persian Wikipedia articles. The answer to each question is a segment of the corresponding article text. To better understand PersianQuAD and ensure its representativeness, we analyze PersianQuAD and show it contains questions of varying types and difficulties. We also present three versions of a deep learning-based QA system trained with PersianQuAD. Our best system achieves an F1 score of 82.97% which is comparable to that of QA systems on English SQuAD, made by the Stanford University. This shows that PersianQuAD performs well for training deep-learning-based QA systems. Human performance on PersianQuAD is significantly better (96.49%), demonstrating that PersianQuAD is challenging enough and there is still plenty of room for future improvement. PersianQuAD and all QA models implemented in this paper are freely available.

INDEX TERMS Dataset, deep learning, natural language processing, Persian, question answering, machine reading comprehension.

I. INTRODUCTION

Developing open-domain Question Answering systems (QA) is one of the main goals in Artificial Intelligence. QA systems receive the users' questions in natural language and respond to them with precise answers. They deploy natural language understanding and information retrieval to understand the users' questions and find the appropriate answers. Classic search engines receive the users' queries and return a list of relevant web pages. The user should read the returned web pages to obtain their required information. QA systems receive the users' questions and find the final answer to that questions. Nowadays, modern search engines such as Google, Yahoo, and Bing, deploy QA techniques to provide precise responses to some types of questions. For example, if someone searches for the question *"what is the normal body temperature?"* in Google, it responds with the precise answer: *"37 degrees Celsius"*.

QA systems mainly focus on factoid questions; questions that can be answered with facts expressed in a few words [1]. Table 1 shows three examples of factoid questions, along with their answers and answers' types.

With the advent of Deep Learning (DL) techniques, NLP tasks such as machine translation, sentiment analysis, and QA have witnessed a significant advance. Although DL performs very well on NLP tasks, they require a considerable amount of annotated data for training. In the QA task, the data are semantically annotated and in the form *"question, paragraph, answer"*, where the question, paragraph, and answer, show a question expressed in natural language, a text that contains the answer, and a span of the paragraph which has the answer, respectively. Given the question and the paragraph,

The associate editor coordinating the review of this manuscript and approving it for publication was Abdel-Hamid Soliman.

TABLE 1. Examples of factoid questions.

Facotoid Question	Answer	Answer Type
who is the co-founder of Apple?	Steve Jobs	Person
when was Mahatma Gandhi born?	October 2, 1869	Date
where is the longest suspension bridge?	Japan	Location

the QA system should select the paragraph's span, which is the answer.

Many annotated datasets have been built for the QA task; most of them are exclusively in English. The most famous QA dataset in English is SQuAD [2]. It contains 100,000+ questions created by crowdworkers on a set of Wikipedia articles. The answer to each question is specified in the article text. In order to conduct QA systems for other languages, some works have automatically translated the English QA datasets to the target language using machine translation tools. Even though translating English QA datasets is a fast and straightforward solution to prepare labeled data for low-resourced languages, automatic translation is not perfect and can not produce high-quality annotated data. In addition to this, the process of translation -even human translation- may produce faulty outputs in the target language [3]. The result is that the quality of the QA systems trained on the purely native datasets is substantially better than the quality of the QA systems trained on the translation datasets [3]–[6].

In order to address the need for a high-quality QA dataset for Persian language, we propose a model for creating dataset for deep-learning-based QA systems. We use the proposed model to create PersianQuAD: The Native Question Answering Dataset for the Persian Language. PersianQuAD contains 20,000 "question, paragraph, answer" triplets and is the first large-scale native QA dataset for the Persian language, to the best of our knowledge. PersianQuAD and also the code of all QA systems implemented in this article are freely available for public use at https://github.com/BigData-IsfahanUni/PersianQuAD. To evaluate the quality of the QA dataset created through the proposed model, we implemented a set of state-of-the-art deep-learning-based QA systems and used the created QA dataset for training these systems. Our best model achieves an $F1$ score of 82.97% and an Exact Match of 78.7%, which are comparable with that of English QA systems trained on the English SQuAD, made by the Stanford University.

The remainder of this paper is organized as follows. Section II reviews the related work and puts our work in its proper context. Sections III, IV, and V present in detail the proposed model for preparing QA datasets, including the problem definition, the annotation tool used, and dataset collection process. This is followed by more in-depth analyses of the resulted dataset in Section VI. Section VII contains the experiments carried out to evaluate the quality of the resulted dataset by using it for training three deep-learning-based QA systems. Finally, we outline conclusions in Section VIII.

II. RELATED WORK

Many QA datasets have been produced in the past decade, most of them exclusively in English. In recent years, several QA datasets have been built for other languages, such as Arabic, France, etc. In this section, we present a brief review of the researches on creating QA datasets.

A. ENGLISH

The Stanford Question Answering Dataset (SQuAD) [2] can be considered as the most famous QA dataset in English. It consists of 100,000+ questions posed by crowdworkers on Wikipedia articles. A set of Wikipedia articles were presented to the annotators, and they were asked to pose some questions on the paragraph and specify the corresponding answer. The second version of SQuAD [7] contains over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones.

The WikiQA [8] and MS Marco [9] datasets are built by sampling questions searched by the users in the Bing search engine. For each question, the top-ten documents returned by the Bing engine are presented to the annotators and ask them to find the answer to the question in the documents or say that the documents do not contain the answer. WikiQA contains about 3,000 questions and their corresponding answer sentence on the Wikipedia page. MsMarco contains 100,000 questions with free-form answers. Natural Questions (NQ) dataset [10] is created by sampling questions issued to the Google search engine and consists of over 300,000 examples. To create NQ dataset, a set of questions searched in Google along with the top-five results returned by Google are presented to the annotators. Then the annotators are asked to specify the answer on the pages or mark null if the answer is not inside the returned pages.

The QuAC [11] and CoQA [12] are conversational QA datasets that contain dialogues between a questioner and an answerer. CoQA [12] contains over 127,000 question-answer pairs. The questions are collected by asking two crowdworkers to chat about a passage. During the conversation, one of the crowdworkers poses some questions about the passage, and the other tries to answer them. The questions are conversational, and the answers are free-form text with their corresponding evidence highlighted in the passage. NewsQA [13] is a QA dataset based on the CNN news articles, with over 100,000 question-answer pairs. NewsQA is created in three stages, and the crowdworkers are divided into three groups: 1) questioners, 2) answerers, and 3) validators. In the first stage, the questioners see only the article highlights and headlines and pose some questions. In the second stage, the answerers see the crowdsourced questions and the full article and determine the answer in the article. In the third stage, the validators see the article, the question, and a set of unique answers to that question selected by the answerers. The validators then choose the best answer from the candidate set or reject all of them.

Some QA systems translate the users' question from natural language into a query formulated through a specific data query language that is compliant with the underlying knowledge base. MQALD [14] provides a dataset for evaluating the performance of the QA systems in translating the questions from natural language into a specific data query language.

Some questions can not be answered by reading a single paragraph or document, and several pieces of information in different documents should be considered to find the answer. These questions need multi-step(or multi-hop) reasoning. Several QA datasets have been developed to address multi-step QA. The main challenge in multi-step QA datasets is to answer the questions by reasoning over different documents. QAngaroo [15], HotpotQA [16], ComplexWebQuestions [17], and R4C [18] are examples of multi-hop datasets. ComplexWebQuestions and QAngaroo are built by incorporating a knowledge base with the Web and Wikipedia website documents. HotpotQA and R4C are created by crowdsourcing. HotpotQA consists of 113,000 questions on Wikipedia articles. Answering each question in the dataset needs finding multiple documents and reasoning over them. In multi-hop datasets, in addition to answering questions, QA systems typically should also identify paragraphs that have been used to drive the answer.

B. OTHER LANGUAGES

There are two main approaches for building QA datasets in languages other than English: 1) translating English QA datasets into the target language, typically using machine translation, and 2) building the dataset from scratch (native QA dataset). In the first approach, an English QA dataset's training set is first translated into the target language. Then the QA system for the target language is trained on the translated dataset. Carrino *et al.* [19] proposed a new method to automatically translate SQuAD to Spanish and used the translated dataset to fine-tune a Spanish QA model. Mozannar *et al.* [20] translated 48,000+ SQuAD instances into Arabic using machine translation and built an Arabic QA system. Lee *et al.* [21] translated SQuAD into Korean and built k-QuAD using machine translation. Croce *et al.* [22] proposed a semi-automatic translation method and translated SQuAD into Italian. While the translation approach is a fast and relatively easy way to building QA datasets for low-resourced languages, Clark *et al.* [3] argue against this approach. They discuss that the translation process, even human translation, tends to produce problematic artifacts in the output language, such as preserving the word order of the source language when translating to a target language with free-word order or using more formal in the target language. As a consequence, the text that is obtained from human- or automatic-translation may be significantly different from original native text [6], [23]. In addition to this, speakers in various languages and of different nationalities may have questions about different topics [3] and in different ways. For example, a Persian speaker may ask about the recipe of ''Ghormeh sabzi'', a traditional Iran food. These types of questions may never appear in the translated datasets. These issues encourage following the second approach, i.e., building native QA datasets from scratch by native annotators. In this work, we follow the second approach for creating the PersianQuAD dataset.

Native QA datasets are mostly constructed in a similar way to SQuAD. The SberQuAD [24] is a Russian native QA dataset and contains above 50,000 samples. The DRCD [25] is a native Chinese QA dataset, consists of 30,000+ questions posed by the annotators on 10,014 paragraphs extracted from 2,108 Wikipedia articles. KorQuAD [26] is a Korean QA dataset and PIAF [27] is a French QA dataset, consisting of 70,000+ and 3835 question-answer pairs, respectively. Since large-scale QA datasets in languages other than English rarely exist and building native QA datasets is time- and cost-consuming, developing QA systems for these languages is challenging. Cross-lingual QA datasets have been developed to address this challenge. These datasets are typically deployed in training the QA model on one language and transfer the model to another language. It has been shown that the resulted models perform well in the zero-shot setting [28]. MLQA [29] is a cross-lingual QA dataset developed for seven languages: English, Arabic, German, Spanish, Hindi, Vietnamese and Chinese. It consists of over 12,000 samples in English and 5,000 samples in other languages. MMQA [30] is a parallel QA dataset in Hindi and English, containing 5,000+ parallel instances. BiPar [31] is another parallel QA dataset in English and Chinese. XQA [32] consists of a training set in English and the development and test sets in eight languages: English, French, German, Portuguese, Polish, Chinese, Russian, Ukrainian, and Tamil. XQuAD contains 1190 instances from SQuAD, along with their translations in 10 languages.

There are only a few works on building open-domain QA datasets for the Persian language. Abadani *et al.* [33] automatically translated SQuAD into the Persian language and built a translated Persian QA dataset, called ParSQuAD. They created two versions of ParSQuAD: ParSQuAD(manual) and ParSQuAD(automatic), with 25000 and 70000 instances, respectively. For creating ParSQuAD(manual), after automatic translation of SQuAD, some of the translation errors have been corrected manually. ParSQuAD(automatic) is the result of automatic translation of a part of SQuAD, without any manual correction on the translation. As we discussed earlier, the translation approach for creating a QA dataset has a number of limitations. Hence, in this paper, we create a native QA dataset for the Persian language. Khashabi *et al.* [34] created a Persian QA dataset containing 1300 instances and trained a QA system using this dataset. To the best of our knowledge, currently, there is no native large-scale QA dataset for answering the Persian questions, neither as a monolingual nor as a cross-lingual dataset. In this paper, we present the first large-scale and native dataset for the Persian language, called PersianQuAD.

A review of the discussed QA datasets is presented in Table 2.

TABLE 2. Review of some QA datasets.

Title	Year	Language(s)	Type	Size
Squad: 100,000+ questions for machine comprehension of text [2]	2016	English	Native	100K+
Know what you don't know: Unanswerable questions for SQuAD [7]	2018	English	Native	150K+
Wikiqa: A challenge dataset for open-domain question answering [8]	2015	English	Native	3K+
MS MARCO: A human generated machine reading comprehension dataset [9]	2016	English	Native	100K+
Natural questions: a benchmark for question answering research [10]	2019	English	Native	300K+
Quac: Question answering in context [11]	2018	English	Native	100K+
Coqa: A conversational question answering challenge [12]	2019	English	Native	127K+
Newsqa: A machine comprehension dataset [13]	2016	English	Native	100K+
Constructing datasets for multi-hop reading comprehension across documents [15]	2018	English	Native, Multi-hop	50K+
Hotpotqa: A dataset for diverse, explainable multi-hop question answering [16]	2018	English	Native, Multi-hop	113K+
Repartitioning of the complexwebquestions dataset [17]	2018	English	Native, Multi-hop	63K+
R4C: A benchmark for evaluating RC systems to get the right answer for the right reason [18]	2019	English	Native, Multi-hop	4K+
Automatic spanish translation of the squad dataset for multilingual question answering [19]	2019	Spanish	Translation	100K+
Neural arabic question answering [20]	2019	Arabic	Translation	48K+
Semi-supervised training data generation for multilingual question answering [21]	2018	Korean	Translation	81K+
Neural learning for question answering in italian [22]	2018	Italian	Translation	60K+
SberQuAD–Russian reading comprehension dataset: Description and analysis [24]	2020	Russian	Native	50K+
Drcd: a chinese machine reading comprehension dataset [25]	2018	Chinese	Native	30K+
Korquad1. 0: Korean qa dataset for machine reading comprehension [26]	2018	Korean	Native	70K+
Project PIAF: Building a Native French Question-Answering Dataset [27]	2020	French	Native	3K+
Parsinlu: a suite of language understanding challenges for persian [34]	2021	Persian	Native	1K+
ParSQuAD: Persian Question Answering Dataset based on Machine Translation of SQuAD 2.0 [33]	2021	Persian	Translation	25K, 70K

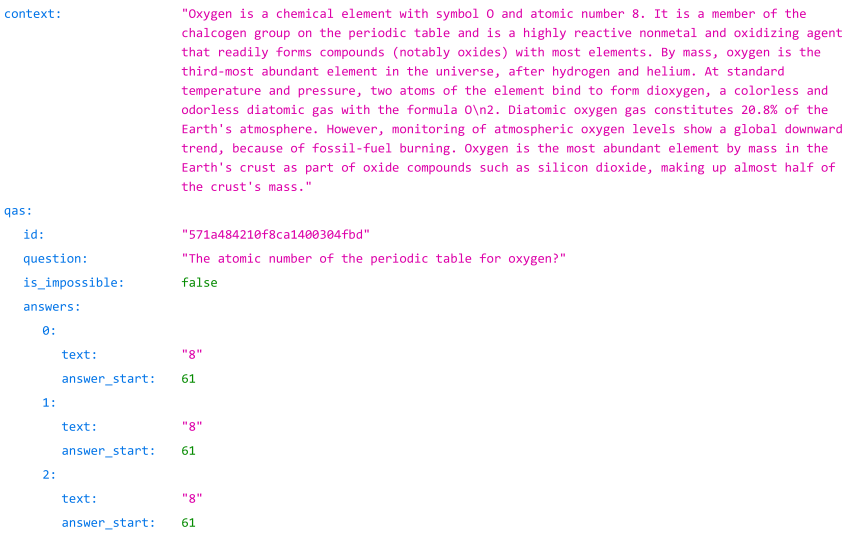

FIGURE 1. An instance of SQuAD dataset.

III. PROBLEM DEFINITION

In the QA systems, given a question and a paragraph containing the answer, the model needs to find the answer to that question in the paragraph text. The answer is determined by an *"start"* and an *"end"* token, which show the start and end tokens of the answer in the paragraph text, respectively.

To define the QA problem formally, assume that the question $Q = \{q_1, q_2, \ldots, q_n\}$ and the paragraph $P = \{p_1, p_2, \ldots, p_m\}$ are given, where q_i and p_j are i'th and j'th words of the question and paragraph text, respectively. The QA system needs to find the answer $A = \{a_1, \ldots, a_k\}$ of the question Q in paragraph P. The paragraph P contains the answer, hence, $A = \{a_1, \ldots, a_k\} = \{p_{start}, p_{start+1}, \ldots, p_{end=start+k-1}\}$, where *start* and *end* are the start and end tokens of the answer in the paragraph text. The QA system finds the answer of the question, by predicting the *start* and *end* tokens on condition that $start < end$. To be more precise, the QA systems specify the answer by estimating $Pr(start, end|P, Q)$, where Pr shows the probability.

Modern QA systems employ DL techniques to solve the stated problem, i.e., to predict the start and end tokens of the answer in the corresponding paragraph text. DL algorithms require a large number of annotated data. As mentioned earlier, for the QA task, the annotated data are in the form (Q, P, A) where Q, P, and A are the question, the paragraph,

FIGURE 2. SAJAD's annotation page.

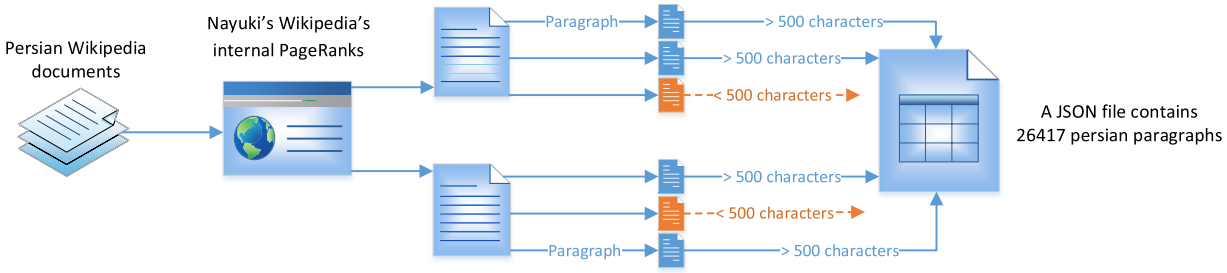

FIGURE 3. Wikipedia article selection.

and the answer, respectively. Usually, the QA datasets are stored in JSON format. Fig. 1 shows an instance of the SQuAD dataset in JSON format.

In order to build high-quality QA systems by using DL techniques, a large labeled dataset is required. While these datasets are available for a limited number of languages, there is no large-scale and native QA dataset for most of the non-English languages, such as the Persian language. In this paper we build the first native Persian question answering dataset. In order to create the dataset, we first implement SAJAD, an effective annotation tool for gathering QA datasets (Section IV). Then we collect the data through a participatory approach on Wikipedia articles (Section V). We build three QA systems and employ PersianQuAD for training of these systems in Section VII. In the rest of the paper, we explain the mentioned steps in detail.

IV. SAJAD ANNOTATION TOOL

In Fig. 2 the *Document Title* shows the title of the Wikipedia page that the paragraph is extracted from. The *Paragraph* shows the paragraph on which the annotator should pose a question. The annotator pose a question on the paragraph and type it in the *Question* field. The annotator then specifies the answer to the question by highlighting the answer within the paragraph text. By highlighting the answer within the text, it will be automatically placed in the *Answer* field. The annotator adds the question to the JSON file of the PersianQuAD by the *Add* button. In the case the annotator can not pose any

TABLE 3. Example of a paragraph, a question, and the corresponding answer.

Paragraph	Mohammad-Reza Shajarian was an Iranian vocalist and master of Persian traditional music. *{Shajarian}* was also known for his skills in Persian calligraphy and humanitarian activities. In 1999, UNESCO in France presented him with the Picasso Award and in 2006 with the UNESCO Mozart Medal. In 2017, Los Angeles Times cited him as the "Greatest living maestro of Persian classical music"
Question	Which Iranian vocalist is also known for his skills in calligraphy?
Answer	Mohammad-Reza Shajarian

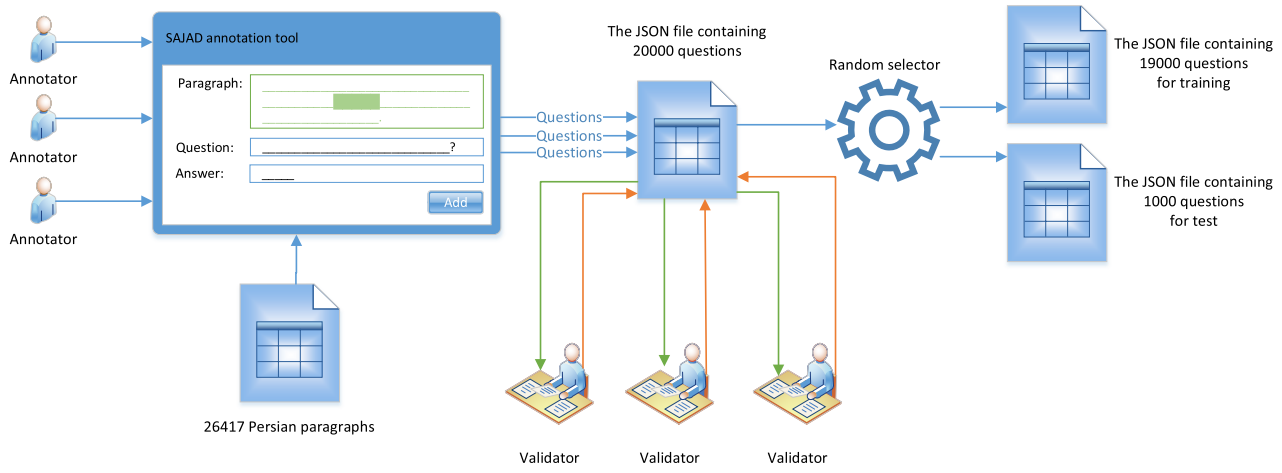

FIGURE 4. Question-answer collection.

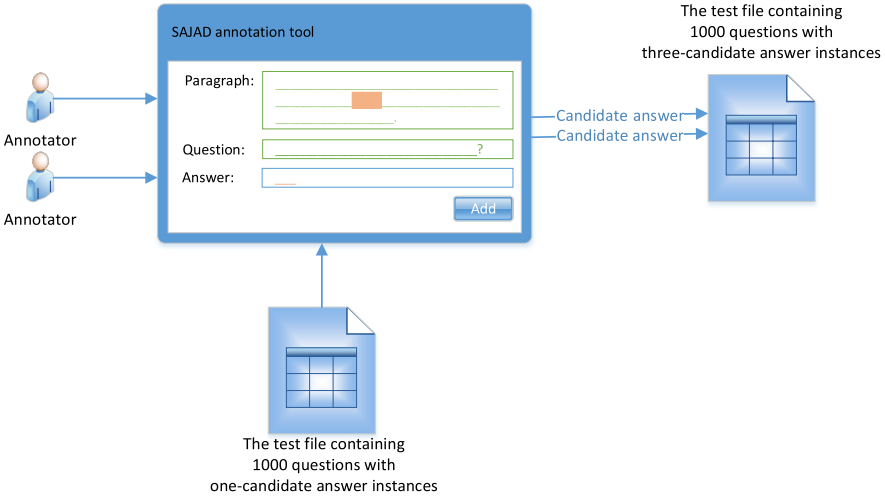

FIGURE 5. Three-candidates test set preparation.

question on a paragraph, s/he can go to the next paragraph by clicking the *Skip* button.

The following features characterize SAJAD:

- *Web-based and Mobile-friendly:* SAJAD is a web-based application, and hence it does not need to be installed on the participants' systems. It also can be accessed easily from any device and browser.
- *Multi Account:* SAJAD is a multi-user tool. It enables the system administrator to create multiple accounts. Each participant is provided with an account, and the participants' IDs will be stored in the questions they pose. This enables the system administrators to actively evaluate the quality of questions that the annotators posed.
- *SQuAD Format:* To enable the community to use the dataset easily and quickly, the SAJAD input and output are the same as SQuAD. Each paragraph can have many questions, and each question can have multiple answers.

V. DATASET COLLECTION

In line with recent works on QA tasks, we follow the same format as in SQuAD and other QA datasets for our dataset. The data in QA datasets is in the form (Q, P, A), where Q is the question, P is the paragraph that contains the answer, and A is the answer to the question. As described in Section IV, we develop SAJAD and use it for gathering the QA data. Our proposed model for collecting QA datasets consists of four steps: 1) Wikipedia article selection, 2) question-answer collection, 3) three-candidates test set preparation, and 4) Data Quality Monitoring.

A. WIKIPEDIA ARTICLE SELECTION

In this step, we selected a set of high-quality Persian Wikipedia articles to use in the dataset collection process. We used Project Nayuki's Wikipedia's internal PageRanks to retrieve the top 10,000 articles of Persian Wikipedia with the highest page rank. We extracted the individual paragraphs from the selected article and kept only the paragraphs with more than 500 characters. Ultimately, we got 26,417 paragraphs with different topics and converted them to the JSON input format to be used as the SAJAD input. Fig. 3 shows the article selection process.

B. QUESTION-ANSWER COLLECTION

We employed several participants to make questions. Most of them are graduated, or fourth-year students in linguistics and computer science, and all of them are native Persian speakers. We first provided the annotators with a set of written guidelines and oral explanations about the annotation process, the level of complexity that the questions should have, the answers span, and the shortest span. The annotators were also shown a set of example paragraphs and some good and bad questions and answers on each paragraph. Then the annotators were asked to make 50 questions, and if they pose at least 45 questions correctly, they could participate in the annotation process.

We used the SAJAD tool for annotation, as explained in Section IV. In the annotation process, the annotators are shown a random paragraph from the selected articles. Then they were asked to pose some questions on the paragraph and highlight the answer within the paragraph text. The participants were asked to spend one minute making each question and posing at least three questions in each paragraph. They were also asked to make questions without using the paragraph's text and express it in their own words. In the case that the participants can not make any question on a specific paragraph, they can skip it. Fig. 4 shows the process of question-answer collection. In this way, we collected about 20,000 question-answer and stored them in the JSON format.

C. THREE-CANDIDATES TEST SET PREPARATION SET

As explained earlier, the annotators were asked to specify the shortest span as the answer. However, in some cases, there is a disagreement between the annotators about the shortest span. For an example, consider the paragraph and the corresponding question in Table 3. For the question in Table 3, *"Mohammad-Reza Shajarian"* and *"Shajarian"* are both correct answers. In the case that we just specify "Mohammad-Reza Shajarian" as the correct answer, a QA system that selects "Shajarian" as the answer will be wrongly punished. Hence, to make the evaluation more accurate and reliable, we followed the SQuAD protocol and specified two extra answers for the questions in the test set.

In order to prepare three-candidates test instances, we show each annotator the questions on the test set along with the corresponding paragraphs. Then ask the annotators to specify the shortest answer to the question in the paragraph's text. In this way, we have three answer candidates for each question in the test set. Fig. 5 shows the process of preparing a three-candidates test set.

D. DATA QUALITY MONITORING

In SQuAD's data collection protocol, the annotators were required to have a high HIT acceptance rate to participate in the process. However, no constant supervision is made on the questions that the annotators make. Since SQuAD is a huge dataset with 100,000 question-answer pairs, some wrong or inappropriate instances made by the annotators might be tolerable. PersianQuAD contains 20,000 instances, and we compensate for the lack of quantity by improving our dataset's quality. To this end, we actively check the quality of the instances.

As mentioned earlier, the annotators were asked to perform an annotation task, and if they pose at least 45 questions correctly, they could participate in the data collecting process. In addition to this, to ensure the quality of the dataset, all of the questions were constantly checked by three supervisors.[1] The entire data collection process took approximately two months to complete. At the end of each day, the supervisors sampled a number of questions from each annotator and verified the quality of the questions based on the following criteria: 1) Whether the question is fluent in Persian?, 2) Whether the answer exists in the paragraph text?, 3) Whether the specified answer is correct? And 4) Whether the specified answer is the shortest one? The questions that failed to satisfy these criteria were removed from the dataset. The supervisors also checked the type and variety of the questions that each annotator made and prevented the dataset from being biased toward specific types of questions.

VI. DATASET ANALYSIS

A. DATASET STATISTICS

We split PersianQuAD into train and test sets containing 18,567 and 1000 instances, respectively. The test set was selected randomly from the dataset, with the remaining part of the dataset used for training. Table 4 gives the statistics of training and test sets of PersianQuAD. As mentioned in Section V-C, we specified three candidate answers for the questions in the test set.

B. QUESTION TYPE ANALYSIS

In order to evaluate the proposed model for creating QA datasets and to ensure the representativeness of the resulted data, we analyze the PersianQuAD by extracting the type and number of its questions. To compare the question type distribution of our dataset with that of other datasets, such as SQuAD and TyDi QA, we use English question types as the basis of our analysis. To this end, we mapped each of the question words in English to their corresponding question words

[1] All of the supervisors are native Persian speakers, One of them is a linguist and the others are experts in NLP

TABLE 4. PersianQuAD statistics.

Dataset	Train	Test
No. of Instances	18,567	1000
Avg. Question Length	10.7	10.5
Avg. Answer Length	2.6	2.3
No. of Candidate Answers	1	3

TABLE 5. Question type distribution over PersianQuAD, SQuAD, and TyDi QA datasets.

Question Word	TyDi QA	SQuAD	PersianQuAD
What	30%	51%	28.14%
How	19%	12%	15.24%
When	14%	8%	10.7%
Where	14%	5%	13.6%
Who	9%	11%	16.15%
Which	3%	5%	15.26%
Why	1%	2%	0.92%

TABLE 6. Question type distribution over the training and test parts of PersianQuAD.

Question type	Train	Test
What	28.57%	23.2%
How	15.54%	13.2%
When	11.0%	8.30%
Where	13.21%	21.1%
Who	16.13%	13.0%
Which	14.61%	20.7%
Why	0.94%	0.60%

in Persian. Table 5 shows the question type distribution over PersianQuAD, SQuAD, and TyDi QA datasets. As Table 5 shows, PersianQuAD contains a more balanced distribution of the question words in comparison with SQuAD and even TyDiQA. In all datasets, *"What"* and *"Why"* questions have the highest and the lowest frequency, respectively.

Table 6 shows the statistics of the question types over the training and test sets of the PersianQuAD. As this table shows, the frequency of different question types over the test set is similar to that of the training set, confirms that the test set is a good presented of the whole dataset.

C. QUESTION DIFFICULTY

We take the lexical similarity between the question and the corresponding answer sentence as an indicator of the question's difficulty. The less similarity between a question and its answer sentence, the more difficult it is for a QA system to answer that question. We measure the lexical similarity between the question and the answer sentence by using the Jaccard Coefficient [35].

Assume that the question and the answer sentences are shown as $Q = \{q_1, q_2, \ldots, q_n\}$ and $A = \{a_1, a_2, \ldots, a_m\}$, respectively. Jaccard Coefficient measures the similarity between the question Q and answer sentence A as shown in (1).

$$Jaccard(P, A) = \frac{|P \cap A|}{|P \cup A|} \tag{1}$$

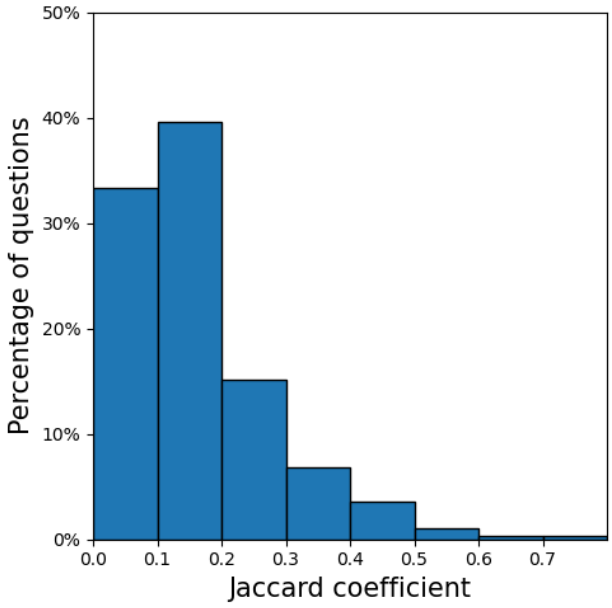

FIGURE 6. Histogram of lexical similarity between question and answers over the PersianQuAD test set.

As (1) shows, Jaccard Coefficient measures the similarity as the ratio of the number of common words between the question and answer sentence to the total number of words in question and answer. Jaccard Coefficient varies between 0 and 1, where 0 shows there is no lexical similarity between the question and the sentence, while 1 shows the sentence contains all the question words. Fig. 6 shows the lexical similarity between the questions and corresponding answer sentences over the PersianQuAD test set. As Fig. 6 shows, for almost 90% of the questions, the similarity between the question and its answer sentence is less than 0.3% in terms of the Jaccard Coefficient. This demonstrates that the lexical overlap between the questions and the answer sentences is very low, and hence, the PersianQuAD questions present an adequate level of complexity.

VII. EXPERIMENTS AND RESULTS

A. METHODS

We design and implement three versions of a deep-learning based QA system and deploy PersianQuAD as the training set of the QA systems. In line with the state-of-the-art research on QA tasks [36], we used three pre-trained language models in our QA systems: MBERT [37], ParsBERT [38] and ALBERT-FA [39].

MBERT (Multilingual Bidirectional Encoder Representations from Transformers) is a deep bidirectional language model developed by Google. It has been pre-trained on Wikipedia articles of 104 languages, including Persian. MBERT has shown great performance on a wide range of NLP tasks such as named entity recognition, question answering, part of speech tagging, etc. ALBERT is a lite version of MBERT with fewer parameters and hence, faster training

speed. It also models the inter-sentence coherence by using a supervised loss. ALBERT shows better performance than MBERT on some NLP tasks with multi-sentence inputs such as English QA [39]. ALBERT-FA is the version of ALBERT trained on the Persian (Farsi) texts. ParsBERT is a monolingual BERT trained on the Persian language.

Fig. 7 shows the general architecture of the proposed QA systems. As Fig. 7 shows, the QA system first tokenizes the paragraph text and the question sentence using the BERT tokenizer. Then it passes the generated tokens to the BERT language model. Finally, the BERT language model predicts the answer's start and end token within the paragraph text, and the answer generation component generates the final answer to the question.

B. ALGORITHM

Algorithm 1 indicates the algorithm of the QA system in Fig. 7. In this algorithm, *Initialize* function first inserts a special token [CLS] at the beginning of the question sentence. Likewise, it adds a special token [SEP] between the question and paragraph and a token [SEP] at the end of the paragraph. Equation (2) shows this function.

$$[CLS]question[SEP]paragraph[SEP] = \text{Initialize}(question, paragraph) \quad (2)$$

The packed sequence is then tokenized using the BERT tokenizer. As shown in (3), q_i shows the i^{th} token of the question sentence and p_j shows the j^{th} token of the paragraph.

$$[I_{[CLS]}, q_1, q_2, \ldots, q_n, I_{[SEP]}, p_1, p_2, \ldots, p_m, I_{[SEP]}] = \text{Tokenize}([CLS]question[SEP]paragraph[SEP]) \quad (3)$$

The BERT model [37] provides contextualized embeddings to embed tokens. We get the embedding vectors of tokens using *Embedding* function. Equation (4) indicates the *Embedding* function. The output of this function is passed to the encoders.

$$[EMD_{I_{[CLS]}}, EMD_{q_1}, \ldots, , EMD_{p_m}, EMD_{I_{[SEP]}}] = \text{Embedding}([I_{[CLS]}, q_1, \ldots, p_m, I_{[SEP]}])\text{inputs} \Leftarrow [EMD_{I_{[CLS]}}, EMD_{q_1}, \ldots, , EMD_{p_m}, EMD_{I_{[SEP]}}] \quad (4)$$

To implement BERT language model, transformer encoders [40] are employed. In transformer encoders, self-attention layers are implemented rather than recurrent neural networks to present each token. In each transformer encoder. The inputs are passed to some self-attention layers. In i^{th} self-attention layer, three vectors Query (Q_i), Key (K_i), and Value (V_i) are generated for each embedding vector emd_j. To generate these vectors in i^{th} self-attention layer, emd_j is multiplied to $W_{Q_i} \in \mathbb{R}^{|emd_j| \times |Q_i|}$, $W_{K_i} \in \mathbb{R}^{|emd_j| \times |K_i|}$, and $W_{V_i} \in \mathbb{R}^{|emd_j| \times |V_i|}$. These matrices are learned during training model. Finally, the vector Z_i is generated as the output of i^{th} self-attention layer. Equations (5) to (8) shows these operations. In (8), σ demonstrates the softmax function.

$$Q_i \Leftarrow emd_j \times W_{Q_i} \quad (5)$$

$$K_i \Leftarrow emd_j \times W_{K_i} \quad (6)$$

$$V_i \Leftarrow emd_j \times W_{V_i} \quad (7)$$

$$Z_i \Leftarrow \sigma\left(\frac{Q_i \times K_i^T}{\sqrt{|K_i|}}\right) \times V_i \quad (8)$$

By concatenating Z_i vectors for all self-attention layers, vector $Z_{1..|Self-Attentions|}$ is generated. This vector is multiplied by the matrix $W_O \in \mathbb{R}^{Z_{1..|Self-Attention|} \times 768}$ and vector Z is produced. W_O is a learnable matrix. Equation (9) shows this multiplication.

$$Z \Leftarrow Z_{1..|Self-Attention|} \times W_O \quad (9)$$

Z vector is then passed to a fully connected network and a new vector emb_j^{new} is generated. In this network, $W_F \in \mathbb{R}^{|768| \times |emb_j|}$ and $b_F \in \mathbb{R}^{|emb_j|}$ are learnable parameters. This fully connected network is shown in (10).

$$emd_j^{new} \Leftarrow Z \times W_F + b_F \quad (10)$$

All of $emd_{1..|inputs|}^{new}$ vectors are passed to a new encoder again. This operation is repeated to the number of encoders.

As shown in (11), the output vectors of the last encoder are passed to a fully connected network and start logits ($s \in \mathbb{R}^{|inputs|}$) and end logits ($e \in \mathbb{R}^{|inputs|}$) are produced. In this equation, $W_{qa} \in \mathbb{R}^{|emd_j| \times 2}$ and $b_{qa} \in \mathbb{R}^{|inputs| \times 2}$. Start logit and end logit indicate the start score and end score of the answer span, respectively.

$$[(s_{I_{[CLS]}}, e_{I_{[CLS]}}), (s_{q_1}, e_{q_1}), \ldots, (s_{p_m}, e_{p_m}), (s_{I_{[SEP]}}, e_{I_{[SEP]}})] \Leftarrow inputs \times W_{qa} + b_{qa} \quad (11)$$

Afterwards, based on (12), the system finds i^{th} and j^{th} tokens of the paragraph so that sum of their logits is maximum. This shows the best span of the paragraph which represents the exact answer.

$$a_1, a_f \Leftarrow \underset{\substack{i,j \in [1,m] \\ i<j}}{\text{argmax}} \; s_{p_i} + e_{p_j} \quad (12)$$

Finally, all tokens $p_{a_1}, p_{a_2}, \ldots, p_{a_f}$ are detokenized and the final answer is generated and returned to the user. Equation (13) shows this process.

$$\text{answer} \Leftarrow \text{Detokenize}(p_{a_1}, p_{a_2}, \ldots, p_{a_f}) \quad (13)$$

C. EVALUATION METRICS

Two evaluation metrics are commonly used for evaluating QA systems: *Exact Match* and *F1* [2]. We use the same metrics in this research.

- *Exact Match*: This metric measures the percentage of the predicted answers that exactly match any of the gold candidate answers.
- *(Macro-averaged) F1*: This metric measures the average overlap between the predicted and the gold candidate answers. To compute the overlap, both the predicted and the candidate answers are represented as bags of words, and hence, the order of words is ignored. The F1 of each question is considered as the maximum F1 over all of

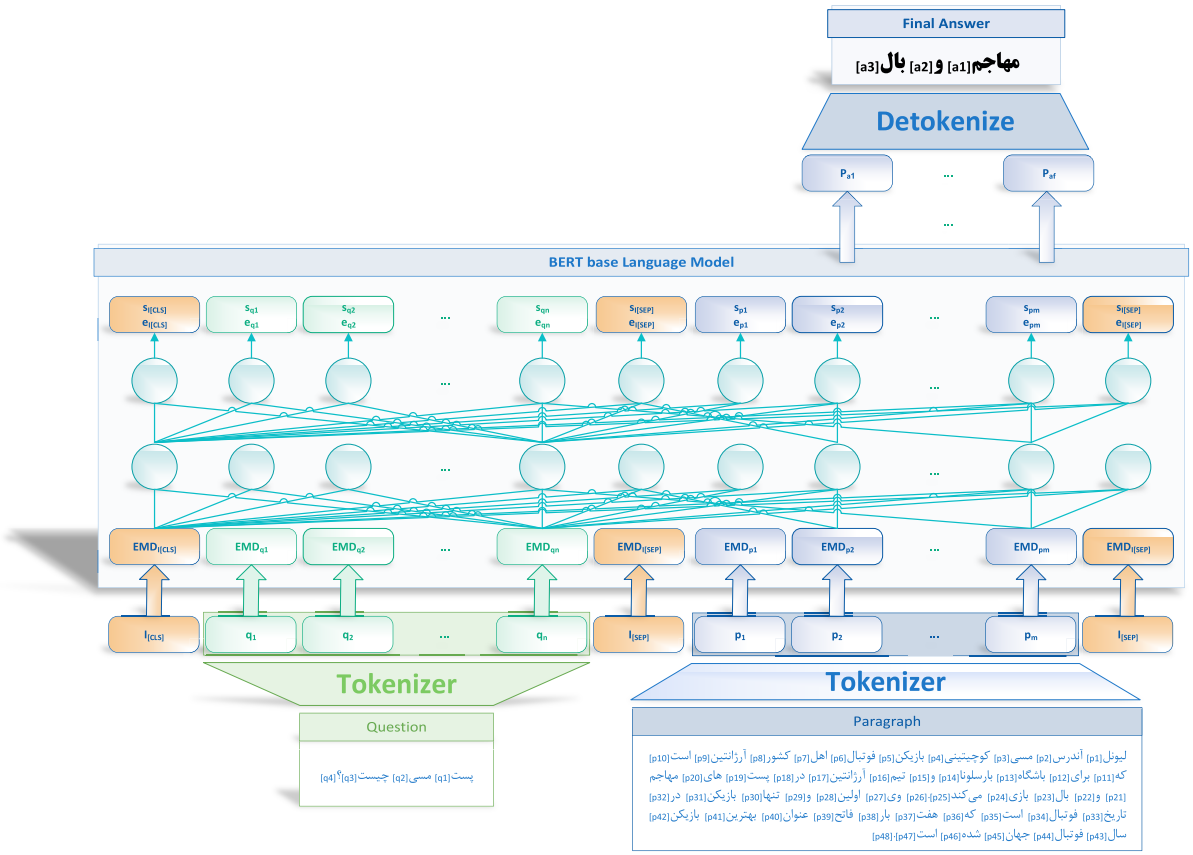

FIGURE 7. The architecture of the proposed QA system.

its candidate answers. The (Macro-averaged) F1 is the average of the F1 scores over all of the questions.

D. HUMAN PERFORMANCE

As explained in Section V-C we prepared a three-candidates test set for PersianQuAD. In order to measure human performance on the PersianQuAD test set, as in SQuAD, we take the third answer to each question as to the human answer and keep the two other answers as gold ones. The human performance on the test set of PersianQuAD is 95.0% and 96.49% in terms of Exact Match and F1 metrics, respectively.

E. SETUP

In order to implement QA systems with MBERT, ParsBERT, and ALBERT-FA, we used Python and PyTorch as our programming language and our deep learning framework, respectively. The models were fine-tuned and tested on Google Colab, with NVIDIA Tesla p100 GPU and 12G RAM. We used the built-in tokenizer in the models (MBERT, ParsBERT, and ALBERT-FA) to tokenize the paragraph and answer text.[2] All of the models were fine-tuned with a batch size of 12, the learning rate of $3e^{-5}$, gradient accumulation steps of 1, and weight decay of 0. We fine-tuned each model for two epochs and used the AdamW optimizer [41]. All of the models were tested with a batch size of 8.

[2]We use *Wordpiece* tokenizer for MBERT and ParsBERT, and *SentencePiece* tokenizer for ALBERT-FA.

F. RESULTS AND ANALYSIS

We build three QA systems according to the pre-trained language models examined (MBERT, ALBERT-FA, ParsBERT). We trained each of the QA systems using the training part of PersianQuAD and evaluate them using the test part. We evaluate each of the QA systems according to two widely used automatic evaluation metrics *Exact Match* and *F1* described in Section VII-C. To have a better understanding of the obtained performance, we compare it with the performance of QA systems trained for other languages. Table 7 shows the performance of the QA systems and human performance on PersianQuAD, along with the performance of the QA systems on the datasets in other languages.

We derive the following observations from the results:

- For PersianQuAD, the best performance in terms of both evaluation metrics is obtained using MBERT. ALBERT-FA, and ParsBERT are in the next ranks, respectively.
- ParsBERT is pre-trained on more massive amounts of Persian data than MBERT. However, this does not have a positive effect on the performance of the Persian QA system and MBERT performs better than ParsBERT.

Algorithm 1 Algorithm of Our Method

Input: question, paragraph
Output: answer

$[CLS]question[SEP]paragraph[SEP] = \text{Initialize}(question, paragraph)$
$[\ I_{[CLS]}, q_1, q_2, \ldots, q_n, I_{[SEP]}, p_1, p_2, \ldots, p_m, I_{[SEP]}\ ] = \text{Tokenize}([CLS]question[SEP]paragraph[SEP])$
$[\ EMD_{I_{[CLS]}}, EMD_{q_1}, \ldots, , EMD_{p_m}, EMD_{I_{[SEP]}}\ ] = \text{Embedding}([\ I_{[CLS]}, q_1, q_2, \ldots, q_n, I_{[SEP]}, p_1, p_2, \ldots, p_m, I_{[SEP]}\ ])$
$inputs \Leftarrow [\ EMD_{I_{[CLS]}}, EMD_{q_1}, \ldots, , EMD_{p_m}, EMD_{I_{[SEP]}}\ ]$
for all $enc \in |encoders|$ **do**
  for all $emd_j \in inputs$ **do**
    for all $i \in |Self - Attentions|$ **do**
      $Q_i \Leftarrow emd_j \times W_{Q_i}$
      $K_i \Leftarrow emd_j \times W_{K_i}$
      $V_i \Leftarrow emd_j \times W_{V_i}$
      $Z_i \Leftarrow \sigma(\frac{Q_i \times K_i^T}{\sqrt{|K_i|}}) \times V_i$
    end for
    $Z \Leftarrow Z_{1..|Self-Attentions|} \times W_O$
    $emd_j^{new} \Leftarrow Z \times W_F + b_F$
  end for
  $inputs \Leftarrow emd_{1..|inputs|}^{new}$
end for
$[(s_{I_{[CLS]}}, e_{I_{[CLS]}}), (s_{q_1}, e_{q_1}), \ldots, (s_{p_m}, e_{p_m}), (s_{I_{[SEP]}}, e_{I_{[SEP]}})] = inputs \times W_{qa} + b_{qa}$
$a_1, a_f \Leftarrow \underset{\substack{i,j \in [1,m] \\ i<j}}{\text{argmax}}\ s_{p_i} + e_{p_j}$
$\text{answer} \Leftarrow \text{Detokenize}(p_{a_1}, p_{a_2}, \ldots, p_{a_f})$
return answer

TABLE 7. Performance of models and humans on PersianQuAD, along with the performance on the QA datasets in other languages.

Dataset	Language	Model	Exact Match	F1
PersianQuAD	Persian	Human	95.0%	96.49%
PersianQuAD	Persian	ALBERT-FA	74.9%	79.25%
PersianQuAD	Persian	ParsBERT	73.8%	79.08%
PersianQuAD	**Persian**	**MBERT**	**78.8%**	**82.97%**
ParSQuAD (manual) [33]	Persian	ALBERT-FA	48.11%	51.66%
ParSQuAD (manual) [33]	Persian	ParsBERT	46.32%	50.06%
ParSQuAD (manual) [33]	Persian	MBERT	52.86%	56.66%
ParSQuAD (automatic) [33]	Persian	ALBERT-FA	64.71%	67.59%
ParSQuAD (automatic) [33]	Persian	ParsBERT	62.42%	65.26%
ParSQuAD (automatic) [33]	Persian	MBERT	67.73%	70.84%
ParsiNLU [34]	Persian	ParsBERT	-	40.70%
ParsiNLU [34]	Persian	MBERT	-	49.70%
SQuAD [2]	English	Bert-base	80.8%	88.5%
SQuAD [2]	English	Bert-large	84.1%	90.9%
SQuAD-es [19]	Spanish	MBERT	48.3%	68.1%
SberQuAD [24]	Russian	BERT	66.6%	84.8%
KorQuAD [26]	Korean	BERT	71.68%	89.76%
PIAF [27]	French	CamemBert	-	79.64%

- While ALBERT performs better than BERT on English SQuAD, this is not the case for the Persian PersianQuAD, and MBERT show higher performance than ALBERT, in terms of both evaluation metrics.
- The performance gap between humans and models on PersianQuAD (17% in Exact Match and 14% in F1) shows that there is still plenty of room for improving the QA models on the PersianQuAD.
- Our best model achieves an $F1$ score of 82.97% and an Exact Match of 78.8%, which are comparable with that of English QA systems trained on the SQuAD. It shows that the PersianQuAD performs well in training the deep-learning-based QA systems.
- The performance of the QA systems trained on PersianQuAD is better than that of QA systems trained on ParSQuAD and ParsiNLU, indicating that PersianQuAD works well on training QA systems for the Persian language.
- The performance of the QA systems trained on PersianQuAD is better than that of QA systems trained on SQuAD-es (Spanish), SberQuAD (Russian), KorQuAD (Korean) and PIAF(French), (Except for the

TABLE 8. F1 score of the models stratified by question types.

Question type	Freq. in PersianQuAD	ParsBERT	ALBERT-FA	MBERT
What	28.57%	62.07%	64.22%	71.12%
How	15.54%	84.09%	81.06%	88.64%
When	11.0%	85.54%	78.31%	92.77%
Where	13.21%	71.09%	74.41%	77.25%
Who	16.13%	74.62%	73.85%	79.23%
Which	14.61%	72.95%	74.88%	77.78%
Why	0.94%	100%	100%	100%

SberQuAD and KorQuAD, with a slight decrease in terms of Exact Match).

Table 8 shows the F1 score of the models on answering different types of questions in PersianQuAD.

Here we observe:

- All models have their best performance on *Why*, *How* and *When* question types. We hypothesize that this can be attributed to the fact that the answer of "Why" questions in PersianQuAD usually appears after a sentence starting with "because", and hence, finding the answer is straightforward. *How* question types include *How much* and *How many*. The answer to *How much*, *How many* and *When* questions are usually quantities and finding quantifiers as answers is relatively straightforward for the models.
- All models show their worse performance on *What* question types. This is because *What* questions in English are mapped into several types of questions in Persian, and finding the answers to them is relatively complicated.
- The ranking of question types, based on the model performance, for MBERT and ParsBERT is the same (1-*What*, 2-*Where*, 3-*Which*, 4-*Who*, 5-*How*, 6-*When*, 7-*Why*), while for ALBERT-FA the ranking is slightly different (1-*What*, 2-*Who*, 3-*Where*, 4-*Which*, 5-*When*, 6-*How*, 7-*Why*). We hypothesize that this is because MBERT and ParsBERT use the same architecture, while ALBERT-FA uses a different one.

VIII. CONCLUSION

This paper presents a model for developing Persian datasets for deep-learning-based QA systems. The proposed model consists of four steps: 1) Wikipedia article selection, 2) question-answer collection, 3) three-candidates test set preparation, and 4) Data Quality Monitoring. We deploy the proposed model to create PersianQuAD, the first native question answering dataset for the Persian language. PersianQuAD consists of approximately 20,000 (question, paragraph, answer) triplets on Persian Wikipedia articles and is created by native annotators. We analysed PersianQuAD and showed that it contains questions of varying types and difficulties and hence, it is a good presenter of real-world questions in the Persian language. We built three QA systems using MBERT, ALBERT-FA and ParsBERT. The best system uses MBERT and achieves a F1 score of 82.97% and an Exact Match of 78.8%. The results show that the resulted dataset performs well for training deep-learning-based QA systems. We have made our dataset and QA models freely available and hope that it encourages the development of new QA datasets and systems for different languages, and leads to further advances in machine comprehension.

ACKNOWLEDGMENT

The authors thank all the members of the Big Data Research Group, University of Isfahan. They also thank the students in linguistics at the University of Isfahan for helping them to prepare the PersianQuAD.

REFERENCES

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ, USA: Pearson, 2009.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proc. EMNLP*, Austin, TX, USA, Nov. 2016, pp. 2383–2392.

[3] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki, "TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 454–470, Dec. 2020.

[4] G. Lembersky, N. Ordan, and S. Wintner, "Language models for machine translation: Original vs. translated texts," *Comput. Linguistics*, vol. 38, no. 4, pp. 799–825, Dec. 2012.

[5] V. Volansky, N. Ordan, and S. Wintner, "On the features of translationese," *Digit. Scholarship Humanities*, vol. 30, no. 1, pp. 98–118, Apr. 2015.

[6] S. Wintner, "Translationese: Between human and machine translation," in *Proc. COLING*, 2016, pp. 18–19.

[7] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2018, pp. 784–789.

[8] Y. Yang, W.-T. Yih, and C. Meek, "WikiQA: A challenge dataset for open-domain question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 2013–2018.

[9] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," in *Proc. CoCo@ NIPS*, 2016, pp. 1–10.

[10] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, and K. Toutanova, "Natural questions: A benchmark for question answering research," *Trans. Assoc. Comput. Linguistics*, vol. 7, no. 29, pp. 453–466, 2019.

[11] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-T. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, "QuAC: Question answering in context," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 2174–2184.

[12] S. Reddy, D. Chen, and C. Manning, "CoQA: A conversational question answering challenge," *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 249–266, May 2019.

[13] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman, "NewsQA: A machine comprehension dataset," in *Proc. 2nd Workshop Represent. Learn. NLP*, Vancouver, BC, Canada, Aug. 2017, pp. 191–200.

[14] L. Siciliani, P. Basile, P. Lops, and G. Semeraro, "MQALD: Evaluating the impact of modifiers in question answering over knowledge graphs," *Semantic Web*, vol. 13, no. 2, pp. 215–231, Feb. 2022.

[15] J. Welbl, P. Stenetorp, and S. Riedel, "Constructing datasets for multi-hop reading comprehension across documents," *Trans. Assoc. Comput. Linguistics*, vol. 6, pp. 287–302, Dec. 2018.
[16] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Brussels, Belgium, 2018, pp. 2369–2380.
[17] A. Talmor and J. Berant, "Repartitioning of the ComplexWebQuestions dataset," 2018, *arXiv:1807.09623*.
[18] N. Inoue, P. Stenetorp, and K. Inui, "R4C: A benchmark for evaluating RC systems to get the right answer for the right reason," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 6740–6750.
[19] C. P. Carrino, R. M. Costa-jussà, and A. R. J. Fonollosa, "Automatic Spanish translation of SQuAD dataset for multi-lingual question answering," in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, May 2020, pp. 5515–5523.
[20] H. Mozannar, E. Maamary, K. El Hajal, and H. Hajj, "Neural Arabic question answering," in *Proc. 4th Arabic Natural Lang. Process. Workshop*, 2019, pp. 108–118.
[21] K. Lee, K. Yoon, S. Park, and S.-W. Hwang, "Semi-supervised training data generation for multilingual question answering," in *Proc. 11th Int. Conf. Lang. Resour. Eval. (LREC)*, 2018, pp. 1–5.
[22] D. Croce, A. Zelenanska, and R. Basili, "Neural learning for question answering in Italian," in *Proc. Int. Conf. Italian Assoc. Artif. Intell.* Cham, Switzerland: Springer, 2018, pp. 389–402.
[23] S. Eetemadi and K. Toutanova, "Asymmetric features of human generated translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 159–164.
[24] P. Efimov, A. Chertok, L. Boytsov, and P. Braslavski, "SberQuAD–Russian reading comprehension dataset: Description and analysis," in *Proc. Int. Conf. Cross-Lang. Eval. Forum Eur. Lang.* Cham, Switzerland: Springer, 2020, pp. 3–15.
[25] C. Chieh Shao, T. Liu, Y. Lai, Y. Tseng, and S. Tsai, "DRCD: A Chinese machine reading comprehension dataset," 2018, *arXiv:1806.00920*.
[26] S. Lim, M. Kim, and J. Lee, "KorQuAD1.0: Korean QA dataset for machine reading comprehension," 2019, *arXiv:1909.07005*.
[27] R. Keraron, G. Lancrenon, M. Bras, F. Allary, G. Moyse, T. Scialom, E.-P. Soriano-Morales, and J. Staiano, "Project PIAF: Building a native French question-answering dataset," in *Proc. 12th Lang. Resour. Eval. Conf.*, Marseille, France, May 2020, pp. 5481–5490.
[28] M. Artetxe, S. Ruder, and D. Yogatama, "On the cross-lingual transferability of monolingual representations," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 4623–4637.
[29] P. Lewis, B. Oguz, R. Rinott, S. Riedel, and H. Schwenk, "MLQA: Evaluating cross-lingual extractive question answering," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7315–7330.
[30] D. Gupta, S. Kumari, A. Ekbal, and P. Bhattacharyya, "MMQA: A multi-domain multi-lingual question-answering framework for English and Hindi," in *Proc. 11th Int. Conf. Lang. Resour. Eval. (LREC)*, 2018, pp. 1–8.
[31] Y. Jing, D. Xiong, and Z. Yan, "BiPaR: A bilingual parallel dataset for multilingual and cross-lingual reading comprehension on novels," in *Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, 2019, pp. 2452–2462.
[32] J. Liu, Y. Lin, Z. Liu, and M. Sun, "XQA: A cross-lingual open-domain question answering dataset," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 2358–2368.
[33] N. Abadani, J. Mozafari, A. Fatemi, M. Nematbakhsh, and A. Kazemi, "ParSQuAD: Persian question answering dataset based on machine translation of SQuAD 2.0," *Int. J. Web Res.*, vol. 4, no. 1, pp. 34–46, 2021.
[34] D. Khashabi, A. Cohan, S. Shakeri, P. Hosseini, P. Pezeshkpour, M. Alikhani, M. Aminnaseri, M. Bitaab, F. Brahman, and S. Ghazarian, "ParsiNLU: A suite of language understanding challenges for Persian," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 1163–1178, 2021.
[35] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.
[36] A. Andrenucci and E. Sneiders, "Automated question answering: Review of the main approaches," in *Proc. 3rd Int. Conf. Inf. Technol. Appl. (ICITA)*, 2005, pp. 514–519.
[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Jun. 2019, pp. 4171–4186.
[38] M. Farahani, M. Gharachorloo, M. Farahani, and M. Manthouri, "ParsBERT: Transformer-based model for Persian language understanding," *Neural Process. Lett.*, vol. 53, no. 6, pp. 3831–3847, Dec. 2021.
[39] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.
[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances Neural Inf. Process. Syst.*, vol. 30, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017.
[41] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

AREFEH KAZEMI received the B.Sc. degree in software engineering, the M.Sc. degree in artificial intelligence, and the Ph.D. degree in artificial intelligence from the University of Isfahan, Isfahan, Iran, in 2008, 2010, and 2017, respectively. She is currently an Assistant Professor with the Computational Linguistics Group, University of Isfahan. Her research interests include natural language processing, computational linguistics, and data mining.

JAMSHID MOZAFARI received the B.Sc. degree in computer engineering from the University of Kurdistan, in 2016, and the M.Sc. degree in computer engineering from the University of Isfahan, in 2019. He is currently a Research Assistant in natural language processing and information retrieval and a member of the BIGDATA Laboratory, University of Isfahan. His research interests include natural language processing, information retrieval, question answering, and machine reading comprehension.

MOHAMMAD ALI NEMATBAKHSH received the B.Sc. degree in electrical engineering from Louisiana Tech University, USA, in 1981, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from The University of Arizona, USA, in 1983 and 1987, respectively. He is currently a Full Professor with the Software Engineering Department, University of Isfahan. He worked with Micro Advanced Company and Toshiba Corporation for many years before joining the University of Isfahan. He has published more than 150 peer-reviewed research articles, several U.S. registered patents, and two database books that are widely used in universities. His main research interests include intelligent web and bigdata technology.