ParsiNorm: A Persian Toolkit for Speech Processing Normalization

Romina Oji*, Seyedeh Fatemeh Razavi†‡, Sajjad Abdi Dehsorkh†‡, Alireza Hariri†‡, Hadi Asheri†‡, and Reshad Hosseini†‡
{romina.oji, razavi_f, sadjad.abdi, alireza.hariri, hadi.asheri, reshad.hosseini}@ut.ac.ir
* School of ECE, College of Engineering, University of Tehran, Tehran, Iran
† Hoosh Afzare Rahbare Aryaman (HARA) Company, Tehran, Iran
‡ Equal contribution

Abstract—In general, speech processing models consist of a language model along with an acoustic model. Regardless of the language model’s complexity and variants, three critical pre-processing steps are needed in language models: cleaning, normalization, and tokenization. Among mentioned steps, the normalization step is so essential to format unification in pure textual applications. However, for embedded language models in speech processing modules, normalization is not limited to format unification. Moreover, it has to convert each readable symbol, number, etc., to how they are pronounced. To the best of our knowledge, there is no Persian normalization toolkits for embedded language models in speech processing modules. So in this paper, we propose an open-source normalization toolkit for text processing in speech applications. Briefly, we consider different readable Persian text like symbols (common currencies, #, @, URL, etc.), numbers (date, time, phone number, national code, etc.), and so on. Comparison with other available Persian textual normalization tools indicates the superiority of the proposed method in speech processing. Also, comparing the model’s performance for one of the proposed functions (sentence separation) with other common natural language libraries such as HAZM and Parsivar indicates the proper performance of the proposed method. Besides, its evaluation of some Persian Wikipedia data confirms the proper performance of the proposed method.

Index Terms—pre-processing, normalization, speech processing

I. INTRODUCTION

Pre-processing is one of the essential steps in the applications of Natural Language Processing (NLP) and Speech Processing (SP) [1]. Normalization is the main component of pre-processing that leads to having a standard format for the text. In this regard, some popular libraries like Spacy[1] and Nltk[2] have been presented. However, the accuracy of these methods for Persian language is relatively low due to limited resources, while this problem is critical in the Persian language. Because, some characteristic of the Persian language leads to appear some challenges in pre-processing. Although some may not be the case for standard texts like news, it will be seen a lot for crawled texts containing conversational and colloquial sentences. Since, the grammar of formal sentences is somewhat different. For example, the following two sentences have the same meaning but do not have the same spelling rule: "آیا یه کتابخانه برود می؟" and "آیا یه کتابخانه برود می؟". Furthermore, some words are written in several ways like "تستگاه," "تستگاه," "تستگاه," "تستگاه," "تستگاه," and "تستگاه." The presence of symbols from other languages such as Arabic and English, emojis, punctuations, etc., make more challenges for the normalization step. In addition to the mentioned points, the normalization step will be more challenging in speech modules when we integrate a language model in SP. Since a language model is suitable, that is close to the human reading of sentences. For example, reading a phone number varies according to separating the number of digits in different people. For example, "1234" is pronounced in many forms like "هزار و دویست و سی و چهار", "دویسته سی و چهار", etc. Therefore, besides mentioned normalization, some other operators need to be added for SP normalization purposes.

All in all, there are not many dedicated libraries for the Persian language and it suffers from a lack of coherent tools. The most famous of them are HAZM[3] and Parsivar [2]. On the other hand, the focus of the available tools has been solely on textual applications. Therefore, there is no tool in this language that is efficient in the combined applications of text and audio. In other words, the pre-processing method should cover the pre-processing steps of SP in addition to NLP. Best of our knowledge, It has not been developed any tools with this consideration. Therefore, we develop a dual-purpose tool that includes both NLP and SP normalization[4]. Our main contributions are as follows:

• Improving existing general normalization (NLP normalization)
• Developing a new algorithm for sentence separation.
• Developing a new SP normalization presented for the first time in the Persian language
• Randomly generating different spelling for non-standard words like symbols, time, etc.

The paper is organized as follows: Sec[II] provides a brief overview of previous methods. Sec[III] describes details of the proposed method. Experiments and comparisons with

1https://spacy.io/
2https://www.nltk.org/
3https://www.sobhe.ir/hazm/
4Our open-source implementation is available at https://github.com/haraai
other available pre-processing toolkits are presented in Sec. IV. Finally, the paper is concluded in Sec. V.

II. RELATED WORKS

Text pre-processing is the primary step in NLP and SP applications. Three essential components in this stage consist of cleaning, normalization, and tokenization. Developing an SP normalization is the primary goal of this paper. Therefore, there is more attention on normalization and related research. In the Persian language, the main focus of researchers has been on text normalization. Therefore, no suitable general tool for text normalization in embedded audio models has been provided, while there are different Persian textual tools. Therefore, briefly, an overview of textual normalizations for the Persian language is presented.

One of the open-source libraries offered for NLP normalization is called HAZM. Normalization and pos-tagging are some of its possibilities. Empirical evidence from the use of this library suggests that despite the library’s capability, the results are not necessarily accurate in some situations. Among the available tools for the Persian language, only Parsivar seems to be a significant competitor for our work. However, its focus has also been on pre-processing the text and unifying the text. The most important things covered in its normalization are converting numbers to their Persian text form, removing extra spaces, and removing punctuation marks related to other languages such as Arabic. Although Parsivar also has instructions for normalization for dates, the important thing is that the way it is processed is not the same as how humans read it. Stemming and lemmatization are other modules in parsivar that make it suitable for textual processing. However, they are not more needed in the SP normalization anymore. Sentence segmentation is another ability of Parsivar that acts based on the end-of-sentence punctuations. While it is not always true. Because there are many exceptions in the Persian sentences like “...” or existing abbreviations separated by “.”. Another critical point is the conversational sentences that are not necessarily distinguishable from the punctuation marks, and the grammar of those sentences is different from the formal sentences.

Besides, the focus of all tools has been limited to textual applications. Meanwhile, due to the importance of SP normalization, we introduce ParsiNorm that performs normalization related to SP modules.

III. PROPOSED METHOD

ParsiNorm is a tool written in python, and it is suitable for Persian text normalization for text and speech applications. Before every SP system, a text normalization pre-processing is needed for handling non-standard words such as numbers, dates, time, etc. Each non-standard word must be written how it is pronounced. For example, room number 101 is pronounced (one oh one), or the year 2021 is pronounced twenty twenty-one, so different numbers have different reading styles based on their types, both in Persian and English. We have developed two different normalizations. Sec. A explains one of them, which can be used for every Persian application using text. Sec. B explains the other normalization which is used as pre-processing for SP systems. We have investigated Persian language features and developed functions that converts words with different reading pronunciation and writing style to their reading forms.

A. General Normalization

Near all raw texts need some normalization. In this section, we explain different general implemented normalizations that can be used for Persian texts regardless of what applications these texts are going to be used.

• **Same character encoding:** The Persian language has the same character shapes as Arabic, but there are some differences. For example, the character "۰" has different written forms in Arabic, like "٠", "٠". We convert all of the Persian alphabets to the same format. It is common to write some English words in Persian writings like some special names, but the problem is that people do not use the same Character format, so we convert all English characters to the same format. For example, the letter "i" can be written in different formats like "ی", "ی", etc. we convert all 26 English characters (Capital and small letters) to the same format.

• **Same Persian numbers:** There are two different formats for Persian numbers. We replace all of them with a unique form. Also, different English number formats are used in Persian texts, so we find and replace them with the same Persian forms. For example, number six in Persian has two different writing forms. Also, number six in English has various formats like "6", "6", "6", "6", "6", "0", "0", "0", etc. we find them in sentences and replace them with Persian number "۶".

• **Convert English and Arabic symbols to text:** Both in Persian and Arabic, some symbols are used instead of writing the whole word. For instance, some authors prefer to write symbol "/" instead of "ٌ" because it is more convenient. We find all of these symbols and convert them to whole words.

• **Normalize punctuations:** Some punctuations have different formats. We find all the formats and convert them to the same format. For example, "%" can be written in different Unicode such as "٪" and "%".

• **Convert Math symbols to Persian numbers:** Some math symbols such as "½" are used in the Persian corpus. But the correct format is "½" which is a complete writing form with Persian numbers.

• **Replace HTML tags with text:** Because some Persian data sources are crowded web texts, there are some HTML tags in these texts. These tags are replaced with the correct and readable form. For example, less than HTML tag "&lt" is replaced by "<".

• **Remove emojis:** In some tasks, emojis are not important, and it is better to be cleaned. We have gathered a large list of emojis and remove them from sentences.
• **Separate sentences:** Separating sentences at correct positions is an important task that can improve some text models. For instance, one of the tasks in the Bert language model is next sentence prediction, so it is important to separate each sentence at its correct position. In Persian, all sentences end with a dot, like English sentences. We can split sentences with dot marks, but there is a big problem: dots can be used in other parts of the sentence, such as abbreviations, floating-point numbers, emails, URLs, etc. We have gathered an extensive list of abbreviations from different categories and replace them with how they are read. Also, emails and URLs are converted to how they are pronounced. More information about abbreviations, emails are described in part SP Normalization. Floating-point numbers with dot splitters have a different mechanism that separates them from other dots in the sentence. Parsivar just considers dot floating points, but for the rest words with punctuation, it splits sentences wrongly. We have improved Parsivar sentence separation and add the aforementioned features. This sentence separation considers that all sentences have a dot at the end. But some people forget to put a dot at the end of each sentence. We use other Persian language features to separate sentences. In Persian formal sentences, verbs come at the end of the sentence. Using this feature, we can separate sentences by Verbs. To do this, we use the HAZM POS tagger to find the verb in a sentence, and then we separate each paragraph into some sentences.

**B. Speech Processing Normalization**

Following normalization is used for SP applications. Some symbols, numbers, and texts are read differently from how they are written, so the spelled form must be replaced in speech applications.

• **Replace all symbols with text:** Some symbols are used in both Persian and English texts. We have gathered a list of these symbols and replace them with how it is read. Some of these symbols are: `start(*)`, `square(#)`, `degree(°)`, `percent(%)`, and etc.

• **Replace common currency with their readable text format:** Currency symbols are used instead of words. When everyone reads these symbols, tell the correct word of it. For example, the symbol `$` is pronounced dollar in English. We have gathered currency signs used in different countries and replaced their symbols to how they are read.

• **Replace Math symbols with Persian readable text:** As mentioned in the general normalization section, some math symbols are used in the Persian corpus. In general normalization, we replace them with Persian Numbers but, they have different reading formats. All of these symbols are converted to how they are read.

• **Replace abbreviations with how they are pronounced:** Both English and Persian abbreviations are used in the Persian corpus. Persian Abbrevians are read differently from how they are written, but English abbreviations are somehow read in the same format as they are written. When English abbreviations are read, we pronounce each letter separately. For example, the abbreviation Ph.D. is read "pi ef ri di" ("پی اف ری دی"). We Find all English abbreviations (can end with a dot or not) and replace each English letter with how they are pronounced in Persian. Persian native speakers read abbreviations how they are written entirely Because all Persian abbreviations stand for the first letter of a long-phrase that contains multiple words. For example, "روژ" is the abbreviation of the phrase "روژنامه" which is used the first letter of this phrase "رو" and "روژ". We have gathered an extensive list that contains about 200 abbreviations and their complete format. They are categorized into different categories such as law, books, date, time, and others. All abbreviations are replaced with their long-phrase form.

• **Replace URL and Emails to the text how it is pronounced:** Emails and URLs are pronounced differently from how they are written because some symbols such as "@", "-", ",", etc. are a part of them. We replace these symbols with the words how they are pronounced. For instance, "@" is replaced by "at". Furthermore, we separate each part of emails and URLs and put a space between them because this is how they are read. Some URLs are very big when non-English characters are used and converted to other characters such as "%", digits, and English alphabet. For these long URLs, we keep the main part of it and remove the sub-directory. For example "http://wpc.be1.e.edgecastcdn.net/news/20ak9y4prra.html" is converted to "http do noghte slash slash wpc dot be1e dot edgecastcdn dot net" by removing its sub-directories.

• **Convert date to text:** Dates can be written in two formats: text format like tenth January, twenty eighteen or numbers of the day, month, year, which are split by punctuations like ",", ",", ",", and ",," for example, 10/1/2018. We find second forms and replace them with the first mentioned form. In Persian Corpus, three types of dates are used which are solar hijri, gregorian calendar, lunar hijri. They can be detected based on their years. We use some roles to detect the date type and replace them with how they are read. Detecting the type of date is important because the month names of these types are different from each other. For instance, the first month of solar hijri, gregorian calendar, lunar hijri are in order: Farvardin, January, Moharam. We have made ten different templates that date can be read, and for each date, we select one of them randomly and replace the blanks with the appropriate form of day, month, and year. One example of converting solar hijri calendar date to text is converting

3https://www.sobhe.ir/hazm/
TABLE I: Comparison among available text-specific Persian toolboxes with ParsiNorm.

| Semiotic class                                      | ours | Parsivar | HAZM | person-tool |
|-----------------------------------------------------|------|----------|------|-------------|
| English Abbreviation                                | ✓    | x        | x    | x           |
| Abbreviation of different topics                    | ✓    | x        | x    | x           |
| acronyms read as letters                            | ✓    | x        | x    | x           |
| times dates                                         | ✓    | ✓        | ✓    | x           |
| Same character encoding                             | ✓    | ✓        | ✓    | x           |
| Same Persian numbers                                | ✓    | ✓        | ✓    | x           |
| (English/Arabic) symbols to text                    | ✓    | x        | x    | x           |
| Math symbols to Persian numbers                     | ✓    | x        | x    | x           |
| Replace all symbols with text                       | ✓    | x        | x    | x           |
| Replace currencies with text                        | ✓    | x        | x    | x           |
| Numbers to text                                     | ✓    | ✓        | x    | x           |
| Detect telephone numbers                            | ✓    |✓       | x    | x           |
| Convert national-code to text                       | ✓    | x        | x    | x           |
| Convert card-number to text                         | ✓    | x        | x    | x           |
| Convert Sheba to text                               | ✓    | x        | x    | x           |
| Replace HTML tags with text                         | ✓    |          | x    | x           |
| URL and Emails to text                              | ✓    | x        | x    | x           |
| Normalize punctuations                              | ✓    | x        | x    | x           |
| Remove emojis                                       | ✓    | x        | x    | x           |
| Separate sentences                                  | ✓    |          | x    | x           |
| Pinglish to Persians                                | ✓    | x        | x    | x           |
| Informal to formal                                  | ✓    | x        | x    | x           |
| Replace spaces by half-space                        | ✓    |          | x    | ✓           |

1. Harm converts Only two letters, "ای" and "به" from Arabic to Persian form.
2. Only Arabic numbers are converted to Persian numbers, and the various form of numbers which can be the symbol of numbers such as ی are not converted to Persian numbers.
3. The persian-tools just detect the type of numbers such as Telephone, Sheba, card number and national code. These numbers do not turn into the equivalent text of its readings.
4. Sentence separating in Hazm and Parsivar is based on punctuations. This is not a good way of separating sentences because these punctuations can be used in some words that are not at the end of the sentence. So this way of separating sentences has lots of problems. Parsivar detects floating points and does not consider these dots as dots that come at the end of the sentences, But other types of dots do not indicate the end of sentences, such as dots used in abbreviations.

Generally, non-standard words such as numbers, monetary amounts, dates, and other concepts spelled differently than verbalized, need special pre-processing in SP systems. Based on Persian language features, we have converted various non-standard words to how they are pronounced. As far as we are concerned, no publicly available tool for Persian SP normalization is available. We have compared ParsiNorm implemented functions with other normalization tools such as Hazm, Parsivar, persian-tool. The result of this comparison is available in Tab.1.

The results show the superiority of ParsiNorm compared to other available methods for SP applications.

On the other hand, different numbers have different reading formats. For example, long numbers and special numbers (such as phone numbers, national codes, banking card numbers, etc.) are read differently from written ones. Also, times and dates are read differently. For instance, the time "10:30:25", we do not express ""," in daily speaking but define changes from hour to minutes by using special characters such as "و" or telling that what minute and second are ". Therefore, we consider
TABLE II: Some examples of the performance of the proposed method in the face of numerical samples with considering different pronunciations.

| Examples | Semiotic class: times | input: 11:35 |
|----------|-----------------------|-------------|
| بیست و پنجم ماه هزار و چهارصد |
| بیست و پنجم مهر هزار و چهارصد |
| بیست و پنجم سال هزار و چهارصد |
| بیست و پنجم هفته هزار و چهارصد |

| Examples | Semiotic class: dates | input: 1400-07-25 |
|----------|-----------------------|------------------|
| بیست و پنجم ماه هزار و چهارصد |
| بیست و پنجم مهر هزار و چهارصد |
| بیست و پنجم سال هزار و چهارصد |
| بیست و پنجم هفته هزار و چهارصد |

| Examples | Semiotic class: detect telephone numbers and convert | input: 09397796915 |
|----------|-------------------------------------------------------|-------------------|
| صفر نیصد و سی و هشت هفده و هفت نود و شش نیصد و پانزده |
| صفر نیصد و سی و هشت هفده و هفت نیصد و شصت و ون نیصد و پانزده |
| صفر نیصد و سی و هشت هفده و هفت شصت و ون |
| صفر نیصد و سی و هشت هفده و هفت نیصد |

| Examples | Semiotic class: detect persian national code and convert to text | input: 0523924984 |
|----------|---------------------------------------------------------------|------------------|
| صفر نیصد و سی و هشت هفده و هفت نود و دو جیل و هشت نیصد و چهار |
| صفر نیصد و سی و هشت هفده و هفت نیصد و دو جیل و هشت نیصد و چهار |

| Examples | Semiotic class: detect card number and convert to text | input: 6104337852441441 |
|----------|--------------------------------------------------------|-------------------------|
| شصت و نیک صفر چهار سی و هشت هفده و شصت پنجاه و دو جیل و چهار چهارصد |
| شصت و نیک صفر چهار سی و هشت هفده و شصت پنجاه و دو جیل و چهار چهارصد |
| شصت و نیک صفر چهار سی و هشت هفده و شصت پنجاه و دو جیل و چهار چهارصد |

TABLE III: The accuracy of ParsiNorm, HAZM, and Parsivar for 107 random sentences from Wikipedia for sentence separation.

| Criterion | ParsiNorm | HAZM | Parsivar |
|-----------|-----------|------|----------|
| Accuracy  | 89.71     | 88.78| 69.15    |

all ways a number (like date, time, etc.) can be read. Finally, one of them is selected randomly and is replaced with the written format. We have shown different ways of telling each number in Tab. II. As seen, we do not necessarily have a specific output, and each particular number can have a different result every time.

Furthermore, the performance of one of the functions is compared with other methods. Breaking long sentences into shorter ones is usually done by identifying the ‘.’ in the sentence by HAZM or Parsivar. HAZM, however, also takes into account several modes of abbreviation. But the proposed method is not limited to the point but also considers the maximum possible scenarios. Because in Persian, ‘.’ does not necessarily mean the end of the sentence and is used in other cases such as decimal, abbreviation, ‘.’ in the URLs, etc. Accuracy for the three methods on 107 random sentences from Wikipedia is presented in Tab. III.

To further intuition, the performance of the proposed method on formal some extracted sentences from Persian Wikipedia is depicted in Fig. 1.

V. CONCLUSION

This paper proposed ParsiNorm toolkit, an open-source normalization tool for the Persian language in both text and speech processing applications. Randomly generating a different reading form for non-standard words like time, symbol, number, etc., is the main contribution in this paper. Furthermore, proposing a robust Persian sentence recognition method is another one. Comparison with available basic libraries, along with the presented results about the method’s performance on some of the sentences, indicates the excellent performance of the proposed method.

REFERENCES

[1] D. Jurafski and J. H. Martin, Speech and Language Processing, 3rd ed., 2021.

[2] S. Mohtaj, B. Roshanfekr, A. Zafarian, and H. Asghari. “Parsivar: A language processing toolkit for persian.” In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.