Parsivar: A Language Processing Toolkit for Persian

Salar Mohtaj[1], **Behnam Roshanfekr**[2], **Atefeh Zafarian**[3], **Habibollah Asghari**[4]
[1,2,3,4] ICT Research Institute, Academic Center for Education Culture and Research, Tehran Iran
{salar.mohtaj, b.roshanfekr, atefeh.zafarian, habib.asghari}@ictrc.ac.ir

Abstract

With the growth of Internet usage, a massive amount of textual data is generated on social media and the Web. As the text on the Web are generated by different authors with various types of writing styles and different encodings, a preprocessing step is required before applying any NLP task. The goal of preprocessing is to convert text into a standard format that makes it easy to extract information from documents and sentences. Moreover, the problem is more acute when we deal with Arabic script-based languages, in which there are some different kinds of encoding schemes, different kinds of writing styles and the spaces between or within the words.

This paper introduces a preprocessing toolkit named as Parsivar, which is a comprehensive set of tools for Persian text preprocessing tasks. This toolkit performs various kinds of activities comprised of normalization, space correction, tokenization, stemming, parts of speech tagging and shallow parsing. To evaluate the performance of the proposed toolkit, both intrinsic and extrinsic approaches for evaluation have been applied. A Persian plagiarism detection system has been exploited as a downstream task for extrinsic evaluation of the proposed toolkit. The results have revealed that our toolkit outperforms the available Persian preprocessing toolkits by about 8 percent in terms of F1.

Keywords: Preprocessing Tasks, Natural Language Processing, Language Processing Toolkit

1. Introduction

Preprocessing is one of the essential steps in Natural Language Processing (NLP) tasks that convert unstructured texts into a standard text format suitable for NLP applications. This makes it easy to extract knowledge from documents and sentences. Despite of its importance, few efforts have been accomplished on developing preprocessing tools in low-resource languages. Development of preprocessing tools is more challenging in Arabic script-based languages like Persian, in which there are some difficulties including the lack of enough linguistic resources, various character encodings, spaces between or within multi-token words and a wide variety of suffixes.

In this paper, we introduce Parsivar, a Persian preprocessing toolkit that includes a set of tools necessary for different NLP tasks. It normalizes Persian texts to a standard format, corrects the spacing between or within the words, tokenizes words and sentences, extracts word stems, tags the words with their Part of Speech (PoS) and finally performs shallow parsing on sentences. For space correction, it also proposes two different kinds of solutions. The first solution is based on some pre-defined rules and the second one is based on learning methods. The evaluation results have revealed that the proposed toolkit outperforms the other existing preprocessing toolkits in Persian.

The rest of the paper is organized as follows: Section 2 discusses the previous works in the field of Persian text preprocessing. In Section 3 we describe the algorithms for implementing each of the modules in Parsivar toolkit in detail. In section 4, we present the experimental results for evaluating and comparing existing Persian preprocessing toolkits with respect to our approach. For the purpose of benchmarking the algorithms, we evaluate them on the performance of a Persian plagiarism detection application using extrinsic evaluation. Finally, in the last section, conclusion and recommendations for future works will be described.

2. Related work

Due to the importance of preprocessing in NLP applications, some attempts have been done to develop integrated Persian preprocessing packages in recent years. In 2010, Shamsfard et al. (Shamsfard, Sadat Jafari and Ilbeygi, 2010) proposed STeP1, which includes a combination of tools such as tokenization, morphological analysis and a POS tagging. The ParsiPardaz toolkit was proposed by Sarabi et al. (Sarabi, Mahyar and Farhoodi, 2013) which provides the STeP1 capabilities along with some other tasks like normalization and spell checking. One of the issues related to the mentioned toolkits is that they are not publicly available as open source applications. Sobhe (Hazm, 2014) introduced Hazm, an open source preprocessing toolkit which includes some major tasks such as normalization, tokenization and POS tagging. Although Hazm outperforms STeP1 and ParsiPardaz toolkits from the run time point of view, its output results are not as accurate as them.

The proposed preprocessing toolkit in this paper provides different kinds of tasks including normalization, tokenization, stemming, POS-tagging and shallow parsing. Moreover, Parsivar is publicly available on the web for research purposes[1]. We have compared the performance of our toolkit with ParsiPardaz and Hazm in an extrinsic evaluation platform using a Persian plagiarism detection algorithm. The results showed that our toolkit has a higher performance when compared with the other toolkits while its runtime is near to Hazm toolkit.

[1] https://github.com/ICTRC/Parsivar

3. Our Approach

Parsivar is an integrated package written in python which performs different kinds of preprocessing tasks in Persian. It should be noted that this toolkit allows for an adjustment between speed and accuracy depends on the user needs. Each task is described in detail in the following subsections.

3.1 Normalization and Tokenization

One of the main problems in Persian text processing is the existence of different character encodings in text documents. For example, the word "آب" (water), might have different encodings in different documents that causes the text processing algorithms to consider them as different words. This problem is more obscurant when we deal with various character encodings in punctuations. To solve this problem, for each character we extracted all different encodings from a corpus of text documents with more than 2 million documents which gathered from Persian weblogs.

Another important task in this step is to find word boundaries in documents. In Persian language, multi-token words can be written in three formats; completely separated by a space delimiter, separated by half-space[2], or be attached to each other. Therefore, determining word and phrase boundaries is a complicated task in Persian. The challenge in Persian is that the space cannot be considered as the only delimiter in all cases.

There are lots of multi-token words in Persian in which, parts of the word are separated by space. For example, in the sentence "من یک برنامه نویس هستم." (I am programmer.), the word "برنامه نویس" (programmer) might be written in two ways of "برنامه‌نویس" and "برنامه نویس". Although the first form of writing is correct and the second one is incorrect, it's usual to write this word in the second form. In this example, considering space as delimiter causes separation of the word "برنامه نویس" into two tokens "برنامه" and "نویس", while it should be taken as one token "برنامه‌نویس". Actually, the challenge is resulted from incorrect spacing in the words. We have tried to solve this problem in two different ways. First, we defined some rules to correct spacing within words. Second, we have tried to train a model that learns how to correct spacing within words.

After the space correction, we tokenize the documents based on spaces and punctuations. In the following subsection, we describe our solutions for space correction in more details.

3.1.1 Rule based space correction

To correct the spaces within words, some certain rules have been defined in the first step using regular expressions. Using these rules, we are able to correct space in many cases such as "می روم" (I'm going), "زمین شناس" (geologist) and "تحلیل گر" (Analyzer). There are still some words that do not match to the rules. These words usually consist of two or three parts which we can't extract a general rule for them such as "گفت و گو" (Conversation). To overcome this problem, we construct a dictionary containing such words and check their existence in the sentences.

3.1.2 Space correction based on learning

To correct the spaces within multi-token words based on learning methods, a model was trained to find words with multiple parts separated by spaces. As a result, we can take all parts of a word as one token.

To train the model, we build a training set using 90% of Bijankhan corpus (Bijankhan et al. 2011). In this corpus, multi token words are placed in one line. We tagged the multi token words using IOB tagging format (Ramshaw and Marcus, 1995) such that the first part is tagged with label "B" and the other parts are tagged with the label "I". Moreover, the other words are tagged with label "O". Then for each word in the sentence, we take the label of previous word, the previous word itself and the next word as features. We trained a Naïve Bayes model to classify each part of the word into classes "B" and "I". At last, we used these labels to find word boundaries. To evaluate the performance of this space correction model, we validated the model on the remaining 10% of Bijankhan corpus. Our model got 96.5% of F1 score in space correction on the validation set.

3.2 Stemmer

Stemming plays an important role in many NLP applications such as information retrieval and text mining. The final goal in stemming is to reduce words to their stem so that for different word forms in a text file, there would be only one stem (Willett, 2006). It is not necessary for the reduced form of the word to be exactly the morphological root. Instead, any other form that improves the performance is acceptable (Krovetz, 1993) There are many stemming algorithms proposed in English. Lovins stemmer (Lovins, 1968) and Porter stemmer (Porter, 1980) are two common stemmers in English. They remove suffixes and prefixes from English words based on some predefined linguistic rules. One of the problems with rule-based stemmers is that they cannot be applied to other languages. Some algorithms are also proposed for stemming in Persian (Sharifloo and Shamsfard, 2008; Taghva, Beckley and Sadeh, 2005). In this section, we propose and implement an algorithm for stemming in Persian language.

Persian words usually derive from other words based on some morphological rules. For example, the word "ستارگان" (stars) is made up of adding suffix "گان" (gan) to the noun "ستاره" (star). We have used such rules to find word stems. In this way, we assumed two set of rules which consider words as verb and non-verb. In the following subsection we describe each of them in detail.

3.2.1 Stemming the Verbs

There are two main roots for present tense and past tense in Persian which can be used to construct various derivations of a verb. For this reason, we collected a list of verb roots in past tense and present tense forms. Then all of the rules were applied to the input verb in order to find the rule to be matched in the best way. Then we search for the resulted roots in the verb dictionary. The first root that is found in the dictionary returns as a word's stem. Some

[2] A Non-Joint Zero Width (NJZW) letter

of the construction rules are shown in Table 1 (Note that the Persian text are read from right to left).

There are also other kinds of verbs which is called prefix verbs. Similar rules like those shown in Table 1 have been used to construct these kinds of verbs, except that a prefix is added at the beginning of the verb. For these kinds of verbs, the algorithm checks for the existence of pre-defined prefix at the first step. After finding and removing the prefixes, the algorithm searches for the stem of the rest of the word in a recursive process.

Table 1 : Some rules for construction of verbs in Persian

Rule	Example
می + بن ماضی + شناسه ماضی (/mi/ + past root + past person identifier)	می‌رفتم (I was going)
می + بن مضارع + شناسه مضارع (/mi/ + present root + present person identifier)	می‌روم (I'm going)
خواه + شناسه مضارع + بن ماضی (/khah/ + present person identifier + past root)	خواهم رفت (I will go)

Algorithm 1 shows the process of finding the verb stem of a word.

Algorithm 1: Verb Stemming

Input: w
foreach $rule$ in $verb\_construction\_rule$ **do**
  if w matches the $rule$ **do**
    $w' = root\ of\ w$
    if w' *is in* $verb\_dict$ **then**
      add w' to $candidate\_set$
    end
  end
end
if $candidate\_set$ is not empty **then**
  return shortest word in $candidate\_set$ as stem
else
  return w
end

3.2.2 Stemming the non-Verbs

We assume that the prefixes and suffixes in non-verbs have a pattern like follows: (to be read from right to left)

{possessive suffix}{plural suffix}{other suffixes}**[stem]**{prefixes}

Note that the items in the brackets are optional.

{Possessive suffix} are suffixes that express ownership and {plural suffix} represents the plurality of the word (Taghva, Beckley and Sadeh, 2005). {Other suffixes} also represent all other kinds of suffixes and {prefixes} shows all kinds of prefixes that would be in the structure of a word. As an example in the word "کتابهایشان" (their books), the possessive suffix "شان" (their) represents the ownership and the suffix "ها" represents the plurality of the word "کتاب" (book). So we have:

"کتابهایشان" (their books)= "شان" (their)+ "ها" (plurality sign)+ "کتاب" (book).

Moreover, in the words "نیرومند" (powerful) and "بی‌صدا" (quiet), the suffix "مند" and the prefix "با" are examples of the "other" category of suffixes.

To find the stem, a list of common suffixes and prefixes for each prefix/suffix category was created. Then for a given word, the algorithm checks for the existence of all the suffixes and prefixes in the list. In the case of finding a suffix/prefix, it would be removed from the word. Then the algorithm checks if the resulted word exists in the lexicon dictionary. If it exists, it returns the resulted word as a stem. Otherwise, other categories of suffixes and prefixes will be checked. The lexicon dictionary is made up of 21151 usual stem words in Persian.

For some cases in which there are more than one prefix/suffix matching of the word, we remove the one which results the smallest stem. This process is described in more details in Algorithm 2.

Algorithm 2: non-Verbs Stemming

Input: w
foreach $suffix\_set/prefix\_set$ **do**
  foreach s in $suffix\_set/prefix\_set$ **do**
    if w ends with s **then**
      $w' = w[0:(len(w) - len(s))]$
      if w' is in $Lexicon\_dict$ **then**
        add w' to $candidate\_set$
      end
    end
  end
end
if $candidate\_set$ is not empty **then**
  return shortest word in $candidate\_set$ as stem
else
  return w
end

In this algorithm we check the existence of each set of suffixes including {possessive suffix}, {plural suffix}, {other suffixes} and {prefixes}, respectively. After removing all of the found suffixes/prefixes, the algorithm searches for the reminder of the word in the lexicon dictionary.

3.3 POS-tagger

Part of Speech Tagging is a preprocessing step in NLP tasks that assign one of the parts of speech tags to the given word. For example, Part of Speech tags for English sentence "I go to school." and its corresponding Persian translation are shown in Table 2.

Table 2: POS-tags for an English sentence with its corresponding Persian translation

Persian Sentence	.	می روم	مدرسه	به	من
English Sentence	.	Go	School	to	I
POS Tags	Punctuation	Verb	Noun	preposition	Noun

The correct assignment of Part-of-Speech tags is an important issue in semantic analysis and syntax parsing. It can also be used as a suitable feature in Natural Language Processing tasks such as Named Entity Recognition, Statistical Machine Translation and also chunking.

We incorporate Maximum Entropy (ME) and Conditional Random Fields (CRF) that has been proved to get successful results in sequence labeling problems such as POS tagging, Name Entity Recognition (NER), chunking, etc. (Lafferty, McCallum and Pereira, 2001; Ratnaparkhi, 1996). ME and CRF are supervised classifiers with a probabilistic approach which determine the most probable tag of a token given its surrounding context (Pisceldo, Adriani and Manurung, 2009).

If we assume $S = s_1, s_2, \dots, s_n = s_1^n$ as a sentence containing n words, ME model estimates the probability of a tag sequence $T = t_1, t_2, \dots, t_n = t_1^n$ as

$$p(t_1^n|s_1^n) = \prod_{i=1}^{n} p(t_i|t_1^{i-1}, s_1^i) \approx \prod_{i=1}^{n} p(t_i|h_i), \quad (1)$$

Where h_i is a contex window of word s_i. Assuming a context window of size 2, the probability of equation (1) becomes as:

$$p(t_1^n|s_1^n) \approx \prod_{i=1}^{n} p\left(t_i\middle|t_{i-2}^{i-1}, s_{i-2}^{i+2}\right), \quad (2)$$

At the last stage, the model assigns maximum likelihood tag sequence to the words of the sentence (Toutanova and Manning 2000). CRF is a probabilistic graphical model tries to estimate the conditional probability $p(T|S)$ based on some independency assumptions (Lafferty, McCallum, and Pereira 2001).

An annotated corpus is needed for training phase of these supervised methods. To generate a Persian POS-tagger, the Bijankhan corpus (Oroumchian et al. 2006) has been used. The Bijankhan collection contains more than 2.6 million manually tagged words that have been labeled with a collection of 550 tag sets. We omitted more fine grained POS-tags and just used a tag set containing 40 tags. We also applied Stanford POS-tagger (Toutanova et al. 2003; Toutanova and Manning, 2000) that is based on ME model and Wapiti tool that is based on CRF model (Lavergne, Cappé and Yvon, 2010).

For getting reliable results, we trained the model with different sets of n-gram features and got the best feature set for Persian POS-tagger. In the experiments, we found that a feature set comprised of two preceding words, the current word, two following words and two preceding tags are suitable for POS-tagging.

3.4 Shallow Parsing

Shallow parsing or text chunking is a subtask of NLP applications that is used as an alternative to full-sentence parsing (Muñoz et al. 2000). The goal of the text chunking is to divide a sentence into some distinct phrases in a way that syntactically related words grouped as one phrase. These phrases don't have overlap with each other, i.e. a word can only belong to one phrase or chunk (Ramshaw and Marcus, 1995; Tjong Kim Sang, 2002). As an example, in the following sentence the chunks are represented in brackets and each chunk is specified with a label which denotes its type.

$[_{NP}$ I$][_{VP}$ saw$][_{NP}$ the yellow umbrella$]$

Generally speaking, the models proposed to solve this problem are based on pre-defined rules (Grover et al. 2006) or machine learning techniques (Muñoz et al. 2000; Zhai et al. 2017). There have been few efforts to solve this problem in Persian and there aren't any suitable dataset to train a statistical model. In this section, we employed some linguistic rules to find chunks in Persian sentences using regular expressions. These rules are based on POS-tags of the words in the sentence. For example one of the rules that is used in our model is:

NP: {<N_SING><ADJ_SIM><N_SING>}

It means that a singular noun following a simple adjective following another singular noun creates a noun phrase. The following is an example:

- "ضخامت دقیق سیاره"
- "Exact thickness of the planet"

In our approach, we have extracted fifteen linguistic rules to find Persian noun phrases, verb phrases and prepositional phrases. The main advantage of using linguistic rules is that it doesn't need a training corpus. The disadvantage of this approach is that extracting a set of rules which models language complexities is a challenging issue. Moreover, the use of a rule set causes the approach to be language dependent.

4. Experimental Results

The main challenge in evaluation of different parts of the model is the limitation of available resources. For this reason, the performance of some parts of the toolkit were checked manually. To evaluate and compare the performance of Parsivar with the other available toolkits, different types of experiments were performed. For evaluating the tokenization and stemming, we used 10 random documents from the Hamshahri corpus (AleAhmad et al. 2009) which consist of 2552 tokens. After removing the stop words, 1465 words are remained. The accuracy of tokenization tool was 98.29% among these words. Table 3 also shows the evaluation results of stemming among these words.

Table 3 : Evaluation results of Stemmer

	Precision[%]	Recall[%]	F-Measure[%]
Stemmer	98.71	81.91	89.53

To train the space correction model mentioned in section 3.1.2, an IOB tagged dataset has been built using Bijankhan corpus. The dataset contains 2428732 words tagged with "O", 160775 words tagged with "B" and 169518 words tagged with "I". Then it considered as a sequence labeling problem and a Naïve Bayes classifier was trained using 90 percent of this dataset to classify each word into classes "I", "O" and "B". Using these labels, we can specify word boundaries. To test the performance, we validated the model with the remaining 10 percent of the dataset. The results are shown in Table 4.

Table 4 : Evaluation results of space correction model

IOB Accuracy [%]	Precision[%]	Recall[%]	F-Measure[%]
93.2	87.3	91.9	89.5

For the POS tagger, as mentioned in section 3.3, two types of POS taggers have been trained. The training on ME and CRF methods were done with one million words of Bijankhan corpus. These models were tested using 10 percent of Bijankhan corpus with different window sizes. At last, the best one of each type was chosen. Table 5 shows the test accuracy for each model. The accuracy was measured in two levels. In the word level, it presents the correctness of the tagging for each word and in sentence level it presents the correctness of the tagging for a complete sentence.

Table 5 : Evaluation results of POS tagger

		Window size	
		3	5
Maximum Entropy	Word Accuracy [%]	0.91	0.95
	Sentence Accuracy [%]	0.75	0.78
CRF	Word Accuracy [%]	0.93	0.95
	Sentence Accuracy [%]	0.76	0.79

To evaluate the Shallow Parser mentioned in section 3.4, we applied it on 100 randomly selected sentences from Hamshahri corpus. Based on the manually evaluation of the results, the performance was 76.8%. Since this model is based only on POS-tags extracted from sentences, a part of the error is the error propagated by POS-tagger.

We also tried to measure the effect of applying different parts of Parsivar in a downstream NLP task such as plagiarism detection. In the task of plagiarism detection, the goal is to find parts of a text which have been reused from other documents (Asghari et al. 2016). The process starts with a suspicious document d_q and a collection D of documents from which d_q's author may have plagiarized. Within a heuristic retrieval step, a small number of candidate documents D_x, which are likely to be sources for plagiarism, are retrieved from D. Then, within a detailed analysis step, d_q is compared section-wise with the retrieved candidates. All pairs of sections (s_q, s_x) with $s_q \in d_q$ and $s_x \in d_x$, $d_x \in D_x$, are to be retrieved such that s_q and s_x have a high similarity under some retrieval model (Potthast et al. 2010). For this purpose, we performed a preprocessing step on the input text using different toolkits including Hazm, ParsiPardaz and Parsivar. In the next step, we used the output of each toolkit as the input to the Persian plagiarism detection model. At the last step, we compared the results of the plagiarism detection algorithm for each preprocessing toolkit. In our experiments, we used a part of plagiarism detection corpus introduced in (Mashhadirajab et al. 2016) and also (Khoshnavataher et al. 2015) as well.

For plagiarism detection model, we used a VSM based method proposed in (Zechner et al. 2009). In this model, all the sentences of both suspicious and source documents are converted into vectors using TF-IDF weighting method. Then, all sentences of the suspicious document are compared to all sentences of source ones using cosine similarity metric. Pairs of sentences which are similar (based on a pre-defined threshold) have been considered as cases of plagiarism.

The experimental results of plagiarism detection are depicted in Figure 1. Each curve specifies the F-score of the plagiarism detection model for a particular preprocessing toolkit at a specific similarity threshold.

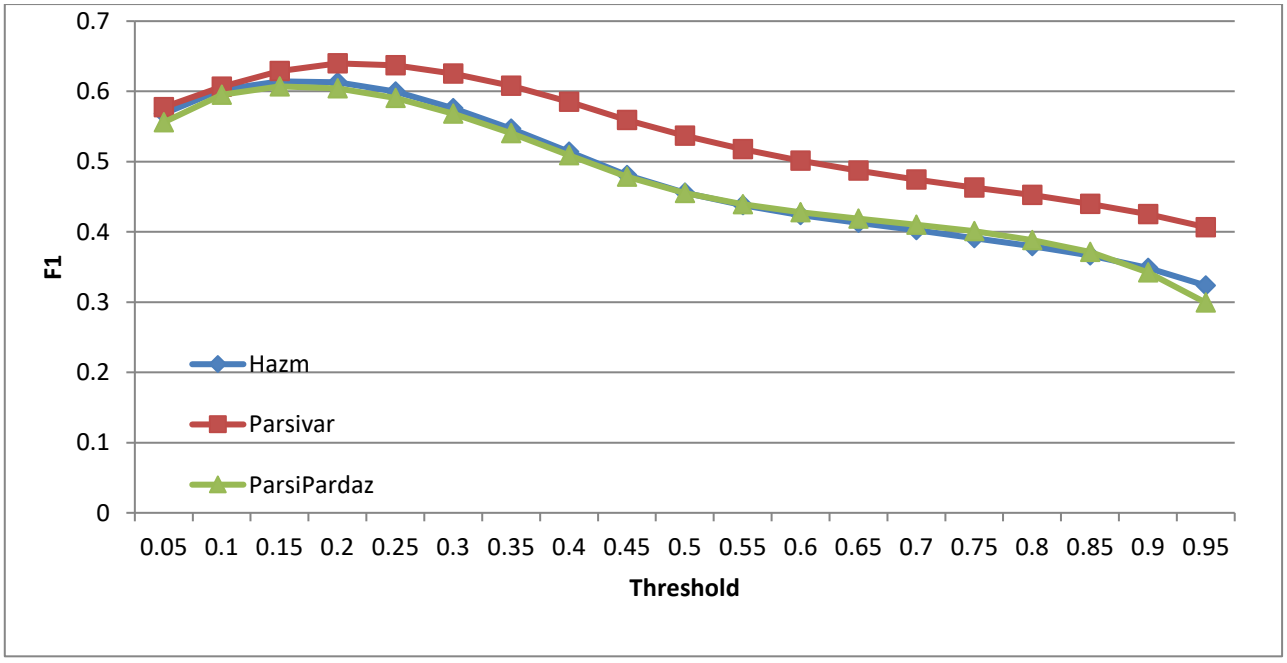

Figure 1: The F-score of a plagiarism detection algorithm for each preprocessing toolkit in various similarity thresholds.

The character level F-measure has been used for evaluating performance of detection based on following equations:

$$Precision\ (S,R) = \frac{1}{|R|}\sum_{r\in R}\frac{\bigcup_{s\in S}(s\sqcap r)}{|r|} \quad (3)$$

$$Recall\ (S,R) = \frac{1}{|S|}\sum_{s\in S}\frac{\bigcup_{r\in R}(s\sqcap r)}{|s|} \quad (4)$$

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (5)$$

Where S denotes the set of plagiarism cases in the suspicious documents, and R denotes the set of plagiarism that detected by detector for these documents.

It should be noted that the plagiarism detection model is the same for all experiments. As depicted in Figure 1, our toolkit outperforms the other preprocessing toolkits for all threshold values.

The main advantage of our preprocessing toolkit over the other toolkits is in normalization and tokenization steps. As they are used in almost all NLP tasks, the errors generated in these steps will propagate into the other stages and cause more error generation. Unification of the character encodings and correcting the spaces between/ within the words (which results in a better detection of word boundaries) are some reasons which cause a better performance in our toolkit with respect to the other preprocessing toolkits.

5. Conclusion

In this paper we proposed Parsivar, a Persian preprocessing toolkit written in python. This package provides some important preprocessing tasks for NLP purposes such as normalizing, tokenizing of words and multi-token words, stemming, POS tagging and shallow parsing. In the normalization step, we convert different kinds of character encodings to a unique format. In tokenization step, we correct the spacing between and within the words and multi-token words as well. In stemming and shallow parsing, we defined some rules to solve the problem. Finally, in POS-tagging, we trained a CRF and ME based model. As a final stage of this research, we tested our toolkit on a Persian plagiarism detection system. The results show that our toolkit outperforms other similar Persian preprocessing toolkits with respect to F-score.

As a work for the future, we are planning to recognize Persian Ezafe tags as in (Asghari, Maleki and Faili, 2014) to our toolkit. Another work can be accomplished to improve the performance of the shallow parser using statistical approaches.

6. References

AleAhmad, A., Amiri, H., Darrudi, E., Rahgozar, M., & Oroumchian, F. (2009). Hamshahri: A standard Persian text collection. *Knowledge-Based Systems*, 22(5), 382-387.

Asghari, H., Mohtaj, S., Fatemi, O., Faili, H., Rosso, P., & Potthast, M. (2018, February). Algorithms and Corpora for Persian Plagiarism Detection. In *Text Processing: FIRE 2016 International Workshop*, Kolkata, India, December 7–10, 2016, Revised Selected Papers (Vol. 10478, p. 61). Springer.

Asghari, H., Maleki, J., & Faili, H. (2014). A probabilistic approach to persian ezafe recognition. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2: Short Papers (pp. 138-142).

Bijankhan, M., Sheykhzadegan, J., Bahrani, M., & Ghayoomi, M. (2011). Lessons from building a Persian written corpus: Peykare. *Language resources and evaluation*, 45(2), 143-164.

Grover, C., & Tobin, R. (2006, May). Rule-based chunking and reusability. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation* (LREC 2006).

Hazm. (2014). Python library for digesting Persian text, "Https://github.com/sobhe/hazm."

Khoshnavataher, K., Zarrabi, V., Mohtaj, S., & Asghari, H. (2015). Developing monolingual Persian corpus for extrinsic plagiarism detection using artificial obfuscation. *Notebook for PAN at CLEF*.

Krovetz, R. (1993, July). Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 191-202). ACM.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lavergne, T., Cappé, O., & Yvon, F. (2010, July). Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 504-513). Association for Computational Linguistics.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2), 22-31.

Mashhadirajab, F., Shamsfard, M., Adelkhah, R., Shafiee, F., & Saedi, C. (2016). A Text Alignment Corpus for Persian Plagiarism Detection. *In FIRE (Working Notes)* (pp. 184-189).

Munoz, M., Punyakanok, V., Roth, D., & Zimak, D. (2000). A learning approach to shallow parsing. *arXiv preprint cs/0008022*.

Oroumchian, F., Tasharofi, S., Amiri, H., Hojjat, H., & Raja, F. (2006). Creating a feasible corpus for Persian POS tagging. UOWD Technical Report

Series.

Pisceldo, F., Manurung, R., & Adriani, M. (2009). Probabilistic part-of-speech tagging for bahasa indonesia. In *Third International MALINDO Workshop.*

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3), 130-137.

Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., & Rosso, P. (2010). Overview of the 2nd international competition on plagiarism detection. In *Proceedings of the 4th Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse* (pp. 1-14).

Ramshaw, L. A., & Marcus, M. P. (1999). Text chunking using transformation-based learning. In *Natural language processing using very large corpora* (pp. 157-176). Springer, Dordrecht.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing.*

Sarabi, Z., Mahyar, H., & Farhoodi, M. (2013, October). ParsiPardaz: Persian Language Processing Toolkit. In *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on* (pp. 73-79). IEEE.

Shamsfard, M., Jafari, H. S., & Ilbeygi, M. (2010, May). STeP-1: A Set of Fundamental Tools for Persian Text Processing. In *LREC*.

Sharifloo, A. A., & Shamsfard, M. (2008). A bottom up approach to Persian stemming. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II.*

Taghva, K., Beckley, R., & Sadeh, M. (2005, April). A stemming algorithm for the farsi language. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on* (Vol. 1, pp. 158-162). IEEE.

Tjong Kim Sang, E. F., & De Meulder, F. (2003, May). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (pp. 142-147). Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003, May). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1* (pp. 173-180). Association for Computational Linguistics.

Toutanova, K., & Manning, C. D. (2000, October). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13* (pp. 63-70). Association for Computational Linguistics.

Willett, P. (2006). The Porter stemming algorithm: then and now. *Program*, 40(3), 219-223.

Zechner, M., Muhr, M., Kern, R., & Granitzer, M. (2009, September). External and intrinsic plagiarism detection using vector space models. In *Proc. SEPLN* (Vol. 32, pp. 47-55).

Zhai, F., Potdar, S., Xiang, B., & Zhou, B. (2017, January). Neural Models for Sequence Chunking. In *AAAI* (pp. 3365-3371).