Introducing E4MT and LMBNC: Persian pre-processing utilities

Zakieh Shakeri^{1}, S.Mehran M.Ziabary^{1}, Behrooz Vedadian^{1},

Fatemeh Azadi^{1}, Saeed Torabzadeh^{2}, Ariyan Atefi^{1}

1 Targoman Intelligent Processing Co. Pjc., Tehran, Iran
{z.shakeri, ziabary, vedadian, f.azadi, a.atefi}@targoman.com

2 Amirkabir University of Technology, Tehran, Iran
saeed.torabzdeh@aut.ac.ir

December 13, 2021

Abstract

In this paper, we introduce two utilities, extensively used in our services and products. A Persian pre-processor(E4MT) we use for both training and inference in our machine translation services and a corpora-level language model-based error corrector(LMBNC), which we apply to corpora before training. E4MT(Essential tools for MT) consists of character normalization, spell correction, entity tagging, and tokenization/detokenization modules. It handles the Persian large vocabulary size problem by approximately reducing the vocabulary size by a factor of 2. Applying E4MT on the English-Persian translation task, yields an improvement of 1.51 BLEU over Parsivar. We apply LMBNC on the training corpora, which uses a domain-specific language model to identify context-dependent misspellings. The results show, using this corrected training corpora improves the English-Persian translation quality by 0.6 BLEU over its baseline. Additionally, the manual evaluation shows 97.9% precision for E4MT and 98.1% precision for LMBNC.

1 Introduction

In recent years, NLP models have improved significantly by using deep learning methods. The existence of large amounts of data is one of the essential requirements for deep learning. Powerful and huge pre-trained NLP models like BERT [1] and GPT2 [2] can be trained due to the existence of such amounts of data. Working with low-resource languages, having a suitable linguistic toolkit helps increase the quality of existing corpora and compensate for data size.

In this paper, we introduce two utilities(E4MT[1] and LMBNC[2]), which served well improving the quality of our corpora and consequently our MT systems. Both of them are publicly available as open source tookits.

Since the Persian alphabet is based on the Arabic alphabet as do many other middle eastern languages, many non-Persian characters have been entered into Persian written text. Also, OCR tools often use glyph codes in case of Persian language instead of the canonical characters. So the same letter can be found in various forms (for example, the Persian letter “ ک” /ke, has 32 different Unicodes!). This can enlarge the

[1]https://github.com/Targoman/E4MT
[2]https://github.com/Targoman/LMBNC

vocabulary size of the Persian language, which causes problems in NLP-based tasks. Our E4MT normalizer converts these different forms of characters to their canonical form. This operation covers the all UTF-8 2-byte characters.

Persian is an agglutinative language. Many verbs are constructed by appending a prefix and a suffix to a stem. Most adjectives and adverbs have at least one affix in their body. In written text, these affixes can be separated from their base by `<zwnj>`[3] character. Different authors have different manners of dealing with this character (using, deleting, or replacing it with space character). This causes the same Persian entity to have different written forms. We handle this kind of misspellings with our spelling correction tool in E4MT.

Although E4MT is tuned for the Persian language, it can be used for other languages as well.

In low-resource languages like Persian, the errors in training corpora can affect the QoS(quality of service) and user experience. We have encountered a few cases of dissatisfaction of this nature where translation quality was heavily shadowed by obvious misspellings. Inspecting the problem, we noticed there are incorrect n-grams in our training corpora; some can not be corrected by simple rules and need the context to be considered. We utilize a domain-specific language model to correct these mistakes.

The remainder of the paper is organized as follows. In the next section, we review some related works. E4MT is explained in section 3 and in section 4 we explain LMBNC. In section 5 our experiments are reported and finally section 6 concludes.

2 Related work

There are some efforts to prepare toolkits for the Persian language, which can be used in the pre-processing step of NLP-based tasks but to the best of our knowledge, there is no Persian toolkit similar to LMBNC that uses a language model to correct context dependant misspellings. STeP-1 introduced by [3] contains tokenizer, spelling checker, and morphology analyzer. [4] introduced ParsiPardaz, which contains a tokenizer, lemmatizer, POS tagger and dependency parser. Another toolkit is Hazm[4] introduced by Sobhe, which contains normalizer, tokenizer, and POS tagger. Also, Parsivar[5] toolkit, which contains normalizer, tokenizer, and POS tagger plus shallow parser and stemmer. Among these toolkits, Hazm and Parsivar are publicly available as open source projects, and we compare E4MT with them. We examined the impact of each toolkit on the machine translation task, which is described in section 5 in detail. The results show improvements in BLEU measure over both Hazm and Parsivar baselines.

3 E4MT

E4MT is a C++ implementation of fundamental tools for Persian language . It consists of character normalization, spell correction, tokenization/detokenization, and entity tagging modules. First, input text characters are converted to their canonical form, and then various forms of misspelling are corrected using language-specific sets of rules. Finally, a limited set of entities (such as numbers, URLs, emails, list bullets and numberings, abbreviations, ...) is tagged to enable us to identify the punctuations and end of sentences.

3.1 Normalization

As mentioned before, since the Persian alphabet is based on the Arabic alphabet similar to many other middle eastern languages, many Arabic characters are entered into Persian written text. Also, there are some characters from Urdu and Pashto alphabet very similar to Arabic characters but with a different encoding and there are some multi-character words encoded using just one codepoint. For example the word “ سنه” (meaning ‘year’), is encoded sometimes as “ ؁” instead of the three-character canonical form. Furthermore, OCR tools often generate incorrect characters. These cause different written forms for the same entity. For example, the word “ نتیجه‌گیری” (meaning ‘conclusion’), appears 30,628 times in our corpora but in 22 different forms! Table 1 shows most of these different forms and their occurrence frequencies and the corrections made by the normalizer and spelling corrector.

E4MT normalizer converts a character to its canonical form according to the previous character, the next character, and a set of predefined lists (allow-list, block-list, and replacement-list).

[3]Zero-Width Non-Joiner(<0x200c>)
[4]Https://github.com/sobhe/hazm

Written form	**Corr. operation**	**Module**	**Freq.**
نتیجه + <sp> + گیری	<sp> → <zwnj>	SC	29,763
نتیجه + <sp> + <zwnj> + گیری	<sp> → <deleted>	SC	430
نتیجه + <zwnj> + <sp> + گیری	<sp> → <deleted>	SC	306
نتیجه + <sp> + <zwnj> + <zwnj> + گیری	[<sp> → <deleted>,	SC	35
	<zwnj> → <deleted>]	SC	
نتیجه + <zwnj> + <zwnj> + گیری	<zwnj> → <deleted>	SC	16
نتیجه + <zwnj> + <zwnj> + <sp> + گیری	[<sp> → <deleted>,	SC	13
	<zwnj> → <deleted>]	SC	
نتیجه + <ltrm> + <sp> + گیری	[<sp> → <deleted>,	SC	11
	<ltrm> → zwnj]	SC	
نتیجه + <sp> + <dash> + گیری	[<sp> → zwnj,	SC	9
	<dash> → <deleted>]	SC	
نتیجه + <sp> + <ltrm> + گیری	[<sp> → <deleted>,	SC	8
	<ltrm> → zwnj]	SC	
Glyphs of “ نتیجه گیری”	[0xfee7 → < ن> ,	N	8
	0xfe98 → < ت> ,	N	
	0xfef4 → < ی> ,	N	
	0xfea0 → < ج> ,	N	
	0xfeea → < ه> ,	N	
	<sp> → <zwnj>,	SC	
	0xfb94 → < گ> ,	N	
	0xfef4 → < ی> ,	N	
	0xfeae → < ر> ,	N	
	0xfef4 → < ی>]	N	
نتیجه + <sp> + گیری	[<sp> → <zwnj>,	SC	7
	< ی>(0x649) → < ی>(0x6cc)]	N	

Table 1: Example of applying normalization and spell correction on different written forms of نتیجه‌گیری and their occurrence frequencies. Modules(SC: Spelling corrector, N:Normalizer).

Allow-list characters remain unchanged in the output, e.g., [a-zA-Z], [0-9], [ا-ی]. Block-list characters are removed, e.g., Arabic diacritics (ُ ً ٍ ٌ), Invalid control characters{<0x1>-<0x8>, ...}, and replacement-list characters are replaced by their defined replacements e.g., “ أ, ؤ, إ, ۀ ” are replaced by “ ا, و, ا, ه ” and all digits from any language are converted to Latin digits, etc. These replacements are not just for Persian/Arabic characters but also include similar Latin and punctuation characters. For example it converts “ <0xFF41>” to “ <0x41>” and “ <0xFF5B>” to “<0x7B>”.

Although most characters can be converted using these predefined lists, some need additional information. For example, using `<zwnj>` before a character that is not left joinable, or before a digit or punctuation is useless. For removing these extra `<zwnj>`s, we define some rules that, according to the character itself and its previous and next characters, identify and correct these mistakes.

3.2 Spell correction

Persian language is an affixal system consisting of suffixes and prefixes. Different authors write these affixes in different manners. Some use `<zwnj>` to concatenate the affix to its base. Some omit it, and others use space character instead; There are also plenty of control characters used instead of `<zwnj>`. So in most of the Persian texts, there is a mix of `<zwnj>`, and other characters used instead of `<zwnj>`. Again, this causes the same Persian entity to have different written forms; e.g. the word “ بامعرفیت‌ترین‌های‌شان” can be found in 54 different written forms in text! We address this issue in the spell correction module. Using predefined lists containing verb stems, adjectives, adverbs, and nouns, the spell correction module examines n-grams present in the sentence to determine if it can be decomposed using any valid inflection rule. The identified segments are then concatenated. For example, different correction operations for " نتیجه‌گیری" can be found in table 1.

Also, some common spelling mistakes frequently encountered in Persian digital contents are fixed. For example, a common mistake in building PDF from Persian documents is that glyph of characters are saved instead of their Unicodes, or some mandatory `<zwnj>` between characters are deleted. This causes problem in converting PDFs to text. Our spelling corrector converts these glyphs to their proper forms and inserts missing `<zwnj>` between characters. Beside there are common misspellings on some words, which we automatically correct using a huge dictionary of correction rules.

3.3 Entity tagging

For most NLP-based tasks, tokenization is a mandatory module to be applied in the preprocessing step; Some of them also need sentence-level corpora for training, like most recent machine translation or question answering systems. E4MT can split blocks of text into sentences with the help of its entity tagger. This tagger uses a set of predefined rules to tag a limited set of entities. Using these entities, we identify the punctuations and end of sentences.

Hand-tailored regular expressions are used to identify and tag numbers, email addresses, abbreviations, dates, lists, and URLs. Having tagged these entities, all remaining punctuations and non-text symbols can be safely separated from their neighboring characters. We can also break the text upon finding any of [.?!] without mistakenly breaking any sentence.

Identified entities can be further used to simplify the corpora and limit the vocabulary size. For example, numbers in all the sentences can be replaced by a symbol representing a number entity. This will remove all instances of numbers from the vocabulary and insert just one. Other entity tags like URL can be used for simplifying the corpora like numbers.

Furthermore, our tokenization/detokenization modules use spelling correction and entity tagging to tokenize/detokenize, and at the same time convert Latin/Arabic numbers and punctuations to their equivalent.

4 LMBNC

Some of the incorrect n-grams in our training corpora were generated by concatenation of two or more continuous uni-grams, like: " اینمطالعه " which is a concatenation of " این" (meaning 'this'), and " مطالعه" (meaning 'study'), and, some by splitting a uni-gram, like: " کام لا", which is " کاملا" split into two unigrams.

4.1 Incorrect uni-grams

These misspellings are two continuous unigrams concatenated to each other mistakenly. Some of them are meaningless words in the Persian language, like: " اینافراد" , which is a concatenation of " این" (meaning 'these') and " افراد" (meaning 'people'); and some are meaningful, like “ سرو” (meaning ‘cypress’) which can also be a concatenation of “ سر” (meaning ‘head’) and “ و” (meaning ‘and’).

Finding these kinds of misspelling is challenging and can be done only by considering the context. The following example, which is selected from our corpora, clarifies this.

In sentence:

“بیشتر ضایعه‌ها در سرو گردن قرار داشت . ”

all of the words are meaningful, but the word “ سرو” should be corrected to “ سر <sp> و” according to the sentence context: ’most of lesions were located in head and neck .’

4.2 Incorrect bi-grams

Incorrectly split unigrams generate these bigrams. Again, these kinds of corrections can be done only by considering the context.

Consider the following sentences and their corrected form obtained by our method:

Sentence #1: برای تقلیل هزینه‌های سالانه دیابت با ید در درجه اول به بررسی مشخصات مراقبتهای درمانی ارایه‌شده و چگونگی خود مراقبتی در جمعیت مبتلا به دیابت پرداخت .

Corrected sentence #1: برای تقلیل هزینه‌های سالانه دیابت باید در درجه اول به بررسی مشخصات مراقبتهای درمانی ارایه‌شده و چگونگی خود مراقبتی در جمعیت مبتلا به دیابت پرداخت .

Sentence #2: هدف مطالعه حاضر مقایسه دو روش درمان با ید رادیواکتیو و داروهای ضد تیروویید است .

Corrected sentence #2: هدف مطالعه حاضر مقایسه دو روش درمان با ید رادیواکتیو و داروهای ضد تیروویید است .

Bigram “ با ید” exists in both sentences, But in sentence #1, it is incorrect and is changed to “ باید” (meaning ‘should’), and in sentence #2, it is correct (meaning ‘with Iodine’), and it remained unchanged.

4.3 Proposed correction method

We utilize a domain-specific language model to correct these mistakes. First we extract all possible alternatives for the words of the sentence as described in Algorithm 2, and then the correct form of the sentence is determined using the prepared language model(Algorithm 1).

```
Algorithm 1 correctNgrams(corpus)
for all $sentence$ in $corpus$ do
    $alternatives \leftarrow extractAlternatives(sentence)$
    $scores \leftarrow LM.score(alternatives)$
    $correctForm \leftarrow argmax(scores)$
end for
```

Our approach corrects not only Persian n-grams but also incorrect English n-grams as well. For example, the incorrect word ”rhizoctoniasolani” is corrected in the sentence below:

Original sentence:

مهم‌ترین عوامل این بیماری قارچهای pythiumspp و rhizoctoniasolani می‌باشند .

Corrected sentence:

Algorithm 2 extractAlternatives(sentence)

```
$sentenceAlternatives \leftarrow []$
if $len(sentence) = 0$ then
  return ['']
end if
$i \leftarrow 0$
$w \leftarrow$ first word of the sentence
$firstWordAlternatives \leftarrow []$
while $i \leq len(w)$ do
  $w_1, w_2 \leftarrow$ split $w$ at $i$th character
  if $occurrenceFrequency(w_1$<sp>$w_2) > occurrenceFrequency(w)$ then
    Append $(w_1$<sp>$w_2)$ to $firstWordAlternatives$
  end if
end while
if $len(sentence) = 1$ then
  return $firstWordAlternatives$
else
  for all $alternative$ in $extractAlternatives(sentence[1{:}EOS])$ do
    for all $w$ in $firstWordAlternatives$ do
      Append ($w$<sp>$alternative$) to $sentenceAlternatives$
    end for
  end for
  $w_0 \leftarrow sentence[0]$
  $w_1 \leftarrow sentence[1]$
  if $occurrenceFrequency(w_0w_1) > occurrenceFrequency(w_0$<sp>$w_1)$ then
    for all $alternative$ in $extractAlternatives(sentence[2{:}EOS])$ do
      Append ($w_0w_1$<sp>$alternative$) to $sentenceAlternatives$
    end for
  end if
  return $sentenceAlternatives$
end if
```

مهم‌ترین عوامل این بیماری قارچ‌های pythiumspp و rhizoctonia solani می‌باشند .

Table 2 shows more examples of sentences and their corrected forms after applying the above-mentioned processes.

En	a high percentage of preventable cancers , respiratory and non respiratory diseases are due to cigarette smoking .
Fa	در صد بالایی از سرطان‌های قابل پیش‌گیری , بیماری‌های تنفسی و غیر تنفسی به مصرف سیگار منسوب شده‌است .
Corr	درصد بالایی از سرطان‌های قابل پیش‌گیری , بیماری‌های تنفسی و غیر تنفسی به مصرف سیگار منسوب شده‌است
En	in addition , no significant correlation was found between swimming type and shoulder dysfunction , or between age , age of swimming onset , height , weight , and dominant hand with shoulder problem .
Fa	علاوه براین بین نوع شنا , سن , سن شروع شنا , قد و وزن و دست غالب بادرگیری کمربند شانه از نظر آماریارتباط معنی‌داری وجود نداشت .
Corr	علاوه بر این بین نوع شنا , سن , سن شروع شنا , قد و وزن و دست غالب با درگیری کمربند شانه از نظر آماری ارتباط معنی‌داری وجود نداشت .

Table 2: Examples of corrected sentences, En: English sentence, Fa: original Farsi(Persian) sentence, Corr: corrected sentence by our method

5 Experiments

To investigate the effect of different tools on machine translation task, an English-to-Persian Transformer model [6] was trained using Marian[7] toolkit. The state-of-the-art Transformer model, which is now the predominant model used in most machine translation systems, is an encoder-decoder-based model with an emphasis on the attention mechanism. For all of the MT systems, we process the training, development and test data sets by one of the tools (E4MT, Parsivar, Hazm, and LMBNC) and evaluate the systems using automatic measure BLEU[8], on 10% held-out set from the training data.

5.1 E4MT

We performed two kinds of evaluation for E4MT. An automatic evaluation to explore its effect on machine translation and a manual evaluation to obtain its precision and its effect on vocabulary size.

5.1.1 E4MT impact on machine translation

The experiments has been done on two English-Persian parallel corpora. A publicly available corpora, TED2020[5][9], and an in-house corpora which is about 5 times greater than TED2020 and contains scientific papers published in journals. Table 3 shows the statistics of them. We

	Sentences	En tokens	Fa tokens
Public corpora	300K	5M	5.6M
In-house corpora	1M	23.8M	26.7M

Table 3: Statistics of corpora

trained the translation system three times; each time, we pre-process the corpora with one of the E4mt, Hazm and Parsivar tools. Since Hazm and Parsivar only support the Persian language, we pre-process their English side with Moses tokenizer[6].

As table 4 shows, applying E4MT on TED2020 corpora, yields an improvement of 1.28 BLEU over Parsivar and 1.15 BLEU over Hazm; Also, for in-house corpora, 1.51 BLEU improvement over Parsivar and 1.17 BLEU improvement over Hazm achieved. Since, there is no general detokenizer for Persian language, we calculated BLEU on the tokenized outputs.

	BLEU on TED corpora	BLEU on in-house corpora
Hazm	19.27	18.36
Parsivar	19.14	18.02
E4MT	**20.42**	**19.53**

Table 4: Effect of E4MT, Hazm and Parsivar on machine translation task.

5.1.2 Manual evaluation

Our experiments show that, 4,434,795 Persian tokens of the in-house corpora have been corrected by E4MT, which is about **15.4%** of the corpora running words. To inspect the precision of these corrected words, we select 100 sentences randomly and 97.9% precision was observed.

[5]https://opus.nlpl.eu
[6]https://github.com/moses-smt

E4MT also reduced the vocabulary size by 54% on total running words (64% reduction on singletons and 35% reduction on non-singletons) as table 5 shows.

	Vocab. size	**Singletons**	**Non-singletons**
Original corpora Running words: 27M	662,964	441,937	221,027
Corrected by E4MT Running words: 28M	302,376(**-54.4%**)	156,848(**-64%**)	145,528(**-35%**)

Table 5: Effect of E4MT on vocabulary size

5.2 LMBNC

In this section, we first explain the LM preparation process and then examine the impact of LMBNC on machine translation task.

5.2.1 Preparing language model

We used Fairseq[10] for training language models. We first trained a general-domain LM. A combination of several corpora was used to train the LM using more data. These corpora were from different domains; 75% of them were news and remaining were scientific papers. All of these corpora included about 4M sentences (108M tokens). Using this LM, we tried to correct the mistakes of our domain-specific corpus. Although many context dependent mispellings were identified, the side effect – number of mistakenly changed words – was not negligible.

To mitigate the problem, we trained the LM on the same corpus that we wanted to improve; The Persian side of the in-house corpora mentioned in table 3. The results were desirable where both many incorrect n-grams had been corrected, and the number of incorrectly changed words was negligible.

5.2.2 LMBNC impact on machine translation

To investigate the effect of LMBNC on machine translation, we trained two similar translation systems. The only difference between them was their training corpora. We trained one system with the original version of the in-house corpora, and the other with the corrected-by-LMBNC version. As table 6 shows, an improvement of 0.6 BLEU is achieved.

	BLEU
Baseline	20.28
Corrected by LMBNC	20.87(+**0.59**)

Table 6: Impact of LMBNC on English-Persian machine translation task.

Also, The accuracy of the correction process was evaluated manually. From about 1M sentences, 119,861 sentences were modified including 176,971 corrected words. Manually inspecting 300 randomly selected sentences we observed a **98.1%** precision.

6 Conclusion

In this paper we introduce two utilities(E4MT and LMBNC) extensively used in our services and products. Although we developed E4MT to improve our machine translation quality, but we believe it can be applied to other NLP-based tasks as well. It consists of fundamental tools(character normalizer, spell corrector, and tokenizer/detokenizer) which can address most challenging writing issues of the Persian language . It handles the Persian large vocabulary size problem by approximately reducing the vocabulary size by a factor of 2. We compared E4MT with two publicly available Persian toolkits (Hazm and Parsivar) on machine translation task. Results show an improvement of 1.17 BLEU over Hazm and 1.51 BLEU over Parsivar. Our manual evaluation shows 97.9% precision for it. Also, we introduced LMBNC, which applies to the training corpora and uses a domain-specific language model to identify context-dependent misspellings. Experiments show that applying it on MT task improves the quality by 0.6 BLEU. We also evaluated the quality manually, where we observed 98.1% precision.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[3] Mehrnoush Shamsfard, Hoda Sadat Jafari, and Mahdi Ilbeygi. Step-1: A set of fundamental tools for persian text processing. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *LREC*. European Language Resources Association, 2010.

[4] Zahra Sarabi, Hooman Mahyar, and Mojgan Farhoodi. Parsipardaz: Persian language processing toolkit. *Computer and Knowledge Engineering*, pages 73–79, 2013.

[5] Salar Mohtaj, Behnam Roshanfekr, Atefeh Zafarian, and Habibollah Asghari. Parsivar: A language processing toolkit for Persian. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[7] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[9] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[10] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019.