Generating Regular Expressions from Natural Language Specifications: Are We There Yet?

Zexuan Zhong[†], Jiaqi Guo[‡], Wei Yang[†], Tao Xie[†], Jian-Guang Lou[§], Ting Liu[‡], Dongmei Zhang[§]
[†]Department of Computer Science, University of Illinois at Urbana-Champaign, USA
[‡]Ministry of Education Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, China
[§]Microsoft Research Asia, China
{zexuan2, weiyang3, taoxie}@illinois.edu, jasperguo2013@stu.xjtu.edu.cn, tingliu@mail.xjtu.edu.cn
{jlou, dongmeiz}@microsoft.com

Abstract

Recent state-of-the-art approaches automatically generate regular expressions from natural language specifications. Given that these approaches use only synthetic data in both training datasets and validation/test datasets, a natural question arises: are these approaches effective to address various real-world situations? To explore this question, in this paper, we conduct a characteristic study on comparing two synthetic datasets used by the recent research and a real-world dataset collected from the Internet, and conduct an experimental study on applying a state-of-the-art approach on the real-world dataset. Our study results suggest the existence of distinct characteristics between the synthetic datasets and the real-world dataset, and the state-of-the-art approach (based on a model trained from a synthetic dataset) achieves extremely low effectiveness when evaluated on real-world data, much lower than the effectiveness when evaluated on the synthetic dataset. We also provide initial analysis on some of those challenging cases and discuss future directions.

Introduction

A regular expression is a sequence of characters that define a search pattern. It is very common to use regular expressions in string-searching tasks that play essential roles in various software applications such as information-extraction applications and web applications (with input validators). For example, people may use the regular expression 'A.*' to search for strings that begin with 'A'. Although regular expressions are widely used, writing regular expressions can be time consuming and error prone. However, it is often easier for users to specify their tasks (with regular expressions) in natural language (NL). Therefore, there is a strong need to automatically generate regular expressions from NL specifications. In addition, automatic generation of regular expressions can be further applied for other tasks of formal language synthesis such as synthesizing program scripts (Raza, Gulwani, and Milic-Frayling 2015).

Recent research has already attempted to automatically generate regular expressions from NL specifications. Ranta (1998) proposes a rule-based approach to build an NL interface to regular expressions. Kushman and Barzilay (2013) use a parallel dataset of NL sentences and regular expressions as training data. They train a probabilistic parsing model, which reads a sentence and outputs the top parse tree. They then generate a regular expression from the parse tree. Recently, Locascio et al. (2016) use a sequence-to-sequence learning model to directly translate an NL sentence to a regular expression without applying any domain-specific knowledge. The use of deep neural model in this work reduces the target problem to a general problem of machine translation and serves as a representative state-of-the-art approach. Given that these approaches use only synthetic data in both training datasets and validation/test datasets, a natural question arises: are these approaches effective to address various real-world situations?

To explore this question, in this paper, we conduct a characteristic study on comparing the synthetic datasets used by the recent research (Kushman and Barzilay 2013; Locascio et al. 2016) and real-world dataset collected from the Internet, and conduct an experimental study on applying a state-of-the-art approach (Locascio et al. 2016) on the real-world dataset. Specifically, to build the real-world dataset, we collect regular expressions and corresponding NL specifications from an online library for regular expressions. In our study, we find that there exist distinct characteristics between the synthetic datasets and real-world dataset. We evaluate the model learned by a state-of-the-art approach (Locascio et al. 2016) (from a synthetic dataset) by using both the synthetic dataset and real-world dataset. The results show that the model achieves much lower effectiveness on the real-world dataset than on the synthetic dataset. We collect some challenging cases that cannot be handled by the model, and provide initial analysis on them. To provide solutions for generating regular expressions from NL specifications, we also discuss future directions based on our study findings.

Characteristic Study

Synthetic datasets. We study two synthetic datasets:

- **KB13 (Kushman and Barzilay 2013).** KB13 includes 824 pairs of NL and regular expression. During labeling time, labeling workers are provided a certain regular expression and example strings that match the regular expression, and are asked to write their own original NL query to capture the example strings. The NL query writ-

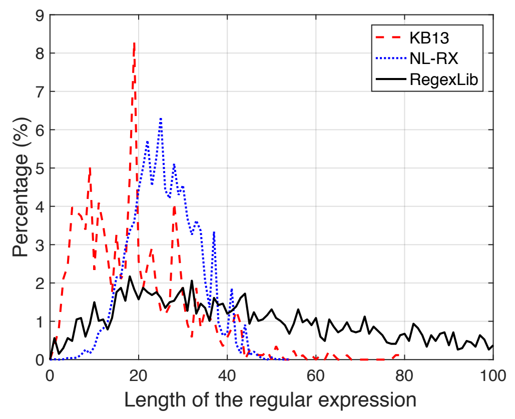

Figure 1: Distribution of regular expressions with no more than 100 characters in the three datasets (note that 26.2% of regular expressions in RegexLib have more than 100 characters whereas no regular expression in KB13 or NL-RX has more than 100 characters).

Table 1: Length statistics of regular expressions in the three datasets: maximum (Max), average (Avg), median (Med), and standard deviation (Std).

Dataset	Max	Avg	Med	Std
KB13	79	19.0	18	11.9
NL-RX	54	26.0	25	7.4
RegexLib	685	58.8	53	44.2

ten by the workers is regarded as the corresponding description of the regular expression.

- **NL-RX (Locascio et al. 2016).** NL-RX consists of 10,000 regular expressions and their corresponding NL descriptions. This corpus is created in two steps. First, a small manually-crafted grammar is used to parse a regular expression and generate its initial corresponding NL description. Second, labeling workers are asked to paraphrase the generated description.

Real-world dataset (RegexLib). We collect a real-world dataset from RegexLib (`http://regexlib.com/`), an online library for regular expressions. The library includes various regular expressions and the corresponding NL descriptions contributed by authors of the regular expressions. For our study, we use all the 3,619 pairs of NL and regular expression that can be retrieved from the search interface of the library.

By comparing the synthetic datasets and the real-world dataset, we find distinct characteristics between them in the following two main aspects.

- **Complexity of regular expressions.** Regular expressions in the real-world dataset (RegexLib) are much more complex than the two synthetic datasets (KB13 and NL-RX). First, KB13 and NL-RX support only a subset of the regular expression patterns that appear in the RegexLib. For example, the question mark '?', indicating appearing zero or one time, is not included in KB13 or NL-RX, while it is one of the most common symbols in RegexLib. Second, regular expressions in RegexLib are much longer than the ones in the two synthetic datasets. As is shown in Table 1, the average and median lengths of regular expressions in RegexLib are both more than twice of the average and median lengths in the synthetic datasets. Figure 1 shows the length distribution of regular expressions. Most of regular expressions in KB13 and NL-RX lie in the range between 10 and 40 characters, while in RegexLib more than half regular expressions contain more than 40 characters. More statistics in Table 1 show the difference in complexity of regular expressions between the synthetic datasets and the real-world dataset.

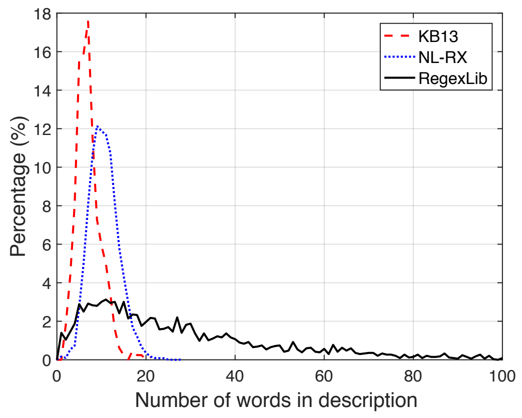

Figure 2: Distribution of NL descriptions with no more than 100 words in the three datasets (note that 9% of descriptions in RegexLib have more than 100 words whereas no description in KB13 or NL-RX has more than 100 words).

Table 2: Word count statistics of descriptions in the three datasets: maximum (Max), average (Avg), median (Med), and standard deviation (Std).

Dataset	Max	Avg	Med	Std
KB13	21	7.1	7	2.8
NL-RX	28	10.6	10	3.3
RegexLib	968	36.4	23	45.5

- **Complexity of NL sentences.** NL sentences from RegexLib contain a lot of words that never occur in KB13 or NL-RX. Specifically, there are 13,491 distinct words in RegexLib, 715 distinct words in KB13, and 560 distinct words in NL-RX. Only 219 distinct words occur in both RegexLib and KB13, and 350 distinct words occur in both RegexLib and NL-RX. Such result shows that the sentences from the synthetic datasets are much simpler than those from RegexLib. Figure 2 shows the distribution of the number of words in NL descriptions. Most of descriptions in KB13 and NL-RX have no more than 20 words, whereas the descriptions in RegexLib tend to have more than 20 words on average. More statistics in Table 2 show the difference in complexity of NL sentences be-

Table 3: Deep-Regex effectiveness on NL-RX and KB13. We reuse the result from (Locascio et al. 2016), which reports only the DFA-Equal metric for KB13.

NL-RX		**KB13**
DFA-Equal	String-Equal	DFA-Equal
58.7%	40.3%	65.6%

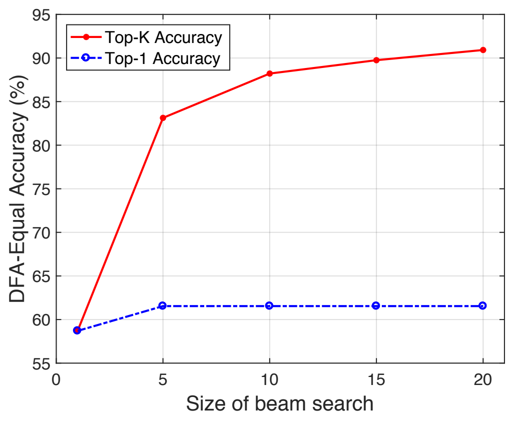

Figure 3: Effectiveness of Deep-Regex when using beam-search.

tween the synthetic datasets and the real-world dataset.

Experimental Study

We study the effectiveness of Deep-Regex (Locascio et al. 2016) (a state-of-the-art approach) on a synthetic dataset (NL-RX) and the real-world dataset (RegexLib). Deep-Regex reduces the target problem of generating regular expressions from NL specifications to a black-box task of machine translation. It uses sequence-to-sequence learning model (Sutskever, Vinyals, and Le 2014) augmented with attention mechanism (Luong, Pham, and Manning 2015) for the translation task. The sequence-to-sequence model consists of an encoder RNN and a decoder RNN. The encoder takes an NL sentence as input and generates representation vectors (for the sentence), which are further fed into the decoder. The decoder predicts next words given the words that have been predicted.

Effectiveness on synthetic datasets

We train the model on a synthetic dataset and evaluate the model on it (we reuse the split training, validation, and test data from the dataset as used by Locascio et al. (2016)). We employ String-Equal and DFA-Equal to test whether two regular expressions are equivalent. For String-Equal, two regular expressions are regarded as equivalent when they are exactly the same string. For DFA-Equal, two regular expressions are regarded as equivalent when the semantics of the regular expressions' corresponding DFAs is the same. Table 3 shows the effectiveness of Deep-Regex on NL-RX and KB13, respectively.

To understand the sources of errors and find ways to further improve the effectiveness, we randomly select 100 failed samples from NL-RX. We analyze the error types and summarize the main causes of errors as follows.

- **Ambiguity of NL.** NL descriptions in 28% failed samples are ambiguous, resulting in that the model predicts a regular expression embedding other meanings. As Example #1 shown in Table 4, the description is ambiguous because it is unclear what part of the string would appear at least twice.
- **False prediction for wildcard and quantifier.** Similar to Example #2 in Table 4, 35% of failed samples are due to that the model predicts incorrect wildcard (.) or quantifier (*, +). These symbols appear in almost every regular expression, causing the model not to be able to differentiate their semantics dependent on their positions in regular expressions.
- **False prediction for keywords.** There are 17% failed samples being predicted incorrectly by the model because the model does not make correct prediction for some keywords. Example #3 in Table 4 is a sample where the model fails to predict "and".

Other samples (20%) among the 100 failed samples cannot be easily categorized into one of the preceding categories.

To evaluate the potential capacity of Deep-Regex, we conduct an experiment that uses beam search instead of a greedy strategy. The effectiveness result is shown in Figure 3. When we use beam search of size k, we will get k candidates. Here we test (1) whether there is at least one correct result in all k candidates (denoted as Top-K) and (2) whether the candidate with the highest likelihood is correct (denoted as Top-1). The result shows that the model is more likely to generate the correct regular expression when using a larger beam size. However, finding the correct candidate among the candidates still remains another challenge.

Effectiveness on real-world dataset

We evaluate the effectiveness of Deep-Regex on the real-world dataset (RegexLib). First, we use Deep-Regex to train a model using the synthetic NL-RX dataset, named as the Deep-Regex model. Then we evaluate the model on the RegexLib dataset (as the test set). From the RegexLib dataset, we eliminate the entries with long descriptions because the inputs of Deep-Regex model have a limitation of length in our implementation. Finally, our test set contains $1,091$ pairs of NL and regular expression. The Deep-Regex model cannot generate any correct regular expressions for $1,091$ samples when using the greedy strategy. When we use beam search with size of 20 instead, the model can generate 5 samples with the DFA-Equal Top-20 metric (4 samples with the String-Equal Top-20 metric). Such extremely low effectiveness 0.5% is in sharp contrast with 90.9% (as shown in Figure 3) when applied on the NL-RX dataset. We find that many NL descriptions in the failed cases (from the RegexLib dataset) are quite different from the sentences in the training data from NL-RX, i.e., words in the test data are often not covered by the training data and are masked as unknown words. In addition to the three error-cause categories

Table 4: Example failed cases generated by Deep-Regex.

Index	**Description**	**Ground Truth**	**Predicted Result**
#1	items with a small letter preceding “dog”, at least thrice.	([a-b].*dog.*){3,}	([a-b]).*((dog){3,})
#2	lines with vowels after lower-case letter.	.*([a-z]).*([aeiou]).*	([a-z]).*([aeiou]).*
#3	lines with a lower-case letter and a character at least 6 times.	(.*[a-z].*)&((.){6,})	([a-z]).*((.){6,})
#4	match the numbers 100 to 199.	1[0-9][0-9]	([0-9])*

on the synthetic dataset (discussed in the preceding subsection), there are two types of real-world regular expressions that the Deep-Regex model trained from the synthetic NL-RX dataset cannot handle.

- **Variations of NL.** In the synthetic NL-RX dataset, NL descriptions tend not to have variations because they are first generated by a pre-defined grammar. In the RegexLib dataset, sentences with similar meanings could be very different syntactically. We find that the model trained from the synthetic NL-RX dataset cannot handle various expressions. Augmenting the training data to instill language variations may alleviate such issue.
- **Numerical range.** The Deep-Regex model cannot handle the descriptions that contain a numerical range. As shown in Example #4 in Table 4, the Deep-Regex model cannot generate a regular expression to specify a number between 100 and 199. Because the sequence-to-sequence learning model learns the translation alignment between two sequences, the Deep-Regex model may not be able to address such challenge even provided with more similar (but not the same) samples in the training data.

Future Directions

Large real-world benchmark. Ideally, we would like to retrain the Deep-Regex model (originally trained on the NL-RX dataset) on a real-world dataset such as the RegexLib one before applying the model on real-world test data entries. However, currently we can hardly train a model on the real-world dataset used in this paper because such dataset is too sparse to be a sufficient training set. Specifically, the dataset consists of $3,619$ entries from RegexLib and $13,491$ distinct words. In contrast, the NL-RX dataset is dense: it has $10,000$ entries but only 560 distinct words. Since the data collected from public websites are typically insufficient to serve as the training data, it is an open problem for collecting sufficient labeled real-world data or synthesizing additional correlated synthetic data to supplement the collected real-world data.

Testability of regular expressions. Deep-Regex treats a regular expression as a general sequence, and employs a machine translation model to address the target problem. However, a major difference between regular expressions and NL is that regular expressions are testable, i.e., the correctness of a regular expression can be tested by matched or unmatched string examples. Using string examples to further improve the generated regular expressions can be a promising direction to explore. When users want to write a regular expression to match strings, they often have some positive (matched) and negative (unmatched) string examples. Using these available string examples can help the generation process in two main ways. First, string examples can help disambiguate NL sentences (Manshadi, Gildea, and Allen 2013). Take #1 sample in Table 4 as an example, if users provide a matched string example “adogadogadog”, then the NL description would not be ambiguous, because we confirm it is “dog” that appears at least thrice from the matched example. Second, string examples can help differentiate regular expression candidates. As discussed earlier, when using beam search in Deep-Regex, a set of candidates are provided by the model. Positive and negative string examples can help select the correct (or best) answer among the candidates.

Acknowledgments

This work was supported in part by National Science Foundation under grants no. CCF-1409423, CNS-1434582, CNS-1513939, CNS-1564274.

References

Kushman, N., and Barzilay, R. 2013. Using semantic unification to generate regular expressions from natural language. In *North American Chapter of the Association for Computational Linguistics*.

Locascio, N.; Narasimhan, K.; DeLeon, E.; Kushman, N.; and Barzilay, R. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Empirical Methods on Natural Language Processing*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods on Natural Language Processing*.

Manshadi, M. H.; Gildea, D.; and Allen, J. F. 2013. Integrating programming by example and natural language programming. In *Association for the Advancement of Artificial Intelligence*.

Ranta, A. 1998. A multilingual natural-language interface to regular expressions. In *International Workshop on Finite State Methods in Natural Language Processing*.

Raza, M.; Gulwani, S.; and Milic-Frayling, N. 2015. Compositional program synthesis from natural language and examples. In *International Joint Conference on Artificial Intelligence*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Neural Information Processing Systems*.