Mathematical Expression Extraction from Unstructured Plain Text

Kulakshi Fernando(✉), Surangika Ranathunga, and Gihan Dias

Department of Computer Science and Engineering, University of Moratuwa, Katubedda 10400, Sri Lanka
{kulakshif,surangika,gihan}@cse.mrt.ac.lk

Abstract. Mathematical expressions are often found embedded inline with unstructured plain text in the web and documents. They can vary from numbers and variable names to average-level mathematical expressions. Traditional rule-based techniques for mathematical expression extraction do not scale well across a wide range of expression types, and are less robust for expressions with slight typos and lexical ambiguities. This research employs sequential, as well as deep learning classifiers to identify mathematical expressions in a given unstructured text. We compare CRF, LSTM, Bi-LSTM with word embeddings, and Bi-LSTM with word and character embeddings. These were trained with a dataset containing 102K tokens and 9K mathematical expressions. Given the relatively small dataset, the CRF model out-performed RNN models.

Keywords: Mathematical expression extraction · Sequential tagging · Information extraction

1 Introduction

Simple mathematical expressions in questions and answers of types arithmetic, linear algebra, etc can be typed inline with text. When such mathematical expressions are added in structured format to documents, for example using Tex or XML, extracting them out from the ordinary text is a trivial process. However, sometimes text containing mathematical expressions in the web are unstructured. For example, a math related content in an email, or a mathematical problem submitted into an educational forum may contain mathematical expressions embedded in plain text. '*If A and B are disjointed sets, how can you find n (A union B)?*' is such a problem submitted to the well-known educational forum, Quora.com[1]. Therefore, any system (e.g. intelligent search engines) that needs to understand the math in a document, answer generation systems for mathematical problems expressed in natural language (math word problems), or answer grading systems in mathematics should be able to recognize unstructured mathematical expressions appearing inline with text.

[1] https://www.quora.com/If-A-and-B-are-disjointed-sets-how-can%2Dyou-find-n-A-union-B.

E. Métais et al. (Eds.): NLDB 2019, LNCS 11608, pp. 312–320, 2019.
https://doi.org/10.1007/978-3-030-23281-8_26

There is much research conducted to answer and grade math word problems in elementary and secondary level mathematical domains such as arithmetic, algebra, and geometry. However, most of the time the input to these systems is expected to be annotated for mathematical expressions [9,14]. When a lay person such as a teacher or a high school student is preparing the content, this is an overhead. If user interfaces are present, these systems [3] require all mathematical expressions to be typed using tools based on Tex or XML. This is an extra effort for simple mathematical expressions that can be typed inline as other text. Such difficulties can be avoided by automatically identifying mathematical expressions from other text in math problems.

Some systems read mathematical expressions as plain text and use regular expressions (regexes) to extract them [4,15]. Fernando et al. [4] focus on automatically solving math word problems related to set theory, and show that 75% of the errors in their system occurs due to the incapability of capturing unexpected expression formats. Therefore, in the presence of unseen expression types, typing errors, and lexical ambiguities, regexes can be less accurate and require more rules to identify relevant text. Tian et al. [16] use a heuristics and vocabulary based filtering mechanism to separate expressions from other text and an Hidden Markov Model (HMM) to verify text as expressions prior to extraction. They show that HMM perform well with large amounts of data, at the expense of a long training time. Therefore, it is worth exploring alternative techniques; especially techniques that do not require predefined rules or states, to extract mathematical expressions.

This research presents a mechanism to extract simple mathematical expressions that appear in-line with other non-mathematical (natural language) text[2]. The task of extracting mathematical expressions is mapped into a sequence tagging task, where space-separated tokens are tagged as expressions and other text using IOB (Inside-Outside-Beginning) format. Experiments were conducted using Conditional Random Fields (CRF), Long-Short Term Memory (LSTM) networks with word embeddings (we refer to this setting as W-LSTM hereafter), Bidirectional-LSTM (Bi-LSTM) with word embeddings (we refer this as W-Bi-LSTM), and Bi-LSTM with both word and character embeddings (we refer to this model as W-CH-Bi-LSTM). Experiments with CRF showed that character level properties of the text contribute noticeably to increasing the accuracy of expression extraction.

2 Challenges in Extracting Mathematical Expressions in Plain Text

The main challenge in identifying mathematical expressions from other text is semantic level ambiguities between expressions and non-expression text. One of the main semantic level ambiguities is variable names used in mathematical expressions. For example, the token *'a'* in *'Let a be a positive integer'* is both

[2] The code and data is available here https://github.com/Kulakshi/math-expression-extraction.

a mathematical expression and a stop word. Another ambiguity is the use of a sequence of numbers in contrast to a list of some numbers. For example in the text *"In the sequence 7, 14, 28, x, 112.. what is the value of x?"*, the mathematical expression is the sequence:*' 7, 14, 28, x, 112..'* whereas in *"Find arithmetic mean of the numbers in the list 8 − a, 8, 8 + a"*, mathematical expressions are individual expressions separated by commas.

Text copied from examination papers or tutorials often include question numbers as digits, roman numbers or letters that can be misinterpreted as mathematical expressions. Years, table or figure labels, and abbreviated names for irrelevant entities are few other occasions that can be misinterpreted as mathematical expressions.

When tokenizing text combined with mathematical expressions, they may get split into different combinations when spaces are considered as the delimiter. For example, if the expression is *'x + 1'*, we get three tokens. If the same is written as *'x + 1'* we get only two tokens. Another important fact is the start and end of a mathematical expression. An expression that contains words such as *'A = total area of five circles of radius r'*, or an expression with typos like *'set A = {Students of grade five and set B = {Girls in grade five}'* do not have a clear lexical separation from the usual text.

3 Related Work

Fernando et al. [4] extracted set theory related mathematical expressions from the unstructured plain text using regexes. However, they claim that regexes fail when unexpected expression formats are met and it is the main reason for reducing the accuracy of the system. Seo et al. [15] also extracted geometry related mathematical expressions using regexes.

Work of Tian et al. [16] is the most relevant research we could find that focused on extracting mathematical expressions in general from unstructured plain text. They eliminate other text in a document using heuristics and vocabulary based filtering. An HMM with 8 hidden states denoting parts of mathematical expressions is then used to verify filtered expressions before extraction. They mapped mathematical symbols that are observable in the text into these hidden states. They trained the model with 13,423 expressions and achieved over 89% accuracy and 77% recall. However, the dataset is not available to be used to compare their results empirically.

HMMs, CRFs [10], Convolution networks and Recurrent Neural Networks (RNNs) are well-known for sequence classification for decades. LSTM networks [7] reduce the vanishing gradient problem present in standard RNNs and perform well for long sequences of data. Bi-LSTM increases the accuracy of sequence tagging by considering both past and future inputs as used in the work of Graves et al. [6]. There is plenty of research [2,8] that successfully used LSTM networks with different varieties and combinations for the sequential classification task. Many research including the work of Ling et al. [12] shows that using character level information with LSTM networks is effective in increasing the performance

of language modeling. In the presence of well defined features, Nikola [13] shows that CRF can perform as well as Bi-LSTM models in a sequential classification task.

4 Methodology

4.1 Dataset and Pre-processing

An adequately annotated dataset for mathematical expressions was not available to use in our research. Thus we adapted the dataset provided by Task 10 of SemEval-2019, "Math Question Answering"[3] and the dataset of Fernando et al. [4]. The former dataset includes mathematical problems that belong to closed-vocabulary algebra, open-vocabulary algebra, geometry, probability, and data representation. Some questions that belong to multiple domains are not categorized. Most of the mathematical expressions in this dataset were given using LaTeX. Such expressions were converted into inline mathematical expressions. For example, '8\times(2 ∧ 4a) = \frac{2 ∧ 2b}{2 ∧ 3}' was converted into $8 \times (2 \wedge 4a) = 2 \wedge 2b/2 \wedge 3$. For now, mathematical expressions that cannot be written along the mean-line; text that includes special scripts such as superscripts and subscripts are not handled in this work. The dataset of Fernando et al. [4] consists of elementary set theory problems that include both text and expressions in set notation. Both the datasets were adapted for this research by tokenizing text and annotating tokens as described next.

Text with inline expressions was then tokenized and tagged in IOB format for expressions and other texts to prepare the dataset. For example, the problem *If* $a - 5 = 0$, *what is the value of* $a + 5$*?* is tagged as *O B-EXP I-EXP I-EXP I-EXP I-EXP O O O O O B-EXP I-EXP I-EXP O*.

Dataset statistics are shown in Table 1 ('elementary set theory' category contains problems of the dataset of Fernando et al. [4]).

Table 1. Statistics of problems in the dataset

Category	#problems	#expressions	#tokens	#expression tokens
Closed-algebra	1088	3832	29541	10886
Open-algebra	360	1059	16332	1372
Geometry	702	2351	22677	4288
Other	86	124	3634	156
Uncatagorized	528	1717	22796	3219
Elementary set theory	487	2419	31380	16308

Cetintas et al. [1] show that traditional text pre-processing tasks such as stemming and stop word removal affect negatively in mathematical text categorization tasks since math related information are lost with such pre-processing

[3] https://github.com/allenai/semeval-2019-task-10.

approaches. For example, words such as *'a', 'than'* and words with suffixes such as *'rd', 'th'* are important to identify math related text. This is relevant to mathematical expression extraction as well, given the ambiguities (see Sect. 2). Therefore we avoided such text pre-processing methods.

Some expressions in the dataset had typos. Specially equations converted from LaTeX to plain text included syntactical errors. They were kept intact since the purpose of this research includes identifying expressions that might contain errors.

4.2 Experiment Setup

All models were trained using 10-fold cross validation. The CRF model was trained by using unigrams as the baseline feature. Other features in Table 2 were added incrementally to evaluate the effectiveness of each feature set. Features that contribute effectively to the model were selected using the validation dataset. Some features (shown in non-italic letters in Table 2) were adapted from the work of Finkel et al. [5] and Huang et al. [8]. These features were originally used for normal NER tagging. A few more features were added that seems relevant to mathematical expressions. These features are distinguished from aforementioned features by the italic font in Table 2).

Table 2. Features used for CRF divided into categories. (The left most column contains a label for each set of features)

Context features	
A	Uni-grams, bi-grams, tri-grams and their frequencies
Token level features	
B	*Whether the token is a single character*
C	Case related features (all upper case, all lower case, starts with capital, contains non-initial capital letters)
D	Features related to character type (*contains only digits*, *contains math operators*, *contains bracket delimiters*, contains only letters, is a mix of digits and letters, contains punctuation marks, word shape, word shape summarization [8])
E	Last two and last three suffixes
Semantic level features	
F	POS tag of the token and surrounding tokens

This being the first research conducted for expression extraction using RNNs, we used vanilla LSTM and Bi-LSTM model architectures for first two settings (W-LSTM and W-Bi-LSTM), which comprised an word embedding layer where the 50-length output is subjected to a 10% dropout to avoid overfitting, one LSTM layer and an output layer with softmax normalization. The third setting

(W-CH-Bi-LSTM) comprises of an additional embedding layer with an encoder for characters, concatenated with word embeddings [11]. The model architecture is shown in Fig. 1. Prior to the training, we experimented with different optimizers (standard SGD, Adam and RMSProp), dropout rates (10%, 20%, 50%), different batch sizes (10, 32, 50 & 100) and epochs (10, 20, 50, 100, 500 and 1000), and selected batch size of 10, 10% dropout and RMSProp optimizer with 0.001 learning rate to train the models.

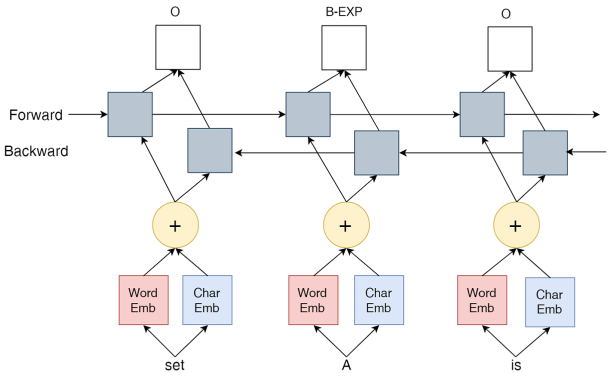

Fig. 1. High level architecture of W-CH-Bi-LSTM model. Source: [8,11]

5 Evaluation and Results

We need to calculate the accuracy of extracting complete mathematical expressions. Given the set of expected expressions E, and the set of predicted expressions P, true-positives (TP), false-positives (FP) and false-negatives (FN) were calculated for each model as follows.

$$TP = \{Expressions\ in\ both\ E\ and\ P\ \}$$
$$FP = \{Expressions\ in\ P\ but\ not\ in\ and\ E\ \}$$
$$FN = \{Expressions\ in\ E\ but\ not\ in\ P\ \}$$

Figure 2 shows the results of the CRF model for cumulatively added features. When considering the CRF model, case related features and character-based features such as whether there are digits in the token, whether only letters contain in the token, and the suffix of the token increased the performance of expression extraction in a significant rate.

Table 3 shows the comparison of the best results of all the models. The W-Bi-LSTM model performs better than other RNN models. Since the dataset is small-sized, the CRF model performs better than RNN models.

When analysing errors, the words such as 'Mickey-Mouse' had been predicted as expressions by RNN models due to non alphanumeric symbols. In addition,

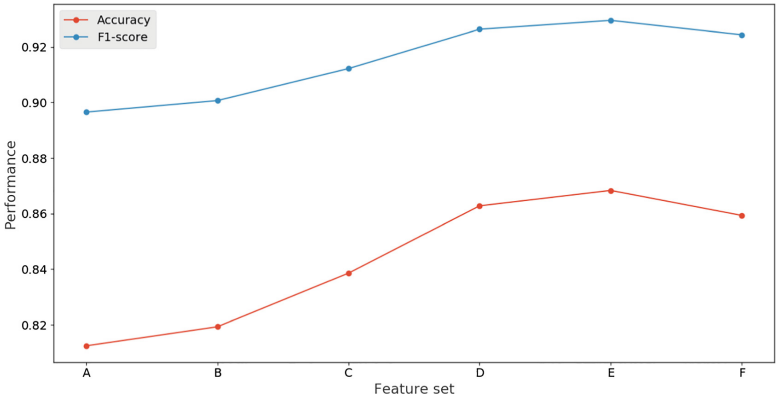

Fig. 2. CRF performance against cumulatively added feature sets from the set A to set G listed in Table 2

Table 3. Accuracy, Recall, Precision and F1-score of best performance of all models

Model	Acc.	Recall	Prec.	F1
CRF	**0.875**	**0.937**	**0.929**	**0.933**
W-LSTM	0.805	0.914	0.870	0.892
W-Bi-LSTM	0.824	0.917	0.890	0.903
W-CH-Bi-LSTM	0.811	0.916	0.875	0.895

the models exhibited poor performance in differentiating related numbers to solve the problem and unrelated numbers such as years in the text. The errors suggest that more semantic level features such as POS could help in increasing the accuracy.

6 Conclusion

In this research, the problem of extracting mathematical expressions from the unstructured plain text was modeled as a sequential text classification problem, an empirical evaluation was carried out on the state-of-the-art classifiers and a manually annotated dataset suitable to identify mathematical expressions in the text is presented. Given the dataset is small-sized, the achieved results are justifiable. While CRF performed the best, Bi-LSTM networks performed better than LSTM network.

LSTM with a CRF in the output layer helps to increase the accuracy of a sequential tagging task since it considers the possible transitions between output labels [8]. We hope to experiment with this in the future. After extracting mathematical expressions, it is useful to identify syntactically correct expressions separately. We plan to develop a rule-based parser to recognize any errors

in filtered expressions, which can be used to give feedback to the user in end-user applications such automatic math problem solvers and e-learning systems.

Acknowledgment. This research was funded by a Senate Research Committee (SRC) Grant of the University of Moratuwa and LK Domain Registry.

References

1. Cetintas, S., Si, L., Xin, Y.P., Zhang, D., Park, J.Y.: Automatic text categorization of mathematical word problems. In: FLAIRS Conference (2009)
2. Chen, T., Xu, R., He, Y., Wang, X.: Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert. Syst. Appl. **72**, 221–230 (2017)
3. Erabadda, B., Ranathunga, S., Dias, G.: Automatic identification of errors in multi-step answers to algebra questions. In: 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), pp. 215–219. IEEE (2017)
4. Fernando, K., Ranathunga, S., Dias, G.: Answer generation for set type math word problems. In: Proceedings of the 2018 International Conference of Advances in ICT for Emerging Regions (2018)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 363–370. Association for Computational Linguistics (2005)
6. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6645–6649. IEEE (2013)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
9. Kadupitiya, J., Ranathunga, S., Dias, G.: Automated assessment of multi-step answers for mathematical word problems. In: 2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 66–71. IEEE (2016)
10. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
11. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
12. Ling, W., et al.: Finding function in form: compositional character models for open vocabulary word representation. arXiv preprint arXiv:1508.02096 (2015)
13. Ljubešić, N.: Comparing CRF and LSTM performance on the task of morphosyntactic tagging of non-standard varieties of South Slavic languages. In: Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018), pp. 156–163 (2018)
14. Matsuzaki, T., Ito, T., Iwane, H., Anai, H., Arai, N.H.: Semantic parsing of pre-university math problems. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), vol. 1, pp. 2131–2141 (2017)

15. Seo, M., Hajishirzi, H., Farhadi, A., Etzioni, O., Malcolm, C.: Solving geometry problems: Combining text and diagram interpretation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1466–1476 (2015)
16. Tian, X., Bai, R., Yang, F., Bai, J., Li, X.: Mathematical expression extraction in text fields of documents based on HMM. J. Comput. Commun. **5**(14), 1 (2017)