CISC 367 Detecting Un-Punctuated Questions

Down by the C++

Jackson Burns
University of Delaware
jburnsky@udel.edu

Kristina Holsapple
University of Delaware
kris@udel.edu

Michael Carr
University of Delaware
mjcarr@udel.edu

ABSTRACT

The following pages contain the research project for Introduction to Computer Science Research (CISC 367) for the group Down by the C++, composed of Jackson Burns, Kristina Holsapple, and Michael Carr. This project aims to detect un-punctuated questions in Slack chat messages through the design and implementation of a simple and efficient regular expression-based method for the identification of punctuated questions in data-mining operations. Results suggest that the question detection algorithm cannot replace higher-order question detection altogether, but can serve as an efficient initial filter. The work proposed herein should be generally applicable to data mining operations involving the identification of questions.

CCS CONCEPTS

• **Software and its engineering** → *Software design engineering*;
• **Applied computing** → **Collaborative learning**;

KEYWORDS

Slack, data mining, regular expressions

ACM Reference Format:
Jackson Burns, Kristina Holsapple, and Michael Carr. 2020. CISC 367 Detecting Un-Punctuated Questions: Down by the C++. In *Proceedings of ACM Newark conference (NEWARK '20),* CISC 367 Peers and Dr. Lori L. Pollock (Eds.). ACM, New York, NY, USA, Article 0, 3 pages. https://doi.org/none

1 INTRODUCTION

The mining of various Question and Answer (Q&A) forums has been a topic of increasing interest in recent years, and for good reason. There is a wealth of knowledge in these locations, containing everything from code snippets to debugging assistance to simple preferences between APIs [1]. This collected insight totals thousands of hours of input from devoted programmers, and this knowledge is invaluable for other developers and students. Paramount to being able to properly identify when an experienced user is offering advice, is being able to identify when a question is asked [13]. This seems like a remarkably simple task on the face of it, though it has been estimated that approximately thirty percent of questions in some Q&A forums are un-punctuated [1] which has historically ruled out simple regular expression approaches. While there have been a number of other approaches to resolving this issue, each relies on complex and computationally expensive algorithms [2][3] [12] [11] [8]. Some solutions have been attempted in classifying asynchronous and synchronous communication platforms [9] and in optimizing Dialogue Act Classification [6], no work has been applied specifically to this large repository of developer input, Slack. We propose that this issue in identify un-punctuated questions can be resolved by applying this work in Dialogue Act Classification with some well-designed regular expression logic to achieve similar or better results to higher-order methods while requiring significantly less computational hardware. The solution will be evaluated using a combination of common statistical measures, such as the Cohen's Kappa, the F-Score, and the recall and precision scores. By applying this solution to number of different data sets, it can be ensured to be applicable to a variety of problems.

NEWARK '20, October 2020, El Newark, Delaware USA
2020. ACM ISBN none…$0.00
https://doi.org/none

2 METHODOLOGY

In order to properly design a system to detect un-punctuated questions using regular expressions, our team needed a set of testing data which has been pre-parsed by a human being in order to ensure that the detection is accurate. This will henceforth be referred to as the Gold Sets [4] and was created my Michael Carr, who is studying English and was able to work through the messages with the highest degree of accuracy. Fine-tuning the algorithm results from this work will require the creation of an automated testing bed that can rapidly analyze the results and supply useful statistical measures of the accuracy. This was created by Kristina Holsapple and required the implementation of a number of Python packages. Scikit-learn [7] was used for the calculation of many of these common parameters, such as the Cohen's Kappa and the F-Score.
The main group of functions which are responsible for the parsing of arrays of text using regular expressions and the breaking of sentences into their components were written by Jackson Burns, again in Python. To break sentences into their parts of speech, the Natural Language ToolKit (NLTK) [5] was used. This package has a function which enables the user to break a string of text into its parts of speech. By subjecting slack messages to this feature, it can be easily deduced if they contain interrogative speech. This process was then automated using regular expressions which are available by default in Python [10] via its internal libraries. These functions, the gold sets, and the testing apparatus are all called by a single central script which outputs the formatted results for further interpretation by the group.

2.1 Evaluation Question

The goal, as explained above, was to design and implement a lower-order regular expression method to evaluate the number of questions present in a data set. This would hopefully reduce execution time and complexity but must maintain similar or better accuracy in order to be worthwhile.

2.2 Evaluation Measures

The success or failure of the algorithm was quantified using the Cohen's Kappa and the F-score, both of which have been previously implemented in Python [7]. By comparing these metrics to those reported for other algorithms in the literature, a comparison was made between the two approaches. Our aim was to, within a given confidence interval, have the regular expression method is able to achieve similar or better performance.

2.3 Gold Set Creation Procedure

The Gold Set was created according to the method proposed by Juckett [4]. The total size of the gold set was determined by the frequency of the desired descriptor, in this case questions, that are in the data set. First a small subset will be chosen as an estimator from which the frequency can be determined, and the Gold Set will be produced from there.

2.4 Evaluation Procedure

The Gold Set and automatically retrieved result were sent by a central script to a test bed, which was responsible for the calculation of the various metrics mentioned above. Using the results, comparisons were drawn to existing solutions and the algorithm was tuned accordingly.

3 RELATED WORK

As mentioned previously, this problem has been approached by a number of developers in the past. All solutions which were conceived rely on highly complex algorithms which require expensive hardware and long computing times. The Duan group was able to achieve accurate results via the implementation of a vector space model, which is a complex multi-dimensional abstraction away from the source text[2]. Another approach by the Jeon group relied on the use of an existing large data set of identified questions to use as a point of reference for a comparison based algorithm [3]. This solution is comparatively lower-order, but creates a chicken-or-the-egg issue. Implementing this method requires an existing large data set, which can be difficult to attain without automation. A number of groups have attempted to implement translation and syntactic tree matching models[12] [11] [8]. While these methods are robust against user grammatical errors and reach high accuracy's, both require a detailed understanding of the linear algebra taking place behind the scenes in order to make any meaningful changes to the approach. These methods are therefore both slower than simple regular expressions and become a black-box for researchers who do not have a strong enough math background, preventing improvement.
Some additional work has been done in the past on how to properly classify messages on both synchronous and asynchronous platforms [9], which will be helpful in attempting to analyze messages from the exclusively synchronous Slack platform. Work has also been done in optimizing Dialogue Act Classification [6], which will be the core aspect of the functions described in the proposed work. The intersection of these two fields is where this new potential solution resides.
Data sets used in mining continue to grow as developers and newcomers continue to turn to these services for advice. Slow methods will become increasingly impractical as this happens and as algorithms grow more complicated, it increases the barrier to entry for new researchers.

4 RESULTS

With the algorithm implemented in Python and the required dependencies installed, the main script was executed on a local, low-power laptop computer. The results, including the implications and the limitations, are explored below.

4.1 Result Metrics

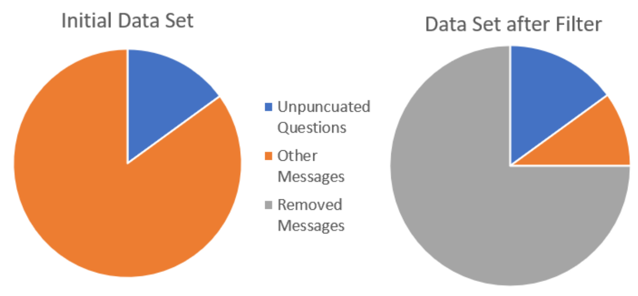
The results of the algorithm are quantified in the table below using common statistical metrics.

Data Set	F-Score	Cohen's Kappa	Execution Time (s.)
Python 2017	0.12	0.01	28
Python 2018	0.21	0.08	60
Python 2019	0.27	0.09	31
Kaggle	0.2	-0.01	3

Execution time was on the order of seconds for all but the largest data set, which is many orders of magnitude faster than the machine learning methods discussed above. Unfortunately, F-score and Cohen's Kappa are low, indicating poor agreement with the gold sets.

4.2 Implications

The low values for the measured metrics indicate that the tool is not successful in identifying unpuncuated questions and should not completely replace existing solutions which implement higher-order machine learning approaches. This is not to say, however, that the tool was wholly unsuccessful. As expected, execution times are negligible and achievable even with low-power local machines. Therefore, implementing this tool as an add-on rather than a replacement could be of some merit.
Line-by-line analysis of the gold set and the automated set reveal that the algorithm is returning a substantial number of false positives but very few false negatives. With this in mind, the algorithm could be used as an initial filter for a more complicated algorithm, decreasing the required computational time, as shown in the graphic below.



4.3 Limitations

The suspected cause of the above issues is the presence of code snippets inline with text. This code is difficult to distinguish with simple methods, as various messages will have non-uniform lengths and placements of code. Additionally, the creation of gold sets large enough to adequately represent these large data sets is tedious and slow work. Finally, the chat channels which are analyzed are a casual environment where English conventions are often ignored. These data sets are definitely worth analyzing due to the volumes of worthwhile input they contain, but detailed must be performed on a question-by-question basis to account for use of slang, acronyms, etc.

5 THREATS TO VALIDITY

5.1 External Threats

The majority of data collected to be used in this experiment was collected a single slack channel. However, this was mitigated by analyzing a data set from a separate source which produced similar results to the other data set.

5.2 Internal Threats

Due to the relatively short timeline of this project, the gold set was created by a member of our team. This potential issue was minimized by limiting his role in the design of the retrieval apparatus to debugging help only when necessary.

6 CONCLUSION

Within the timeline of two and half weeks, our group has created a portable Python environment with a software tool capable of detecting unpuncuated questions with a reasonable rate of accuracy given the simplicity of the methods. Any future user of the work can visit the GitHub repository, which includes documentation and UML diagram to familiarize themselves with the code. We firmly believe that, given further development, the performance of the algorithm could be substantially improved. In its current state, there is a substantial number of false positives (i.e., non-questions being identified as questions) in each of the experimental data sets. This can be attributed to difficulty in separating inline code snippets from text, and from the lack of adherence to English conventions in the informal setting of chat rooms. There are, however, valid use cases for the tool. Because the algorithm can successfully filter out the majority of non-questions, it can be used to reduce the total number of messages that a more complicated algorithm would need to classify. This could potentially save a substantial amount of execution time, ultimately achieving the established goal of the project in an indirect manner.

7 WORK ATTRIBUTIONS

Jackson Burns

- UML Diagram of Repository
- RegEx parser
- Grammar Parser
- Main Script
- Results and Conclusion Sections

Michael Carr

- Data Set Collection
- Gold Set Creation
- GitHub ReadMe
- Methodology and Threat to Validity

Kristina Holsapple

- Testbed Scripts
- Abstract

ACKNOWLEDGMENTS

The authors would like to thank Dr. Lori Pollock for providing guidance on the use of Overleaf and approaches to this project.

REFERENCES

[1] Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding Question-Answer Pairs from Online Forums. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 467–474. https://doi.org/10.1145/1390334.1390415

[2] Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching Questions by Identifying Question Topic and Question Focus. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, 156–164. https://www.aclweb.org/anthology/P08-1019

[3] Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM '05)*. Association for Computing Machinery, New York, NY, USA, 84–90. https://doi.org/10.1145/1099554.1099572

[4] David Juckett. 2012. A method for determining the number of documents needed for a gold standard corpus. *Journal of biomedical informatics* 45 3 (2012), 460–70.

[5] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

[6] James D. O'Shea, Zuhair A. Bandar, and Keeley A. Crockett. 2013. Optimizing Features for Dialogue Act Classification. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC '13)*. IEEE Computer Society, USA, 474–479. https://doi.org/10.1109/SMC.2013.87

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[8] Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, 464–471. https://www.aclweb.org/anthology/P07-1059

[9] Maryam Tavafi, Yashar Mehdad, Shafiq Joty, and Raymond Ng. [n.d.]. Dialogue Act Recognition in Synchronous and Asynchronous Conversations.

[10] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[11] Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009. A Syntactic Tree Matching Approach to Finding Similar Questions in Community-Based Qa Services. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 187–194. https://doi.org/10.1145/1571941.1571975

[12] Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval Models for Question and Answer Archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. Association for Computing Machinery, New York, NY, USA, 475–482. https://doi.org/10.1145/1390334.1390416

[13] Hui Yang and Tat-Seng Chua. 2004. Web-Based List Question Answering. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*. Association for Computational Linguistics, USA, 1277–es. https://doi.org/10.3115/1220355.1220542