Snakes in Paradise?: Insecure Python-Related Coding Practices in Stack Overflow

Conference Paper · March 2019

DOI: 10.1109/MSR.2019.00040

3 authors, including:

Nasif Imtiaz
North Carolina State University
8 PUBLICATIONS **43** CITATIONS

SEE PROFILE

Snakes in Paradise?: Insecure Python-related Coding Practices in Stack Overflow

Akond Rahman, Effat Farhana, and Nasif Imtiaz
North Carolina State University, Raleigh, North Carolina
Email: aarahman@ncsu.edu, efarhan@ncsu.edu, simtiaz@ncsu.edu

***Abstract*—Despite being the most popular question and answer website for software developers, answers posted on Stack Overflow (SO) are susceptible to contain Python-related insecure coding practices. A systematic analysis on how frequently insecure coding practices appear in SO answers can help the SO community assess the prevalence of insecure Python code blocks in SO. An insecure coding practice is recurrent use of insecure coding patterns in Python. We conduct an empirical study using 529,054 code blocks collected from Python-related 44,966 answers posted on SO. We observe 7.1% of the 44,966 Python-related answers to include at least one insecure coding practice. The most frequently occurring insecure coding practice is code injection. We observe 9.8% of the 7,444 accepted answers to include at least one insecure code block. We also find user reputation not to relate with the presence of insecure code blocks, suggesting that both high and low-reputed users are likely to introduce insecure code blocks.**

***Index Terms*—python, reputation, security, stack overflow**

I. Introduction

Stack Overflow (SO) is regarded as the most popular question and answer website for software developers [1]. How software developers use and contribute to SO posts for software projects [2] [3] [4], and the technologies and tools [5] [6] they refer to, has drawn interest of researchers.

Despite SO's popularity, answers posted on SO are susceptible to contain code blocks which include insecure coding practices. Let us consider Figure 1 as an example in this regard. In Figure 1 we present an accepted answer posted on SO [7] related to hash calculation in Python. The answer proposes the use of MD5 to compute hash using the 'hashlib.md5()' method. Use of MD5 is insecure, as MD5 is susceptible to collision attacks. The National Institute of Standard and Technologies (NIST) recommends against the use of MD5 [8]. A software developer who wants to implement hash in Python and comes across this answer may use the provided code block, unintentionally leaving a security weakness in the software code. We take motivation from the provided code block in Figure 1, and investigate the prevalence of insecure coding practices in SO answers related to Python, which IEEE Spectrum ranked as the top-most programming language for the year 2018 [9].

We conduct our empirical study to systematically quantify the prevalence of insecure coding practices i.e. recurrent use of insecure coding patterns in Python. We answer the following research questions:

- **RQ1**: How frequently do insecure coding practices appear in Python-related Stack Overflow answers?

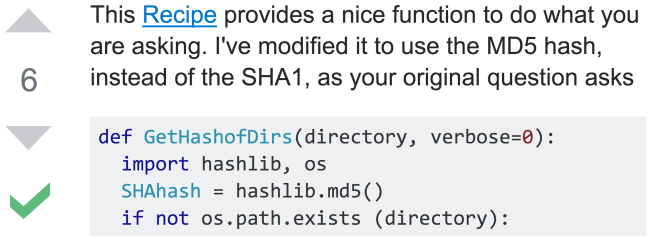

Fig. 1: An accepted answer posted on SO, which uses MD5 to implement hashing in Python.

- **RQ2**: How does user reputation relate with the frequency of insecure Python-related coding practices?
- **RQ3**: What are the characteristics of Python-related questions that include answers with insecure code practices?

We conduct an empirical study by using the MSR2019 Mining Challenge dataset [1]. By systematically applying a set of filtering criteria we identify 44,966 SO answers related to Python, which include 529,054 code blocks in total. Next, we identify the frequency of insecure coding practices. We also quantify the relationship between the presence of insecure coding practices and user reputation by applying statistical analysis. Finally, we apply topic modeling [10] to characterize Python-related SO questions that include answers with insecure coding practices.

Our contribution is an empirical study that characterizes the prevalence of insecure coding practices in Python-related SO answers.

II. Empirical Study

We use the MSR2019 Mining Challenge Dataset provided by Baltes et al. [1]. We summarize the steps to conduct our empirical analysis in Figure 2. Before describing the details of our empirical study we present necessary definitions below:

Insecure coding practice: Recurrent use of any insecure coding patterns listed in Table II.

Insecure code block: A code block in an SO answer that includes any of the insecure coding patterns described in Table II.

Insecure answer: An answer which includes one or multiple insecure code blocks.

Insecure accepted answer: An accepted answer which includes one or multiple insecure code blocks.

TABLE I: Selection of Stack Overflow Posts for Analysis

Initial post count	41,782,536
Criteria-1 (Ques. with at least one answer)	14,207,037
Criteria-2 (Ques. with score $>$ 0)	6,902,332
Criteria-3 (Ques. with $>$ 0 views)	6,902,332
Criteria-4 (Ques. linked to $>$ 0 '.py' files in GitHub)	10,861
Final question count	10,861

Neutral answer: An answer which includes no insecure code blocks.

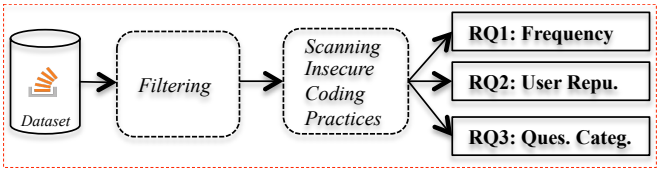

Fig. 2: Methodology to conduct our empirical study.

A. Filtering

Before applying our analysis we filter the provided dataset based on a set of criteria summarized in Table I. We obtain 10,861 questions, which included 44,966 answers. The 44,966 answers included 529,054 Python code blocks. Of the 44,966 answers, 7,444 are accepted answers i.e. answers that are accepted by the question providers. Our selection criteria is inspired by Meng et al. [11]. We use answer count, views, and scores to filter out irrelevant SO questions for analysis. We added Criteria-5, to identify questions that may actually be in use, as evident from GitHub history. Constructed datasets and analysis scripts are available online [12].

B. Scanning Code

To identify insecure coding practices in these code blocks, we use a catalog of insecure coding practices reported by Openstack [13]. The list reported by Openstack included 77 coding patterns of Python that have security weakness. These 77 coding patterns have commonalities with respect to security weakness. For example, both 'Crypto.Hash.MD2.new' and 'Crypto.Hash.MD4.new' both refer to the same insecure coding practice: relying on two Python functions that use weak cryptography algorithms. We apply a categorization scheme so that the common security weaknesses can be grouped together by using two raters who are PhD students, with an average experience of three years in software security. The two raters independently categorized the 77 insecure coding patterns. The first and second rater respectively identified seven and five categories of insecure coding practices, of which five were common. We record a Cohen's Kappa [14] of 0.5. The disagreements are resolved by the first author. Upon resolving disagreements we finally identify a set of six insecure coding practices:

Code injection is the practice of using coding patterns that are susceptible to arbitrary code or command injection. For example, Python's 'eval' function is susceptible to executing harmful commands without validation [15].

Cross-site scripting (XSS) is the practice of using coding patterns such as the 'django.utils.safestring.mark_safe' function [16] that enables injection of client-side scripts into web pages.

Insecure cipher is the practice of using weak cryptography algorithms such as MD2 and MD5 [8] or predictable random number generators such as Python's random.random() function.

Insecure connection is the practice of using coding patterns such as 'httplib.HTTPSConnection' that uses the hypertext transfer protocol (HTTP) and the file transfer protocol (FTP) to create and open connections [17].

Race condition is the practice of using coding patterns such as 'mktemp' that enables creation of temporary files with predictable paths, increasing the possibility of time of check, time of use attacks [18].

Untrusted data serialization is the practice of using data serialization-related coding patterns such as pickle.loads() without validating the authentication of the source [19].

A complete mapping of each insecure coding practice with the corresponding Python coding pattern is available in Table II. We detect the presence of an insecure coding practice if any of the coding patterns listed in column 'Coding Pattern' is present. We detect the presence using string matching.

Sanity checking: Use of string matching can yield false positives and false negatives. We mitigate this limitation by randomly selecting 100 SO answers with code blocks that have been identified to contain insecure code blocks. From our manual inspection we observe no false positives and false negatives.

C. Answer to RQ1: How frequently do insecure coding practices appear in Python-related Stack Overflow answers?

Approach: For each category of insecure coding practice, we answer RQ1 by reporting count of insecure code blocks, count of insecure answers, count of insecure accepted answers and count of insecure questions.

Findings: We report our detailed findings in Table III, where each insecure coding practice is reported as columns, and each frequency measure is reported as rows. For example, we identify 2,319 code blocks for which code injection-related coding patterns appeared. The 'Combined' column presents the frequency when all insecure practices are considered. We observe 0.69% of the 529,054 code blocks to include at least one of the six insecure coding practices, as shown in the 'Combined' column. We observe 18.1% of the 10,861 SO questions to contain at least one insecure answer. We observe 9.8% of the 7,444 accepted answers to include at least one insecure code blocks.

D. Answer to RQ2: How does user reputation relate with the frequency of insecure Python-related coding practices?

We hypothesize that SO users who have less reputation are more likely to introduce insecure answers for SO questions.

Approach: We evaluate our hypothesis by calculating reputation of users who provide answers that include at least

TABLE II: Insecure Coding Practice and Corresponding Coding Pattern

Insecure Practice	Corresponding Coding Pattern
Code injection	input,eval
Cross-site scripting (XSS)	django.utils.safestring.mark_safe
Insecure Cipher	hashlib.md5,cryptography.hazmat.primitives.hashes.MD5,Crypto.Hash.MD2.new, Crypto.Hash.MD4.new,Crypto.Hash.MD5.new,Crypto.Cipher.ARC2.new,Crypto.Cipher.ARC4.new, Crypto.Cipher.Blowfish.new,Crypto.Cipher.DES.new,Crypto.Cipher.XOR.new,cryptography.hazmat.primitives.ciphers.algorithms.ARC4', cryptography.hazmat.primitives.ciphers.algorithms.Blowfish, cryptography.hazmat.primitives.ciphers.algorithms.IDEA', cryptography.hazmat.primitives.ciphers.modes.ECB,random.random,random.randrange,random.randint,random.choice', random.uniform,random.triangular
Insecure connection	httplib.HTTPSConnection,http.client.HTTPSConnection, six.moves.http_client.HTTPSConnection,telnetlib.*,urllib.urlopen,urllib.request.urlopen,urllib.urlretrieve, urllib.request.urlretrieve,urllib.URLopener,urllib.request.URLopener,urllib.FancyURLopener, urllib.request.FancyURLopener,urllib2.urlopen,urllib2.Request,six.moves.urllib.request.urlopen, six.moves.urllib.request.urlretrieve,six.moves.urllib.request.URLopener, six.moves.urllib.request.FancyURLopener,urlopen,urlretrieve,ftplib.*
Race condition	mktemp,tempfile.mktemp
Un-trusted data serialization	pickle.loads, pickle.load, pickle.Unpickler, cPickle.loads, cPickle.load, cPickle.Unpickler, marshal.loads, marshal.load, xml.etree.cElementTree.parse, xml.etree.cElementTree.iterparse, xml.etree.cElementTree.fromstring, xml.etree.cElementTree.XMLParser, xml.etree.ElementTree.parse, xml.etree.ElementTree.iterparse, xml.etree.ElementTree.fromstring, xml.etree.ElementTree.XMLParser, xml.sax.expatreader.create_parser, xml.dom.expatbuilder.parse, xml.dom.expatbuilder.parseString, xml.sax.parse, xml.sax.parseString, xml.sax.make_parser, xml.dom.minidom.parse, xml.dom.minidom.parseString, xml.dom.pulldom.parse, xml.dom.pulldom.parseString, lxml.etree.parse, lxml.etree.fromstring, lxml.etree.RestrictedElement, xml.etree.GlobalParserTLS, lxml.etree.getDefaultParser, lxml.etree.check_docinfo

TABLE III: Answer to RQ1: Frequency of Insecure Python Code Blocks in Stack Overflow Answers

	Code Inj.	XSS	Ins. Cip.	Ins. Conn.	Race. Cond.	Data Serial.	Combined
Code block	2,319	0	564	624	25	153	3,685
Answers	2,263	0	529	311	19	101	3,223
Accepted answers	481	0	130	85	2	33	731
Questions	1,415	0	317	155	14	73	1,974

one insecure coding practice, and reputation of SO users who provide answers that do not include any insecure code block. To account the fact the a SO user who has been a long-time member may have high reputation, we normalize the a user's reputation by the membership period measured in months. We calculate user reputation ($NORM\_USER\_REPU$) using Equation 1:

We use the Mann-Whitney U test [20] to compare two distributions: $NORM\_USER\_REPU$ for answers with at least one insecure coding practice and for answers with no appearance of insecure coding practice. Along with Mann-Whitney U test, we also apply Cliff's Delta [21] to compare the distribution of each metric between insecure and neutral answers. Both Mann-Whitney U test and Cliff's Delta are non-parametric. The Mann-Whitney U test states if one distribution is larger/smaller than the other, whereas effect size using Cliff's Delta measures how large the difference is.

$$NORM\_USER\_REPU = \frac{\text{user reputation score}}{\text{membership period (months)}} \quad (1)$$

Findings: We do not observe any significant difference for user reputation between insecure and neutral answers. The mean $NORM\_USER\_REPU$ scores for insecure answers and neutral answers is respectively 11.0 and 12.1, but the difference is not significant ($p - value = 0.9$, Cliff's Delta = 0.01). The distributions of $NORM\_USER\_REPU$ scores for insecure answers and neutral answers in Figure 3.

We conclude that contrast to our hypothesis we observe user reputation to have no significant relationship with presence of insecure coding practices. Our findings indicate that both answer providers with high and low reputation are susceptible of introducing insecure code blocks in SO answers.

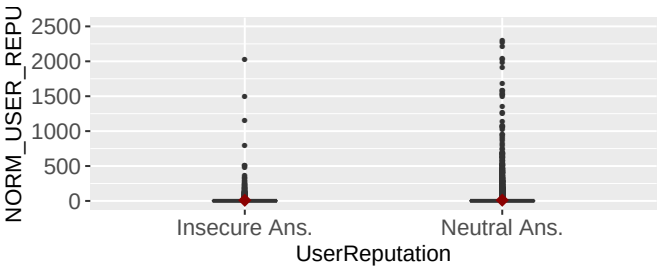

Fig. 3: Distribution of $NORM\_USER\_REPU$ scores for insecure and neutral answers. The difference between the two distributions is insignificant ($p - value = 0.9$)

E. Answer to RQ3: What are the characteristics of Python-related questions that include answers with insecure code practices?

Prior research [5] [22] reported that summarization of SO question titles yield insights on the use and challenges when working with a certain technology. Summarization of questions that include insecure code blocks can provide us clues on what categories of operations or problems are associated with insecure coding practices.

Approach: Similar to prior research [6] [22], we use Latent Dirichlet Allocation (LDA)-based topic modeling [10] to categorize questions associated with an insecure answer. We apply LDA-based topic modeling on the questions title for two categories of questions: questions with at least one insecure answer, and questions with no insecure answer. We apply topic modeling by executing the following steps:

Step-1: Pre-processing: We pre-process the question titles by removing stop words and punctuation symbols, and applying Porter stemming [23].

Step-2: Determining topic count: We determine the required topic count by calculating the perplexity metric. In the case of topic modeling, perplexity is the measurement of how a probability-based topic model predicts a certain corpora

TABLE IV: Answer to RQ3: Characteristics that are exhibited in questions that include insecure answers

	Questions with at least one insecure ans.		Questions with no insecure ans.	
Index	***Label***	Words	***Label***	Words
1	String	string,list,convert,check,python,way,character	Module	module,import,error,using,argument,logging,data
2	Data processing	array,numpy,value,panda,column,using,list	Filesystem	file,python,using,line,directory,text,get
3	Filesystem	file,python,using,get,way,text,download	Data type	list,python,string,way,class,value,function
4	Web	django,object,using,form,field,matplotlib,dictionary	Plotting	numpy,array,matplotlib,image,using,point,plot
5	Random number	list,python,numpy,random,array,file,generate	Data processing	django,using,panda,column,string,file,field

of text. A low perplexity score indicates the topic model being good at predicting the text corpora of interest [10]. We construct topic models using the pre-processed question title with i topics, where i is [5, 10, 15, ... , 100]. The topic count for which we observe the lowest perplexity score is determined as topic count needed for Step-3.

Step-3: Applying LDA : Using the determined topic count we apply LDA on the pre-processed question title corpus. We use the Genesim library [24] to implement LDA.

Step-4: Assigning Labels: Our analysis from Step-3 will yield a set of words that include in each topic. To make the topic models interpretable, we perform qualitative analysis to assign a label for each topic. Two raters who are PhD students independently perform this labeling by inspecting the words that belong to each topic. Upon completion, the disagreements are discussed, and resolved. We also report the Cohen's Kappa [14].

Findings: The topic labels that are unique to questions that include insecure answers are 'String', 'Web' and 'Random number'. Our findings suggest when SO users ask about string, web and random number generation related questions in Python, the provided answers to these questions may include insecure coding practices. We report the labels of each topic along with the top 10 words that include in each topic in Table IV. The 'Label' column presents the assigned label for each topic. The first and the second author individually assigned labels. They agreed on nine labels (Cohen's Kappa = 0.9), the single label that they disagreed upon is resolved upon discussion.

The topic count needed for both text corpuses: questions with at least one insecure answer and questions with no insecure answers in Figure 4. In Figure 4 the x and y-axis respectively presents the count of topics and the perplexity score. For both questions with insecure answers and questions with no accepted answers, the topic count for which perplexity is the lowest five.

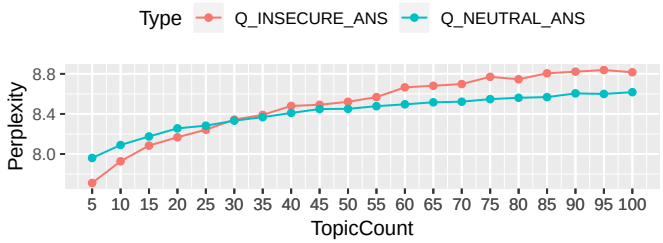

Fig. 4: Perplexity score to determine topic count. The perplexity score is the smallest when topic count is five.

F. Related Work

Our paper is closely related to prior research that have studied the interplay between software security and SO posts. Yang et al. [25] applied LDA on SO post to categorize what security-related questions developers ask about. Meng et al. [11] studied bad coding practices related to the security of Java Spring Framework in Stack Overflow, and reported 9 out of 10 SSL/TLS-related posts to discuss insecure coding practices. Nadi et al. [3] studied 100 SO posts to show that developers find cryptography algorithms to use securely, even though they are confident in choosing the correct cryptography libraries. We take inspiration from these research studies and investigate how frequently insecure Python code blocks appear in SO answers.

G. Threats to Validity

We consider the insecure coding practices provided by practitioners from Openstack, which is not comprehensive. We use string matching to detect the presence of insecure coding practices in SO answers, which is susceptible to false positives. We mitigate this threat by manually inspecting 100 SO answers for which at least one category of insecure coding practice appeared. Also, our categorization process which yielded the six insecure coding practices is subject to rater bias, which we mitigate using multiple raters. The assigned labels in RQ3, is also subject to rater bias. We mitigate this threat by applying two raters.

III. Conclusion

Insecure code blocks posted on SO can help in propagating Python-related insecure coding practices. In our empirical study we use 529,054 code blocks collected from 44,966 answers posted on SO to investigate the prevalence of insecure coding practices. We observe 3,685 code blocks to contain at least of one of the six insecure coding practices: code injection, cross site scripting, insecure cipher, insecure communications, race conditions, and insecure data serialization. The dominant insecure coding practice is code injection. We observe 7.1% of the 44,966 SO answers to contain at least of one insecure coding practice. We observe users' reputation to not relate with frequency of insecure answers. We identify three topics that are unique to questions which include at least one insecure answer. These topics are: string, web, and random number generation. Based on our findings we recommend the SO community to be aware of possible security weaknesses in Python-related answers that are posted in SO, and carefully use them for software development.

REFERENCES

[1] S. Baltes, C. Treude, and S. Diehl, "Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets," in *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)*, 2019.

[2] F. Fischer, K. Bttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy amp;paste on android application security," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 121–136.

[3] S. Nadi, S. Kruger, M. Mezini, and E. Bodden, ""jumping through hoops": Why do java developers struggle with cryptography APIs?" in *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*, 2016.

[4] S. Wang, T.-H. Chen, and A. E. Hassan, "Understanding the factors for fast answers in technical q&a websites: An empirical study of four stack exchange websites," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 884–884. [Online]. Available: http://doi.acm.org/10.1145/3180155.3182521

[5] A. Rahman, A. Partho, P. Morrison, and L. Williams, "What questions do programmers ask about configuration as code?" in *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*, ser. RCoSE '18. New York, NY, USA: ACM, 2018, pp. 16–22. [Online]. Available: http://doi.acm.org/10.1145/3194760.3194769

[6] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, Jun 2014. [Online]. Available: https://doi.org/10.1007/s10664-012-9231-y

[7] "How can i calculate a hash for a filesystem-directory using python?" (Date last accessed 29-Jan-2019). [Online]. Available: https://stackoverflow.com/questions/24937495/

[8] "NIST policy on hash functions," (Date last accessed 15-Jan-2019). [Online]. Available: https://csrc.nist.gov/projects/hash-functions/nist-policy-on-hash-functions

[9] "The 2018 top programming languages," (Date last accessed 30-Jan-2019). [Online]. Available: https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937

[11] N. Meng, S. Nagy, D. D. Yao, W. Zhuang, and G. A. Argoty, "Secure coding practices in java: Challenges and vulnerabilities," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 372–383. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180201

[12] "Paper dataset," (Date last accessed 05-Jan-2019). [Online]. Available: https://figshare.com/s/588b0d450310c05d25ab

[13] "blacklist_calls," (Date last accessed 15-Jan-2019). [Online]. Available: https://docs.openstack.org/bandit/1.4.0/blacklists/blacklist_calls.html

[14] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960. [Online]. Available: http://dx.doi.org/10.1177/001316446002000104

[15] "Exploiting python code injection in web applications," (Date last accessed 28-Jan-2019). [Online]. Available: https://sethsec.blogspot.com/2016/11/exploiting-python-code-injection-in-web.html

[16] "Security in django," (Date last accessed 18-Jan-2019). [Online]. Available: https://docs.djangoproject.com/en/2.1/topics/security/

[17] "Java and python contain security flaws that allow attackers to bypass firewalls," (Date last accessed 28-Jan-2019). [Online]. Available: https://www.bleepingcomputer.com/news/security/java-and-python-contain-security-flaws-that-allow-attackers-to-bypass-firewalls/

[18] "Create, use, and remove temporary files securely," (Date last accessed 28-Jan-2019). [Online]. Available: https://security.openstack.org/guidelines/dg_using-temporary-files-securely.html

[19] "Dangerous pickles malicious python serialization," (Date last accessed 15-Jan-2019). [Online]. Available: https://intoli.com/blog/dangerous-pickles/

[20] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947. [Online]. Available: http://www.jstor.org/stable/2236101

[21] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological Bulletin*, vol. 114, no. 3, pp. 494–509, Nov. 1993.

[22] M. Allamanis and C. Sutton, "Why, when, and what: Analyzing stack overflow questions by topic, type, and code," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, May 2013, pp. 53–56.

[23] M. F. Porter, "Readings in information retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. An Algorithm for Suffix Stripping, pp. 313–316. [Online]. Available: http://dl.acm.org/citation.cfm?id=275537.275705

[24] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[25] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, Sep 2016. [Online]. Available: https://doi.org/10.1007/s11390-016-1672-0