Towards Better Information Highlighting on Technical Q&A Platform

by

Shahla Shaan Ahmed

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science

Under the supervision of
Dr. Shaowei Wang

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

Abstract

Navigating the knowledge on Stack Overflow (SO) remains challenging. To make the posts vivid to users, SO allows users to write and edit posts with Markdown or HTML so that users can leverage various formatting styles (e.g., bold, italic, and code) to highlight the important information. Previous studies show benefits of information highlighting in various domains (e.g., improving the reading time of humans). However, little is known about how information is highlighted on technical Q&A sites (e.g., Stack Overflow).In this study, we carry out the first large-scale exploratory study on the information highlighting in SO answers. It was observed that overall, information highlighting is prevalent on SO,i.e., 47.6% of the answers have information highlighted. More specifically, 38.5%, 11.3%, 7.2% of the answers use Code, Bold, and Italic, respectively. Besides source code-related content (e.g., identifiers, and programming keywords), users also frequently highlight updates (e.g., updates of answers), caveats (i.e., a reminder or warning of in which context or condition the provided solution works or does not work), and reference. Users tend to highlight the code words more than other tags. To ease up the highlighting process, we develop approaches to recommend highlighted content automatically by using neural network architectures initially designed for Named Entity Recognition task. models are trained for each type of formatting (i.e., Bold, Italic, Code, and Heading) using the information highlighting dataset we collected from SO answers. The models with CNN architecture achieve precision values ranging from 0.71 to 0.82. While the recall values are much lower than precision values, the model for automatic code content highlighting achieves a recall of 0.73 and an F1 score of 0.75, outperforming the others. The results of these models were later compared with BERT models trained on our datasets. The analysis of failure cases indicates that the majority of the failure cases are missing identification (i.e., the model misses the content that is supposed to highlight) due to that models tend to learn the more frequent highlighted words while struggling to learn less frequent words.

Acknowledgments

I would like to express my gratitude to my supervisor, Dr. Shaowei Wang for the constant support and guidance on my work. I am grateful to the advising committee Dr. Carson Leung and Dr. Max Targeon for their useful advice regarding my work. I am sincerely grateful to the faculty members of the Department of Computer Science and other departments. I am thankful to my lab members for helping me through the project work. I am truly grateful to my collaborators in my publications for helping in my work. Lastly, I would like to thank my family members and friends for supporting me emotionally during my thesis work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Information presentation in Stack Overflow

Technical question-answering sites for developers have become increasingly important for software developers to share knowledge and contribute to communities. Stack overflow is the most known site to developers due to its popularity. This interactive platform provides good resources of knowledge related to all fields of computer science and software engineering. Despite Stack Overflow's success and prevalence, navigating the knowledge on it remains challenging [45, 43, 20, 7]. The previous study shows that finding answers in long posts remains one of the challenges [43, 20]. 37% of all questions on Stack Overflow has more than one answer and the average length of an answer is 789 characters [20]. To make the posts vivid to users, Stack Overflow platform allows users to edit their posts with Markdown and HTML [32], so that users can leverage various formatting styles (e.g., Bold, Italic, and Code) to highlight text and direct other users' attention toward the most important information within posts. Previous studies show the benefits of information highlighting in various domains (e.g., saving the reading time of humans). However, little is known about how information is highlighted on technical Q&A sites (e.g., Stack Overflow). For example, how prevalent is the information highlighted? What content is highlighted using different formatting styles? Understanding this could provide a landscape of the usage of information highlighting on technical Q&A sites and shed light on the

information that is considered necessary to developers.

1.2 Research Questions and Contributions

1.2.1 Research questions

To conduct the work properly, the study was formulated by answering the following four research questions:

- RQ1: How prevalent is the information highlighting in SO answers?
- RQ2: What types of information are highlighted in *Code* formatting in SO answers?
- RQ3: What types of information are highlighted in *Bold*, *Italic*, *Delete*, and *Heading* formatting in SO answers?
- RQ4: Can we automatically highlight the text?

In RQ1, the aim is to understand the prevalence of the usage of the studied highlighting formatting and its characteristics. Understanding this can provide practitioners an overview of the landscape of information highlighting in Stack Overflow answers. I study *Code* formatting and the rest formatting types (i.e., *Bold*, *Italic*, *Heading*, and *Delete*) in separated RQs (RQ2 and RQ3) due to their different nature. I study the information highlighted by different formatting types. By understanding this, it can provide insights into what information is important from users' perspectives and provide insights for future research. For instance, this study can provide insight into the downstream research that leverages SO information to facilitate software engineering tasks (e.g., API documentation enrichment[14, 36]). For the RQ4, the possibility of automation text highlighting in Stack Overflow using deep learning methods was explored. I used Named Entity Recognition (NER) model for the experiments. I trained the models based on each type of tags.

1.2.1.1 Overview of highlighting in Stack Overflow

The analysis of 55,209,643 information highlighting instances across 14,845,929 answers on Stack Overflow reveals that:

- Overall, information highlighting is prevalent on SO, i.e., 47.6% (14,845,929 out of 52,166,061) of the answers use the studied formatting types to highlight information. 38.5%, 11.3%, and 7.2% of the answers use *Code*, *Bold*, and *Italic*, respectively, and their highlighted content is short (median length is one word).
- *Code* formatting is mainly used to highlight source code content, such as identifiers (63.5%). *Code* is also used to highlight content other than source code, such as Software (4.9%) and Equation (5.2%).
- *Italic* and *Bold* are frequently used to highlight source code content, content that warns about the context where the provided solution works or does not work, updates on answers, and references to an internal or external resource.

1.2.1.2 Automation of the highlighting

Since information highlighting is prevalent on SO, highlighting information is not always straightforward, typically for less-experienced users. Recommending important information to highlight in a post can help improve the presentation of the post and make the important information more visible to SO users. So the potential of recommending highlighted content automatically was investigated. To do so, this work adapts deep learning-based named entity recognition (NER) models to automatically identify highlighted content for each type of formatting. More specially, I train NER models on each formatting (i.e., Heading, Bold, Italic, and Code) type from scratch and recommend the content to highlight. The models based on CNN architecture achieve precision values ranging from 0.71 to 0.82. While the recall values are much lower than precision values, the CNN models for automatic source code content highlighting achieve a recall score of 0.73 and an F1 score of 0.75, outperforming the others. These models were compared with BERT models and the F1 scores were higher than the BERT models. By analyzing the failure cases, it was observed

that the majority of the failure cases are due to missing identification (i.e., the model misses the content that is supposed to highlight). It was shown in the analysis that the models tend to learn the frequent highlighted words while struggling to learn less frequent words (i.e., long-tail knowledge).

1.3 Organization of the document

The following chapter discusses the details of the background and the related work. The next chapter explains the methodology. Chapter 4 explains this work's experiment environment and dataset preparation. In Chapter 5, results of the analysis were presented on highlighted information in Stack Overflow which combines the first three research questions. Chapter 6 elaborates and discusses results on automation of highlighting and the failure cases. Finally, Chapter 7 discusses the implications of this study and threats to validity, remarks, and concludes the study.

Chapter 2

Literature Review

2.1 Background

Stack Overflow allows users to use Markdown and HTML to write and edit posts [32]. Certain formatting types are used to highlight the information in text description with Markdown and HTML tags.

In this study, I focus on the five most commonly used formatting types, which are Bold, Italic, Delete, Code, and Heading. In table 1, the HTML tags were presented and their equivalent Markdown syntax for each formatting type. The HTML tags and

Table 1: The studied formatting types that are used to highlight information, their corresponding HTML tags, and equivalent Markdown. HTML tags are grouped based on their rendering effects.

Formatting type	**HTML tags**	**Equivalent Markdown**
Code	code	'example'
Bold	b, strong	**example**, __example__
Italic	i, em	*example*
Delete	del, s	None
Heading	h1, h2, h3, h4, h5, h6	# example, ## example, ### example, #### example, ##### example, ###### example

Markdown formatting based on their rendering effect were grouped. For example, *b* and *strong* were grouped as Bold since they render the same effect in the browser when users read the posts on Stack Overflow. In this study, it was considered that a highlighted sequence of words as an *highlighted instance*. One post could have multiple highlighted instances. For instance, in Figure 1, the post has 10 Code instances[1].

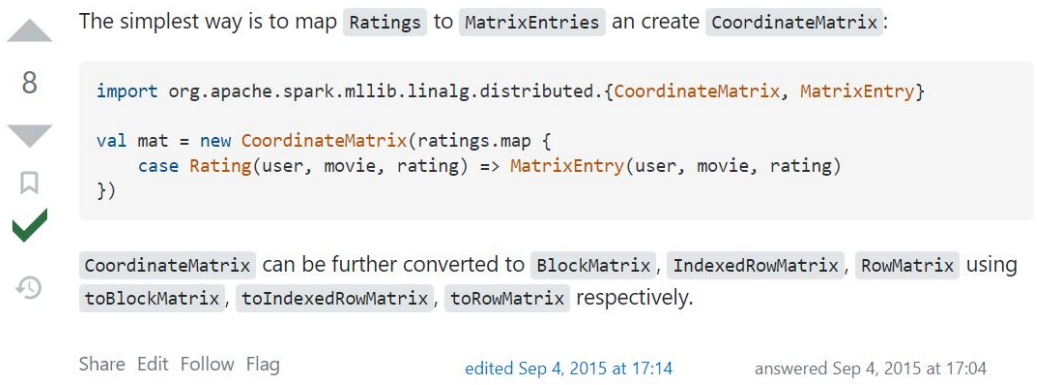

Figure 1: An example of SO answers in which the code-related content is highlighted.

In the rest of this paper, for simplicity's sake, it was referred to the formatting types *Bold*, *Italic*, *Heading*, and *Delete* as *Text* formatting. *Code* and *Code* formatting, *Text* and *Text* formatting were used exchangeably.

2.2 Related Work

2.2.1 Importance of Highlighting

Effects of highlighting have been studied by other domains [33, 42, 41, 24, 21]. Strobelt et al. [33] study suggests design guidelines for the effectiveness of nine web-friendly text highlighting techniques in multi-annotation cases. Wu and Yuan have shown that highlighting could reduce the cognitive load and thus reduce reading time[42]. Jorge et al. investigated the impact of highlighting text in the text classification tasks in machine learning area and found that highlighting is effective in

[1] `https://stackoverflow.com/questions/32402475`

reducing classification effort for humans[24]. On a similar note, to elaborate the output of ML models and demonstrate the effectiveness through the explanation, Ngyuen et al. developed a tool taking the highlighting portion of text into consideration[21]. Various studies have been done to investigate the impact of information highlighting in software engineering, such as code comprehension [2, 8, 22, 26], source code evolution [4]. For instance, Advait Sarkar investigated the effect of syntax highlighting on program comprehension and its interaction with programming experience [26], and found that syntax highlighting significantly improves task completion time. Different from prior studies that focus on understanding the influence of information highlighting, this study focus on understanding the practice of information highlighting on SO.

2.2.2 Knowledge Retrival on Stack Overflow

Several studies have investigated developers' challenges in retrieving knowledge from technical Q&A sites and provided some tools to facilitate the process [43, 45, 7, 20]. For instance, Sarah and Treude investigated the possibility of using two existing techniques, wordpartten and lexrank, to identify the essential sentences from a long SO answer [20]. Gottipati et al. found that in software forums it is a painstaking process for users to manually search through answers in various threads of posts [7], and they developed a customized search engine to find relevant answers from various software forums. Similarly, Xu et al. developed a technique called AnswerBot, which takes input as the technical question and generates an answer summary for the question [43].

For getting the solution correctly, answer quality is important to maintain. The study by Zhang et al.[48] showed that obsolete answers were prevalent in Stack Overflow. They studied the extracted information of answers, categorized their reasons for obsoletion, the tags of the answers, and who identified the obsoleteness. Apart from the answers, Stack Overflow provides a comment section to the users. The role of comments was studied earlier [46, 45, 29]. Zhang et al. [46] investigated and found that comments that were posted later were more informative. Zhang et al. also

pointed out an issue with the current comment ranking and display mechanism on SO (e.g., a large portion of useful comments are not visible to users) and proposed an alternative ranking mechanism to alleviate the issue [45].

Along with the improvement, whether the comments had an impact on answers was studied by Soni et al[29]. They analyzed five tags(Java, Javascript, Python, PHP, Android) and found that among all the comments only 4.6% of the comments got an answer updated.

However, studies related to understanding the use of information highlighting and text formatting tags in posts are yet to be analyzed. Prior studies focused on information retrieval from Stack Overflow. This study focuses on understanding the practice of information highlighting on Stack Overflow.

2.2.3 Recommendation System in Stack Overflow

Apart from analyzing the structural components of Stack Overflow, previous studies developed different recommendation tools to ease the use of the Q&A site such as recommending hyperlinks [16, 15], tags [9, 17, 25, 38, 40], and similar questions [39, 44].

Recommendations for hyperlinks in Stack Overflow posts were developed earlier. Li et al. [16] worked on developing Wislinker, a hyperlink recommendation tool for stack overflow which extracted knowledge from discussion, then turned knowledge into wisdom by learning through the knowledge dissemination history. On a similar note, LinkLiv recommendation system [15] was developed by Li et al. to recommend hyperlinks. It used multiple features, including hyperlink co-occurrences in Q&A discussions, and locations (e.g., question, answer, or comment).

Rekha et al. [25] proposed a hybrid auto-tagging system for Stack Overflow. The auto-tagging system included a programming language detection system and SVM based question classification system. Wang et al. developed EnTagRec, which combines two complementary lines in the statistics community Bayesian and frequentist [40]. Wang et al. [38] developed SOTagRec combining convolutional neural network and collaborative filtering model to infer tags for new postings by studying

the historical postings and their tags. Maity et al. developed [17] DeepTagRec, learnt from the content representation of question title and body, and recommended appropriate question tags on Stack Overflow. Apart from tag recommendation, similar question recommendation systems were developed [44, 39]. Yin et al.[44] provided a question-driven source code recommendation service for users based on the matching model. Similarly, Wang et al. [39] worked on developing their model which combines topical interest, topical expertise, and activeness to recommend answers for new questions.

Similar to these studies, I would like to provide a recommendation for information highlighting in the answers. As the model is designed to learn from the answers highlighted information and classify them as the tags, Named Entity Recognition(NER) models suit this problem more.

Chapter 3

Methodology

Previous studies have shown that highlighting has benefits. Currently, Stack Overflow provides users with different formatting tags in Markdown and HTML(Bold, Italic, Code, etc.). It is provided so that users can focus on important information. This research aims to understand how these highlighting tags are being used, what kind of information is tagged here, and develop a recommendation system.

3.1 Our Approach

In Stack Overflow posts, two types of tags are used; *Code* and *Text* formatting tags. Code tags are mainly used for coded blocks and in the text as well. The scope of work also extends to understanding what type of information is highlighted by the code tag and what type of information is highlighted by other text formatting tags (Bold, Italic, heading, delete).

- Collection and analysis of the highlighted contents
- Analysing the contents under tags
- Train language models to automatically detect the highlighting

3.1.1 Collection and analysis of the highlighted contents

To understand how prevalent this information is, a calculation of the presence of formatting tags(combined and for each type) in answers with respect to all answer posts is done. The distribution of instances for each formatting and the number of words for each instance are calculated. Based on this result, statistics for each type of tag are collected.

3.1.2 Analysing the contents under tags

3.1.2.1 Code tags

To review what kind of content is highlighted under the tag of Code, a sampled amount can be taken and the samples were identified according to the type of information. For this part, the data will be manually labeled.

Since there are no existing terminologies that could be reused for this purpose, it was done by manually performing a lightweight open coding-like process [27, 47] to identify the type of content highlighted with *Code*. The process involves three phases and is performed by the two different persons. For this purpose, I can find another person or friend to perform this analysis. We consider one person as A and another as B. In phase I, A and B can derive a draft list of types based on 50 *Code* instances. After revising and refining the types in phase I, A and B independently apply the derived types to the rest 335 samples. In phase III, A, and B can discuss the results from Phase II to resolve any disagreements until a consensus is reached.

3.1.2.2 Text tags

To review what kind of content is highlighted under the tag of text formatting tags, a sampled amount can be taken and the samples can be identified according to the type of information. We can follow the same methodology as code tags for this part.

3.2 Named Entity Recognition(NER) Model

This task shares similarities with NER, such as the short length of labeled (highlighted) tokens and a limited number of text labeling (formatting) options. Hence, neural network architectures were initially designed for Named Entity Recognition (NER) without actually training NER models. Moreover, both NER and automated information highlighting can be formed as a sequence labeling task. As the targeted entity is different from the regular models, the neural networks were trained from scratch on my information highlighting dataset.

3.2.1 Model Architecture

Figure 3 presents the architecture of the spaCy model. This research uses the *Token2Vec* model [31] provided by a well-known open-source software library for advanced Natural Language Processing (NLP) tasks named spaCy [30]. It opted to utilize spaCy since it has emerged as the de facto standard for practical NLP due to its speed, robustness, and performance [13]. The model architecture comprises a token embedding layer (128 dimensions), four convolutional layers, and an output layer. The token embedding layer is named as *MultiHashEmbed* by spaCy. It features a multi-key hashing mechanism for embedding lookup tables by separately embedding a number of lexical attributes and concatenating them with the vector semantic representation of a token. As it uses hashing to store embeddings in memory, it also supports faster processing of large datasets. The embedding layer is followed by a 4-layered Convolutional Neural Network (CNN), named *MaxoutWindowEncoder*. Each of the convolutional layers is followed by a max pooling layer. The final layer is a softmax layer for predicting labels on tokens. Training employs cross-entropy loss and the Adam optimizer.

3.2.2 Model Architecture

For comparison, BERT is a state-of-the-art transformer model pre-trained on a large dataset. It can be fine-tuned for numerous natural language processing tasks, en-

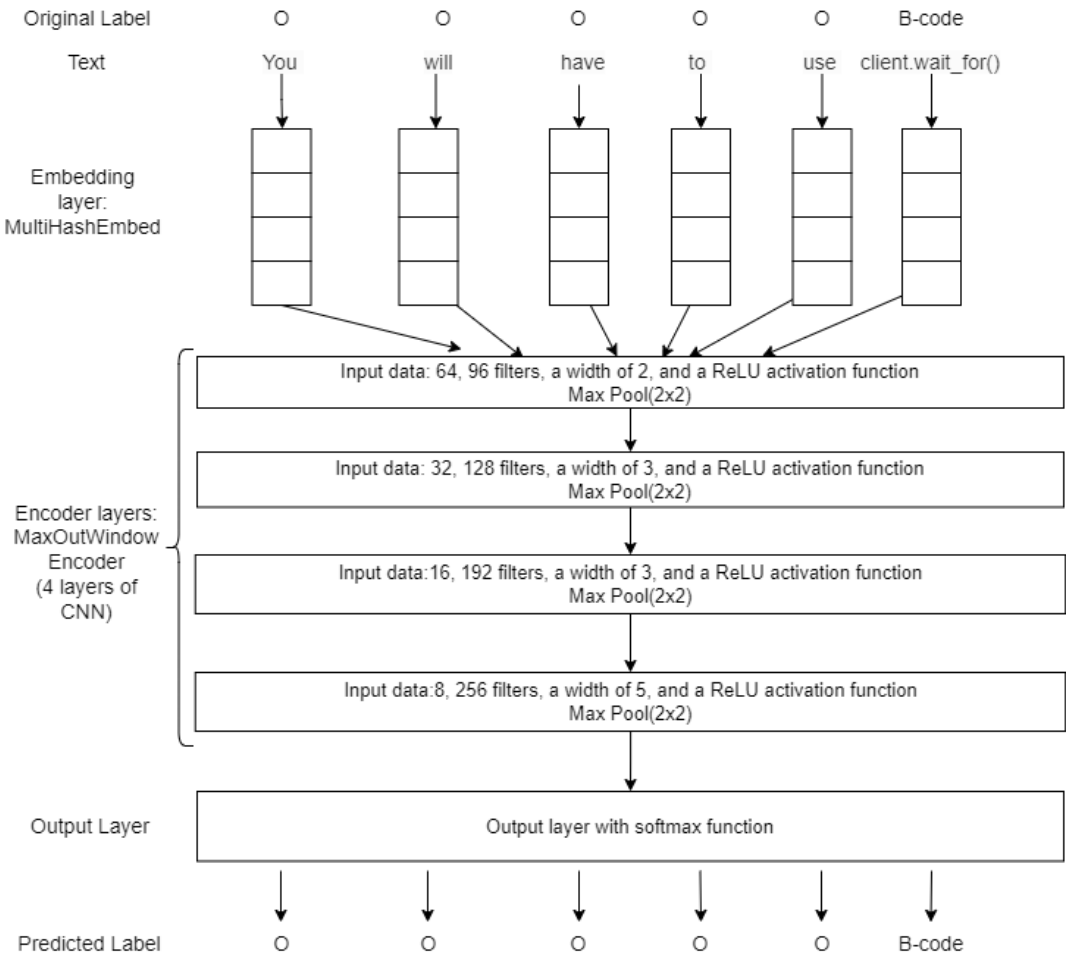

Figure 2: Layers of the NER model by spaCy

compassing question answering, language inference, text classification, and sequence tagging. The architecture of the model is based on a multi-layer bidirectional Transformer encoder developed by researchers [3] of Google. The model considers the contexts of both preceding and subsequent words. The architecture of BERT consists of multiple transformer blocks, each containing a self-attention mechanism for getting context of the input sequence. In this work, the model was fine-tuned with custom labels and a custom dataset for performing NER tasks. In this work, the model was fine-tuned from the pre-trained model provided by the Hugging Face [6] platform.

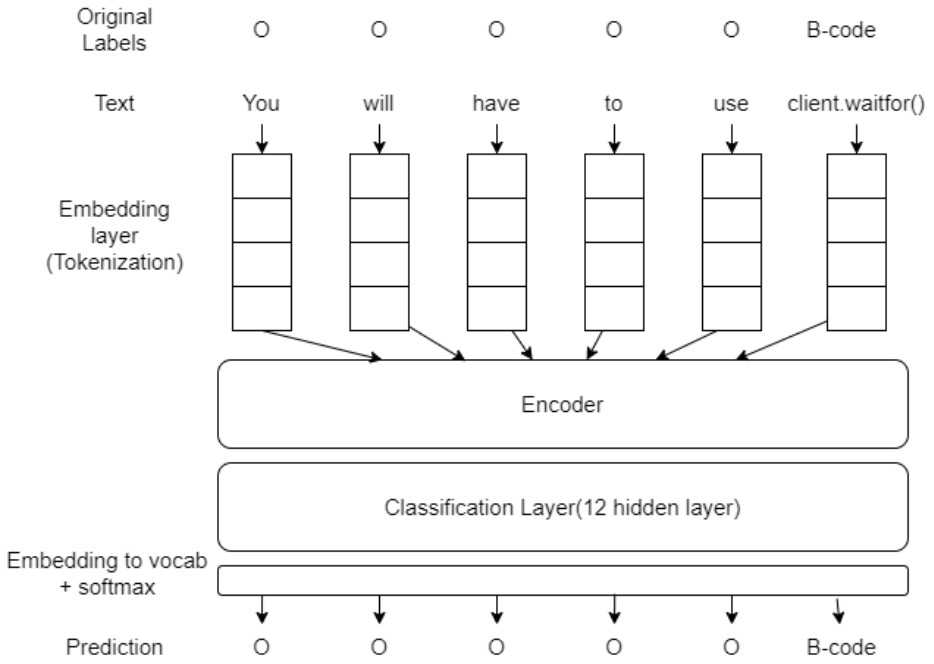

Figure 3: Layers of the NER model of BERT

3.2.3 Applying NER into separate models

As the HTML highlighting tags were categorized into five distinct types (see Chapter 2.1 for details), initially training a universal model that would accommodate all five types was attempted. However, this approach yielded unsatisfactory results, as the model tended to highlight words as code by default while ignoring other formatting types. Consequently, separate models for each highlighting type were trained. The analysis of highlighted content showed that the Delete type was scarcely used in comparison to the other types, leading us to omit it from analysis for training models. Therefore,four models were trained, i.e., $Model_{Bold}$, $Model_{Italic}$, $Model_{Heading}$, and $Model_{Code}$.

Table 2: Models parameters

Parameter	**spaCy**	**BERT**
Epochs	3	6
Learning rate	0.001	0.05
Batch size	32,4-32	16

3.2.4 Train language models to automatically detect the highlighting

For this part, Natural Language Processing models are better to train. Since these tasks need to be identified inside the sentences. Named Entity Recognition or NER models are used to recognize a specific set of words or entities inside the sentence which is mostly used for Parts of speech recognition or in identifying medical entities. For identifying the highlighted information in sentences, locating the entities as tagged and untagged is required. Thus, NER models are more suited for this. Two models for NER are going to be used here. NER Model provided by spaCy which is based on a simple feed-forward neural network with one hidden layer which is similar to Lample's[12] LSTM-based NER architecture. For comparison, the batch size of the models during training was changed.

3.3 Evaluation

This study uses precision, recall, and F1-score to measure the performance of models, following prior studies evaluating NER models [5, 10]. Instead of using the exact match (i.e., a true positive is counted when all of the tokens of a named entity are labeled correctly, both the length and type), it was decided to use a partial match [19, 34], in which it was considered if each individual token is predicted correctly. The partial match allows us to measure the performance of the segmentation aspects of highlighted content when the content has multiple words. More specially, I calculate the precision and recall as follows:

$$Precision = \frac{Number\ of\ words\ that\ are\ correctly\ predicted}{Number\ of\ words\ that\ are\ predicted\ by\ the\ model}$$

$$Recall = \frac{Number\ of\ words\ that\ are\ correctly\ predicted}{Number\ of\ words\ that\ are\ supposed\ to\ tag}$$

As an example for this, suppose for a long highlighted sequence of words (e.g., sudo netstat -antp|fgrep LISTEN), while only the words sudo netstat and

| fgrep LISTEN are highlighted correctly, the words "-antp |" were missing. In the exact match, such a case is considered incorrectly tagged, while in the partial match, the words sudo netstat and | fgrep LISTEN are considered as correctly predicted.

Chapter 4

Experiment Setup

4.1 Data preparation

A data dump of Stack Overflow was downloaded from the Stack Exchange data dump dated March 2021[1]. The data dump contains details information about posts (i.e., questions and answers), as well as their revision history. In this study, all the answers were included, and ended up with 52,166,061 answer posts.

As the dataset was large for processing, it was imported into MySQL. For each answer post, first, the textual content was extracted from its body and excluded code block(s). Note that, code block(s) were extracted using the tag "<pre>". Next, the regular expressions defined in Table 1 were applied to identify information highlighting in each answer post, using the Python library *Regex*. In the end, the content that is highlighted by the studied formatting types is extracted. The replication package is made public at `https://github.com/shaoweiwang2010/REP_2022_Information_highlight_SO`.

4.2 Experiment Setup:

The dataset is constructed for training the NER model by selecting posts that contain highlighted words from Chapter 4. Then the post is divided into sentences and

[1]https://archive.org/details/stackexchange

extracted the corresponding formatting tags described in Chapter 4. In other words, each data point is a highlighted instance (a sentence containing at least one type of information highlighting). When preparing the training data, the content highlighted with different formatting types is considered as the entities in the NER model. The tagged words were annotated with their formatting tags, considering their position in the phrase. For example, in the sample highlighted sentence shown in Figure 3, i.e., "You will have to use client_wait_for() .", "client_wait_for()" is formatted with a code tag. To label this sentence, the label "B-code" is assigned to the highlighted word, indicating that it is the beginning of a highlighted Code content, while the other words in the sentence are labeled as O (Outside). If multiple words are formatted, I (Interior) and E (Ending) were used to label the highlighted content's internal words and the ending word. For example, "you need to run. git push --force --all first" contains more than one word in the highlighted Code content. So the labels for the phrase would be (B-code: "git"), (I-code: "push"), (I-code: "--force") and (E-code: "--all"). Note that separate models are trained for each highlighting type. When preparing the dataset for a particular highlighting type, if a sentence has other highlighting types rather than the one that is targeted, it omits other types of highlighting types from the sentence. Table 3 presents the basic statistics of our dataset for different formatting types.

Table 3: Number of sentences in train and test datasets.

Type	**Training set**	**Testing set**
Bold	2,590,193	646,486
Italic	1,669,344	417,409
Code	9,292,312	2,324,836
Heading	406,946	102,133

Following prior studies, the dataset is partitioned into an 80:20 ratio for training and testing purposes. The models were trained in 3 epochs where the learning rate is 0.001. Two training settings were employed. Samuel et al. found that instead of using fixed-size batches (i.e., *fixed*) for training, increasing the size of batch (i.e., *incremental*) during training could potentially improve the training efficacy [28]. In

the incremental training setting, the batch size is increased for each iteration in an epoch, from 4 to 32. the function is used "util.compounding()" [35] provided by spaCy for incremental training setting. Except for the batch sizes, other parameters are kept the same for both models. I use 32 as the size of the batch in the fixed-size setting (i.e., *fixed*). For comparison, models based on BERT architecture are trained in 0.05 learning rate using 16 batches in 6 epochs. I ran our experiments on a Linux server with four Nvidia RTX 3090 GPUs, AMD Ryzen 48-Core CPU with 256 GB RAM. For simplicity, the spaCy models that are trained using the incremental size of batches are called *incremental models*, and those using the fixed size of batches are called *fixed models*.

Chapter 5

Usage of Highlighting

5.1 Prevalence of Information Highlighting

To understand the prevalence of information highlighting in Stack Overflow answers, the percentage of answers that have the studied formatting types (see Table 1) and the percentage of words highlighted in each answer with each formatting were computed. In addition, the basic statistics of information highlighting instances to understand the characteristics of each formatting type were computed. More specifically, the distribution of instances for each formatting and the number of words highlighted with them were calculated.

Overall, 47.6% (14,845,929 out of 52,166,061) of the studied answers have information highlighted. Among all the answers having information highlighted, an average of 10.6% and a median of 7.1% of the text (in words) is highlighted and each answer has an average of 3.9 and a median of 2 highlighting instances.

***Code* is used the most frequently among all studied formatting followed by *Bold* and *Italic*.** Table 4 presents the statistics of different formatting types. It was observed that *Code* is the most frequently used type. 38.5% of the studied answers have content highlighted with *Code*. Moreover, 78.9% of the highlighted instances are *Code*. In addition, the studied answers have two *Code* instances on the median, which is larger than other formatting types except *Heading*. This finding is expected since users usually discuss programming problems on SO and it is common

Table 4: The answer post-wise and highlighted instance-wise statistics of different formatting types.

Type	**%Answers**	**#Highlighted instance per answer (mean /median/ max)**	**%Highlighted words per answer (mean /median/ max)**	**%Instances**	**#Words per instance (mean /median / max)**
Bold	11.3%	2.0/1.0/241	9.0%/4.1%/100%	12.0%	2.9/1.0/436
Italic	7.2%	1.9/1.0/354	5.9%/2.3%/100%	7.3%	2.9/1.0/477
Delete	0.07%	1.2/1.0/36	18.1%/10.6%/100%	0.04%	15.7/8.0/701
Code	38.5%	3.7/2.0/490	8.7%/6.3%/100%	78.9%	1.4/1.0/4,669
Heading	1.6%	2.1/2.0/168	9.3%/3.9%/100%	1.7%	3.4/2.0/327

to highlight source code in the text. *Bold* and *Italic* are the most frequently used formats besides *Code*. They are used in 11.3% and 7.2% of answers, respectively. *Delete* is rarely used, only 0.07% of the answers use it.

In general, the length of highlighted content is short. The median length of the content highlighted with *Code*, *Bold*, *Italic*, *Deleting*, and *Heading* are 1, 1, 1, 8, and 2 words, respectively. Users tend to highlight single word or phrase using *Code*, *Bold*, and *Italic*. Compared with other formatting types, *Delete* instances are longer.

Information highlighting is prevalent on SO, i.e., 47.6% of the answers use the studied formatting to highlight information. 38.5% of the answers use *Code*, which is the most frequently used format, followed by *Bold* (11.3%) and *Italic* (7.2%). In general, the length of highlighted content is short.

5.2 Highlighted Contents in Stack Overflow

5.2.1 Code tags

To understand what content is highlighted with the *Code*. First, it was randomly sampled 385 *Code* instances with a 5% interval and 95% confidence level. Since there are no existing terminologies I could reuse for this purpose, a lightweight open

coding-like process [27, 47] to identify the type of content highlighted with *Code* was manually performed. The process involves three phases and is performed by two different persons (A1 and A2). In phase I, A1 and A2 derived a draft list of types based on 50 *Code* instances. During the phase, the types were revised and refined. In phase II, A1 and A2 independently applied the derived types to the rest 335 samples. They took notes regarding the diffidence or ambiguity during the labeling. During this phase, no new types were introduced. In phase III, A1 and A2 discussed the results from Phase II to resolve any disagreements until a consensus was reached. The coding process has a Cohen's kappa of 0.8 (measured before starting Phase III), which indicates a substantial level of the inter-rater agreement [37]. Table 5 presents the definition and the corresponding example of the derived types for *Code*.

5.2.2 Results of Code Tag Analysis

***Code* is mainly used to highlight source code elements, such as identifiers (63.5%), programming language keywords (9.9%), and statements (7.0%).** Table 5 presents the derived types of content that is highlighted with *Code* and the distribution of each type. *Code* is used the most frequently to highlight identifiers in source code, such as the name of classes, methods, parameters, and variables. It was also observed that 59% of the highlighted identifiers appearing in the code block of their corresponding answers. That is said, *Code* is commonly used to highlight identifiers in text for referring to the corresponding identifiers in the code blocks. This is reasonable since users usually need to refer to a unit in the source code when discussing the solution. In addition, 9.9% and 7.0% of the studied instances are for Keyword, and Statement, respectively.

***Code* is also used to highlight content other than source code, such as Software (4.9%), Terminology (1.8%), Equation (5.2%), and Version (0.5%).** From Table 5, it was observed that users also use *Code* to highlight content other than code. For instance, in 5.2% of the cases, *Code* is used to highlight content related to equations. For example, in an answer, answerer discuss the time

complexity of binary search "This is O(log n)", which is highlighted using *Code*[1]. *Code* sometimes is used to emphasize the names of a software, a framework, and a lib. For instance, the answerer of an answer mentioned "You are using Mysql as DB ..." and show a piece of code to demonstrate the database connection between Java and Mysql[2].

Table 5: The definition and distribution of types of content highlighted with *Code* formatting.

Type	**Definition (example)**	**Count (%)**
Identifier	The identifier in source code, e.g, "ArrayList".	245 (63.6%)
Keyword	The keywords in programming languages, e.g., "for" and "public".	38 (9.9%)
Statement	A statement of source code, e.g., "plot = last.plot()".	27 (7.0%)
Equation	Mathematical equation, operator, or number, e.g., "O(log n)".	21 (5.2%)
Software	The name of softwares, frameworks, tools, and libs, e.g., "Tensorflow" and "MySQL".	19 (4.9%)
Path	Path and files name, e.g., "src/main/java".	17 (4.4%)
Terminology	Terminologies related to programming, e.g., "hash table".	7 (1.8%)
Cmd	Command, e.g., "cloud-init init"	5 (1.3%)
Version	Version information of a software, e.g., "62.1".	2 (0.5%)
Other	Other than the above defined types.	4 (1%)

Although *Code* is mainly used to highlighted the content related to source code, it is also used to highlight content other than source code, such as Software (4.9%), Terminology (1.8%), Equation (5.2%), and Version (0.5%).

[1]https://stackoverflow.com/questions/17117375
[2]https://stackoverflow.com/questions/24586043

5.2.3 Text tags

In this part, the aim is to understand what content is highlighted with the studied *Text* formatting (i.e., *Bold*, *Italic*, *Delete*, and *Heading*). Similar to the code tag analysis, a manual study was performed. 385 *Text* instances with a 5% interval and 95% confidence level were randomly sampled. This task ended up with 177 *Bold*, 169 *Italic*, 3 *Delete*, and 36 *Heading* instances. We then identified the types of highlighted content by following the methodology used in RQ2. The coding process has a Cohen's kappa of 0.78 (measured before starting Phase III), which indicates a substantial level of the inter-rater agreement [37]. Table 6 presents the definition and the corresponding example of the derived types for *Text* formatting.

5.2.4 Results of Text Tag Analysis

Both *Bold* and *Italic* formatting are most frequently used to highlight content related to source code. Table 6 presents the distribution of each *Text* type. We observe for certain types, *Bold* and *Italic* share similar patterns. For instance, in 28.8% of the *Bold* instances and 30.8% of the *Italic* instances, content-related source code (i.e., Source code) is highlighted. That is said, users also frequently use *Bold* and *Italic* to highlight source code other than using *Code* formatting. Users probably use *Bold*, *Italic*, and *Code* exchangeably, although SO suggests users to tag inline code using *Code* format [32]. Interestingly, it was also observed in some cases, the community members changed formatting of the code-related content from *Bold* and *Italic* to *Code*. For instance, in an answer, a user change the formatting for a funciton "strncmp()" from *Italic* to *Code*[3].

Users frequently use both *Bold* and *Italic* to highlight Caveat, Reference, and Terminology. In some cases, it is important to mention the condition or context in which a provided solution works (i.e., Caveat). We notice that such information is usually highlighted in answers. For instance, in a SO answer, the answerer reminded readers "Donot forget add jmtp.dll files (that comes up with jmtp

[3]https://stackoverflow.com/posts/18437465/revisions

Table 6: The definition and distribution of different types of content highlighted with *Bold, Italic, Heading, Delete*.

Type	Definition (example)	Bold	Italic	Delete	Heading
Update	The update and new edit in the post, e.g., "**Update:** According to your comments, I think you will want to try gzfile() in read.table()".	22 (12.4%)	2 (1.2%)	0	0
Equation	Same as the definition of Equation in Table 5.	3 (1.7%)	2 (1.2%)	0	0
Source code	The union of Identifier/Keyword/Statement/Operator /Cmd/Path in Table 5, e.g., "I'd suggest the usage of **startupOrder** to configure the startup of route though".	51 (28.8%)	52 (30.8%)	0	1 (2.8%)
Caveat	A reminder or warn of in which context or condition the provided solution works or does not work, e.g., "**It won't work in that shell though, you need to open a new one**".	33 (18.6%)	43 (25.4%)	0	1 (2.8%)
Reference	Reference to internal or external resource, e.g., " .not()".	12 (6.8%)	17 (10.1%)	0	0
Results	Results or output.	3 (1.7%)	2 (1.2%)	0	0
Terminology	Same as the definition of Terminology in Table 5.	12 (6.8%)	21 (12.4%)	0	0
Heading	The title of a section, e.g., "**WORKING EXAMPLE**".	27 (15.3%)	2 (1.2%)	0	33 (91.6%)
Options	Options in software, e.g., "you have to enable Less **Secure Sign-In** in your google account".	4 (2.3%)	1 (0.6%)	0	0
Extent	A word/phrase to express extent, e.g., "at least with Hibernate it assumes you want to use a global "hibernate" sequence for *all* tables, which is just stupid."	4 (2.3%)	14 (8.3%)	0	0
Version	Same as the definition of Version in Table 5.	0	3 (1.8%)	0	0
Delete	Deleted outdated/wrong description, e.g., "Just use KVO KVC".	0	0	3 (100%)	0
Other	Other than the above defined types.	6 (3.4%)	10 (5.9%)	0	1

download) as a native library for more info see my answer on ..." using *Bold*.[4]. *Bold* and *Italic* are used to highlight Reference and Terminology, although *Italic* is used

[4]https://stackoverflow.com/questions/6498179

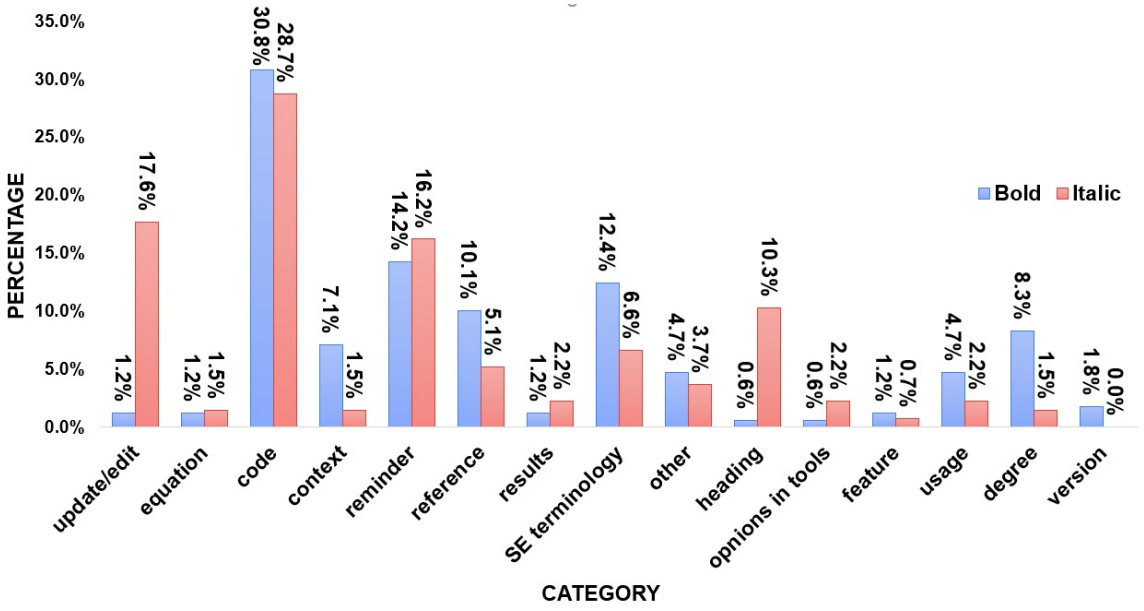

Figure 4: The distribution of categories for bold and italic format.

more often than *Bold*.

Users tend to highlight the content of types Update and Heading with *Bold*, while *Italic* is more likely to be used for the type of Extent. From Table 6, we can see differences between *Bold* and *Italic*. For instance, *Bold* is more frequently used to highlight Update (12.4%) and Heading (15.3%) compare with *Italic*. Interestingly, users also use *Bold* to highlight a heading. One possible reason is that the rendering effect for *Bold* and *Heading* are similar. Prior studies reveal that answers on SO are easy to become obsolete [23, 47]. Therefore, it is not surprising to observe that users highlight the update using *Text* formatting. Different from *Bold*, *Italic* is used more frequently to highlight a word/phrase to express the extent (i.e., Extent).

For *Delete*, it is all used to highlight obsolete or incorrect content. In terms of *Heading*, we observe that in 91.6% of the instances, users used it to highlight heading, which is expected. However, we also observe in one case for highlighting Source Code and one case for emphasizing Caveat.

Apart from source code, both *Italic* and *Bold* are used frequently to highlight content in types of Caveat, Reference, Terminology, and Update.

Chapter 6

Automate the Highlighting in Stack Overflow

Earlier chapters focused on analyzing the highlighted content on Stack Overflow. This chapter explores the feasibility of creating an automated machine learning-based approach for highlighting content. This feature would be helpful, particularly for SO users who are inexperienced in the practice of highlighting information and seek to automatically highlight critical content when composing their posts. Specifically, the aim is to create a model that can highlight information in an input answer post using appropriate formatting types (Bold, Italic, Delete, Code, and Heading).

6.1 Results of NER Models

Out of the four trained models, the Code model obtained the highest F1 score of 0.75. Table 7 presents the results of the trained models in terms of precision, recall, and F1 using fixed and incremental sizes of batches. For models provided by spaCy, in terms of F1, it was observed that the Code models (F1 score is 0.75 and 0.71 using incremental and fixed models) perform better than other models. This is reasonable, since first for code formatting, it had much more training data than other types. Secondly, code typically has certain special characters (e.g., ".", and "()" in API "class.function()") and keywords (e.g., "foreach"), making it easy for

the model to identify. The observation indicates that it is easier to build a model to identify code and highlight it on SO posts than other types of formatting.

On the other hand, BERT archives higher precision(0.699 to 0.92) than spaCy models however the recalls are below 0.05 which is the lowest recall in spaCy models. With this low recall, it ends with the F1 scores lower than the spaCy models. Only the BERT model that was trained for Code performs scores 0.343 in F1 Scoring, which is also lower than the code models of spaCy. So it can be said that spaCy models performed better than BERT models.

Table 7: The performance of models on all formatting types *Bold, Italic, Heading, Code* with different training settings. F denotes the fixed models and I denotes the incremental models.

Type	**Precision** ($\boldsymbol{F}$)	**Precision** ($\boldsymbol{I}$)	**Precision** ($\boldsymbol{Bert}$)	**Recall** ($\boldsymbol{F}$)	**Recall** ($\boldsymbol{I}$)	**Recall** ($\boldsymbol{Bert}$)	**F1** ($\boldsymbol{F}$)	**F1** ($\boldsymbol{I}$)	**F1** ($\boldsymbol{Bert}$)
$Model_{Bold}$	0.76	0.72	0.76	0.11	0.23	0.025	0.19	0.35	0.049
$Model_{Italic}$	0.82	0.58	0.92	0.05	0.12	0.003	0.10	0.19	0.006
$Model_{Heading}$	0.62	0.71	0.84	0.21	0.38	0.0093	0.31	0.50	0.018
$Model_{Code}$	0.78	0.77	0.699	0.65	0.73	0.228	0.71	0.75	0.343

The models achieve an acceptable precision ranging from 0.71 to 0.82 for different formatting types while obtaining low recall. Overall, except for the Code models, other models have higher precision and lower recall. For models for text formatting, Heading, Bold, and Italic models can achieve at least a precision of 0.71 from either fixed or incremental models while models for text formatting struggle from low recall. The results suggest that as long as the text formatting models recommend the content to highlight, the accuracy is acceptable, while the models **miss** most of the content that is supposed to be highlighted (low recall).

In Table 8, several successful examples are presented, in which the corresponding content is highlighted with correct formatting types. In the Code example, the model identifies all code-related content in the post correctly, such as `getElementById` and `querySelector`. For Italic, the model can identify "The Horror of Implicit Globals" in the sentence. For Heading, the model successfully identifies the first heading sentence.

Table 8: Successful examples of content highlighted by the models (using the incremental setting) along with original text for different types of formatting. Note that since these models are trained for one specific type of formatting, therefore, it only shows the content that is highlighted by the specific model in the examples.

Type	**Original text**	**Text highlighted with NER models (Incremental).**
Bold	**NOTE : THIS MIGHT ALREADY SOMETHING THAT YOU HAVE TRIED.** This is probably due to the fact that you are using `–trusted-host=pypi.python.org`.	**NOTE : THIS MIGHT ALREADY SOMETHING THAT YOU HAVE TRIED**. This is probably due to the fact that you are using –trusted-host=pypi.python.org.
Code	For more info on searching for the elements on DOM such as `getElementById`, `querySelector`, please refer here.	For more info on searching for the elements on DOM such as `getElementById`, `querySelector`, please refer here.
Italic	Side note: Your code is falling prey to what I call *The Horror of Implicit Globals* because you never declare filename.	Side note: Your code is falling prey to what I call *The Horror of Implicit Globals* because you never declare filename.
Heading	OPTION 1: GIT STASH You may postpone all current changes of your files in the git stash cache. After that the files are equal to former checkout state.	OPTION 1: GIT STASH You may postpone all current changes of your files in the git stash cache. After that the files are equal to former checkout state.

6.2 Case Study of Failure Cases:

The failure cases of the models were also investigated to provide some insights for future research. The failure cases could be categorized into the following three families:

- **Misidentification:** Although the content that needs highlight is successfully identified, incorrect formatting types are recommended. In other words, the content is not identified by the correct model.

- **False Identification:** The model identifies the content that is not supposed to highlight.
- **Missing Identification:** The model misses the content that is supposed to highlight.

The distribution of three failure cases over each model was calculated using the confusion matrix. For instance, to compute the number of missing identification cases from $Model_{Bold}$, the number of false negative instances in the confusion matrix on the testing data generated by $Model_{Bold}$ was counted. For this analysis, only the results from spaCy models were taken.

Table 10 presents the distribution of those three types of failures across different models. Overall, the majority of the failure cases are missing identification (e.g., 57% for incremental model) and false identification (e.g., 37% for incremental model), while misidentification takes the least portion (e.g., 5.6% for incremental model). All models suffer from high missing identification except Code models, which explains the low recall in Table 7, typically for Bold and Italic models. 87% and 72% of the failures in incremental Bold and Italic models are missing identification. One possible explanation is that the models are easy to learn the frequent highlighted words while struggling to learn less frequent words (i.g., long-tail knowledge) [11, 18]. To test this assumption, the frequency of the words that are correctly predicted and the words that are not identified by models in the training data were counted. Table 9 presents the mean/median frequency of words that are correctly predicted and those that are not identified by models. On top of it, Figure 5 compares their distributions. it is observed that the mean and median frequency of words that are predicted correctly is much larger than those of words that are not identified in all models, except the median value of Code.

Code, Italic, and Heading models tend to have false identification failures, especially the Code models. For instance, in example 7 of Table 11, the Code model identify "setTimeout" while it is not highlighted in the original text. setTimeout is indeed a function name and one possible explanation is that setTimeout is frequently highlighted with Code in the training data, and the model can recognize it. Similarly,

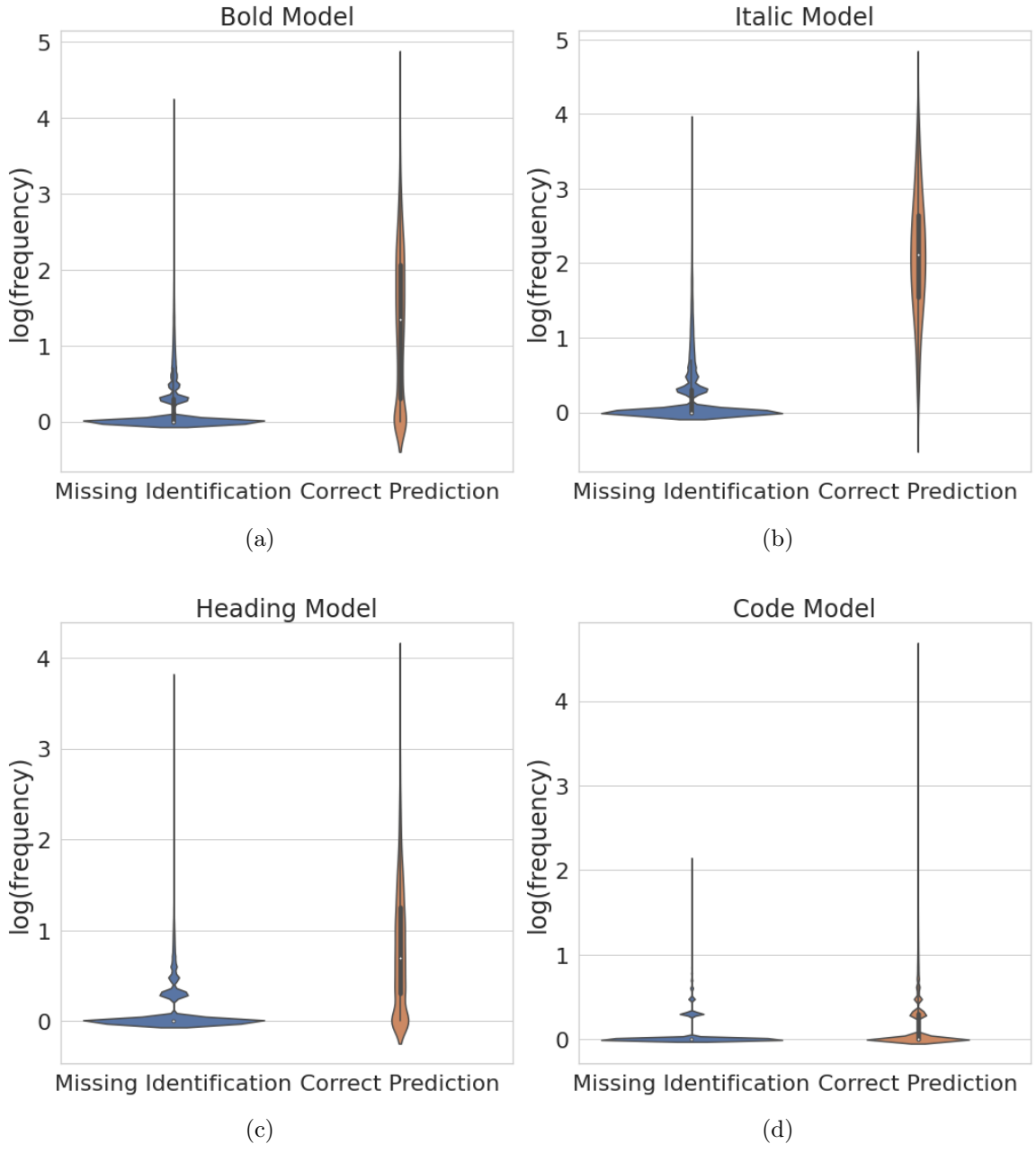

Figure 5: Comparison of the frequency distribution of words that are not identified (missing identification) and those that are correctly predicted (correct prediction) on different models.

Table 9: The mean/median frequency of words that are correctly predicted (correct prediction) and those that are not identified by models (missing identification) in different models.

Type	**Mean/Median of missing identification**	**Mean/Median of correct prediction**
Bold	4.05/1.0	193.05/22.0
Italic	6.11/1.0	674.48/130.0
Heading	2.53/1.0	34.73/5.0
Code	1.35/1.0	5.37/1.0

Table 10: The distribution of different failure types among models of different formatting types.

Failure Type	**Model Type**	**Formatting Type**				
		Bold	Code	Italic	Heading	Total
Misidentification	Fixed	6.8%	6.7%	1.6%	12%	6.2%
	Incremental	3.1%	6.3%	1.3%	1.5%	5.6%
Missing Identification	Fixed	89%	39%	66%	59%	55%
	Incremental	87%	43%	72%	46%	57%
False Identification	Fixed	4%	54%	32%	28%	39%
	Incremental	9.9%	50%	26%	39%	37%

in example 4, the Italic model identifies "I currently" after "Disclaimer:" as Italic. Recall example 3 in Table 8, in the original text, "Disclaimer: I work at SerpApi" was italicized. The model may identify the word "Disclaimer:" and tend to italicize the content after it.

A small portion of misidentification cases were also observed. For instance, in example 1 in Table 11, the original text is highlighted with Heading, while the model identifies the content but formats it using bold. One possible explanation is that those types of formatting share common usage patterns to some extent and confuse the models. Actually, such a phenomenon was observed in Chapter 5. For instance, it showed that both Bold and Italic are used to highlight source code and Caveat.

Heading is used to highlight source code.

Table 11: Failure cases of spaCy detection of tags for all the types *Bold, Italic, Heading, and code.*

ID	Type	Original text	Text highlighted with NER models (Incremental).	Failure type
1	Heading	Swift 5.2.x First of all, you need to declare an "easy to use" typealias for your block:	**Swift 5.2.x** First of all, you need to declare an "easy to use" typealias for your block:	**Misidentification:** Detected heading as bold.
2	Bold	If you want to extend it's expiration date, IT WILL THROW THE EXCEPTION!	If you want to extend it's expiration date, IT **WILL THROW THE EXCEPTION!**	**False Identification:** detected normal texts as bold.
3	Bold	**From now, your frontend application will use access token in the Authorization header for every request.**	From now, your frontend application will use access token in the Authorization header for every request.	**Missing Identification**: missed Bold.
4	Italic	Disclaimer: I currently work for Mapbox.	Disclaimer: *I currently* work for Mapbox.	**False Identification:** detected normal texts as Italic.
5	Heading	Close Sublime and Quit It. Open up your desired project that you want to use Sublime with in Unity	Close Sublime and Quit It. Open up your desired project that you want to use Sublime with in Unity	**False Identification:** detected normal texts as Heading.
6	Heading	Because there are no spaces in the text. Chinese is commonly written without any space characters between words or sentences. The ASS subtitle format, having been originally designed by and for European language speakers, unfortunately only breaks words on actual space characters. Your subtitle file does not contain any space characters, so the lines are never broken.	Because there are no spaces in the text. Chinese is commonly written without any space characters between words or sentences. The ASS subtitle format, having been originally designed by and for European language speakers, unfortunately only breaks words on actual space characters. Your subtitle file does not contain any space characters, so the lines are never broken.	**Missing Identification**: Missing Heading for "Because there are no spaces in the text".
7	Code	setTimeout seems like the right guy for the job. To *off* a click use jQuery's .off() Method. Also, the .one() method to attach a callback only *once* :	setTimeout seems like the right guy for the job. To off a click use jQuery's .off() Method. Also, the .one() method to attach a callback only once :	**False Identification:** Detected normal text as code.
8	Code	Most likely that the website you are trying to scrape is a dynamic website, meaning that it is Javascript generated code and can't be scraped with only requests and beautifulsoup .	Most likely that the website you are trying to scrape is a dynamic website, meaning that it is Javascript generated code and can't be scraped with only requests and beautifulsoup.	**Missing Identification**: Missed the Code tag.

These models achieve a precision ranging from 0.71 to 0.82 for different formatting types. It is easier to build a model to recommend Code than other types. Models for text formatting types (i.e., Heading, Bold, and Italic) suffer low recall. The analysis of failure cases indicates that the majority of the failure cases are due to missing identification. One explanation is that the models are easy to learn the

frequent highlighted words while struggling to learn less frequent words (e.g.long-tail knowledge).

Chapter 7

Conclusion

This chapter mentions the threats to the works and the possible future direction of the work.

7.1 Threats to validity

7.1.1 Internal Validity

This study involved qualitative analysis in Chapter 5. To reduce the bias, each instance was labeled by two persons, and discrepancies were discussed until a consensus was reached. We also showed that the level of inter-rater agreement of the qualitative studies is high. In Chapter 6, the models were evaluated using precision, recall, and F-1 score with partial match. Although, there might be other metrics that could be used for the task., those metrics are commonly used in NER models, and I follow previous studies [5, 10]. In RQ4, a NER architecture from spaCy was utilized and trained from scratch based on our own dataset. It was compared with BERT, a transformer model for understanding the quality of the performances of separate architectures. It is encouraged that future research uses more advanced pre-trained large language models.

7.1.2 External Validity

One external threat is that it is not clear whether our findings still hold on other Q&A websites. it was needed to conduct several qualitative analyses in the Research Questions; however, it is impossible to manually study all instances. To minimize the bias when conducting our qualitative analysis, statistically representative random samples of all relevant revisions were taken, in order to ensure a 95% confidence level and 5% confidence interval for observations.

7.2 Contributions

This work is the first large-scale study [1] of information highlighting in the text description of Stack Overflow answers. it was considered the five most commonly used information highlighting types, which are *Bold*, *Italic*, *Code*, *Delete*, and *Heading*, to understand their characteristics and what information is highlighted in the text description of SO answers with them. Information highlighting is prevalent, with 47.6% of the 52,166,061 studied answers having text highlighted. A terminology to categorize highlighted text in SO answers was proposed and it was found that source code content (e.g., identifiers and programming keywords) are frequently highlighted using *Code* and other highlighting formats (i.e., *Bold* and *Italic*). Besides code, users also tend to highlight updates (e.g., updates of answers), caveats (i.e., a reminder or warning of in which context or condition the provided solution works or does not work), and references. Based on these findings, I studied the prospect of automating the highlighting text containing expected information using deep learning. Named Entity Recognition(NER) model was used for each type of tag. The results show that our CNN models achieve a precision ranging from 0.71 to 0.82 for different formatting types although suffer low recall. Even though BERT showed better precisions, it suffered from even lower recall. The Code model by spaCy achieves good performance with an F1 score of 0.75, outperforming other models.

7.3 Future Direction

Along with the qualitative analysis of information highlighting in Stack Overflow data, this work investigates the potential of recommending the content to be highlighted automatically by adopting named entity recognition (NER) models. For automating the highlighting process, apart from the code models, the text formatting type models suffered from low recall. After analyzing the successful and failure cases, it was derived that low recall occurs because the bold, italic, and heading models fail to highlight words that were originally highlighted. Further looking into the training dataset, it was concluded that it happened because of low frequency of some highlighted words in the datasets. Thus, future work involves finding solution for this issue. A potential solution for this work can be data augmentation and studying the similar highlighting in other websites. This study included two types of deep learning models. More relevant methods and deep learning models can be studied to improve the results. The results of this work are beneficial to the Stack Overflow community and the users. So future work is to incorporate the recommendations with Stack Overflow website and it can suggest to the users which text to highlight with appropriate tags. Further studies could put more effort into investigating how to use the highlighted content for downstream tasks utilizing the information from Stack Overflow. One of the possible tasks can be to consider the highlighted text and develop an answer summarizing tool based on that.

Bibliography

[1] S. S. Ahmed, S. Wang, H. Zhang, T.-H. Chen, and Y. Tian. A first look at information highlighting in stack overflow answers. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 369–373. IEEE, 2022.

[2] T. R. Beelders and J.-P. L. du Plessis. Syntax highlighting as an influencing factor when reading and comprehending source code. *Journal of Eye Movement Research*, 9(1), 2016.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] R. Escobar, J. P. S. Alcocer, H. Tarner, F. Beck, and A. Bergel. Spike–a code editor plugin highlighting fine-grained changes. In *2022 Working Conference on Software Visualization (VISSOFT)*, pages 167–171. IEEE, 2022.

[5] A. Esuli and F. Sebastiani. Evaluating information extraction. In *Multilingual and Multimodal Information Access Evaluation: International Conference of the Cross-Language Evaluation Forum, CLEF 2010, Padua, Italy, September 20-23, 2010. Proceedings 1*, pages 100–111. Springer, 2010.

[6] H. Face. BERT. `https://huggingface.co/docs/transformers/model_doc/bert`, 2023. Accessed: 2023-08-23.

[7] S. Gottipati, D. Lo, and J. Jiang. Finding relevant answers in software forums.

In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pages 323–332. IEEE, 2011.

[8] C. Hannebauer, M. Hesenius, and V. Gruhn. Does syntax highlighting help programming novices? *Empirical Software Engineering*, 23:2795–2828, 2018.

[9] J. He, B. Xu, Z. Yang, D. Han, C. Yang, and D. Lo. Ptm4tag: Sharpening tag recommendation of stack overflow posts with pre-trained models. *2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC)*, pages 1–11, 2022.

[10] R. Jiang, R. E. Banchs, and H. Li. Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*, pages 21–27, 2016.

[11] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel. Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*, 2022.

[12] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. Neural architectures for named entity recognition, 2016.

[13] N. Le Guillarme and W. Thuiller. Taxonerd: deep neural models for the recognition of taxonomic entities in the ecological and evolutionary literature. *Methods in Ecology and Evolution*, 13(3):625–641, 2022.

[14] H. Li, S. Li, J. Sun, Z. Xing, X. Peng, M. Liu, and X. Zhao. Improving api caveats accessibility by mining api caveats knowledge graph. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 183–193. IEEE, 2018.

[15] J. Li, Z. Xing, and A. Sun. Linklive: discovering web learning resources for developers from q&a discussions. *World Wide Web*, 22:1699–1725, 2018.

[16] J. Li, Z. Xing, D. Ye, and X. Zhao. From discussion to wisdom: Web resource recommendation for hyperlinks in stack overflow. In *Proceedings of the 31st*

Annual ACM Symposium on Applied Computing, SAC '16, page 1127–1133, New York, NY, USA, 2016. Association for Computing Machinery.

[17] S. K. Maity, A. Panigrahi, S. Ghosh, A. Banerjee, P. Goyal, and A. Mukherjee. Deeptagrec: A content-cum-user based tag recommendation framework for stack overflow. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, editors, *Advances in Information Retrieval*, pages 125–131, Cham, 2019. Springer International Publishing.

[18] F. Mireshghallah, A. Uniyal, T. Wang, D. Evans, and T. Berg-Kirkpatrick. Memorization in nlp fine-tuning methods. *arXiv preprint arXiv:2205.12506*, 2022.

[19] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[20] S. Nadi and C. Treude. Essential sentences for navigating stack overflow answers. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 229–239. IEEE, 2020.

[21] A. T. Nguyen, B. C. Wallace, and M. Lease. Combining crowd and expert labels using decision theoretic active learning. In *Third AAAI conference on human computation and crowdsourcing*, 2015.

[22] M. E. Palma, P. Salza, and H. C. Gall. On-the-fly syntax highlighting using neural networks. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 269–280, 2022.

[23] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto. Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, 47(3):560–581, 2019.

[24] J. Ramírez, M. Baez, F. Casati, and B. Benatallah. Understanding the impact of text highlighting in crowdsourcing tasks. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 144–152, 2019.

[25] V. S. Rekha, N. Divya, and P. S. Bagavathi. A hybrid auto-tagging system for stackoverflow forum questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, ICONIAAC '14, New York, NY, USA, 2014. Association for Computing Machinery.

[26] A. Sarkar. The impact of syntax colouring on program comprehension. In *PPIG*, page 8, 2015.

[27] C. B. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering*, 25(4):557–572, 1999.

[28] S. L. Smith, P. Kindermans, and Q. V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017.

[29] A. Soni and S. Nadi. Analyzing comment-induced updates on stack overflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 220–224, 2019.

[30] SPACY. Linguistic Features. `https://spacy.io/usage/linguistic-features#named-entities`, 2023. Accessed: 2023-01-30.

[31] SpaCy. Model architectures, 2023. Accessed: 2023-01-30.

[32] StackExchange. How do I format my posts using Markdown or HTML? `https://meta.stackexchange.com/help/formatting`.

[33] H. Strobelt, D. Oelke, B. C. Kwon, T. Schreck, and H. Pfister. Guidelines for effective usage of text highlighting techniques. *IEEE transactions on visualization and computer graphics*, 22(1):489–498, 2015.

[34] L. Tanabe, N. Xie, L. H. Thom, W. Matten, and W. J. Wilbur. Genetag: a tagged corpus for gene/protein named entity recognition. *BMC bioinformatics*, 6:1–7, 2005.

[35] Thinc. Schedules · thinc · a refreshing functional take on deep learning, 2023. Accessed: 2023-01-30.

[36] C. Treude and M. P. Robillard. Augmenting api documentation with insights from stack overflow. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 392–403. IEEE, 2016.

[37] A. J. Viera, J. M. Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5):360–363, 2005.

[38] H. Wang, B. Wang, C. Li, L. Xu, J. He, and M. Yang. Sotagrec: A combined tag recommendation approach for stack overflow. In *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, ICMAI 2019, page 146–152, New York, NY, USA, 2019. Association for Computing Machinery.

[39] L. Wang, L. Zhang, and J. Jiang. Iea: an answerer recommendation approach on stack overflow. *Science China Information Sciences*, 62(11):212103, Sep 2019.

[40] S. Wang, D. Lo, B. Vasilescu, and A. Serebrenik. Entagrec++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, 23:800–832, 2018.

[41] S. Wilson, F. Schaub, R. Ramanath, N. Sadeh, F. Liu, N. A. Smith, and F. Liu. Crowdsourcing annotations for websites' privacy policies: Can it really work? In *Proceedings of the 25th International Conference on World Wide Web*, pages 133–143, 2016.

[42] J.-H. Wu and Y. Yuan. Improving searching and reading performance: the effect of highlighting and text color coding. *Information & Management*, 40(7):617–637, 2003.

[43] B. Xu, Z. Xing, X. Xia, and D. Lo. Answerbot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 706–716. IEEE, 2017.

[44] H. Yin, Z. Sun, Y. Sun, and W. Jiao. A question-driven source code recommendation service based on stack overflow. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642-939X, pages 358–359, 2019.

[45] H. Zhang, S. Wang, T.-H. Chen, and A. E. Hassan. Are comments on stack overflow well organized for easy retrieval by developers? *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(2):1–31, 2021.

[46] H. Zhang, S. Wang, T.-H. Chen, and A. E. Hassan. Reading answers on stack overflow: Not enough! *IEEE Transactions on Software Engineering*, 47(11):2520–2533, 2021.

[47] H. Zhang, S. Wang, T.-H. Chen, Y. Zou, and A. E. Hassan. An empirical study of obsolete answers on stack overflow. *IEEE Transactions on Software Engineering*, 47(4):850–862, 2019.

[48] H. Zhang, S. Wang, T.-H. Chen, Y. Zou, and A. E. Hassan. An empirical study of obsolete answers on stack overflow. *IEEE Transactions on Software Engineering*, 47(4):850–862, 2021.