

# Studying and Recommending Information Highlighting in Stack Overflow Answers

Shahla Shaan Ahmed<sup>a</sup>, Shaowei Wang<sup>a</sup>, Yuan Tian<sup>b</sup>, Tse-Hsun (Peter)  
Chen<sup>c</sup>, Haoxiang Zhang<sup>d</sup>

<sup>a</sup>*Department of Computer Science, University of Manitoba, Canada*

<sup>b</sup>*School of Computing, Queen's University, Canada*

<sup>c</sup>*Department of Computer Science and Software Engineering, Concordia  
University, Canada*

<sup>d</sup>*Huawei, Canada, Canada*

---

## Abstract

**Context:** Navigating the knowledge of 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. Nonetheless, there have been limited studies on the highlighted information.

**Objective:** We carried out the first large-scale exploratory study on the information highlighted in SO answers in our recent study. To extend our previous study, we develop approaches to automatically recommend highlighted content with formatting styles using neural network architectures initially designed for the Named Entity Recognition task.

**Method:** In this paper, we studied 31,169,429 answers of Stack Overflow. For training recommendation models, we choose CNN-based and BERT-based models for each type of formatting (i.e., Bold, Italic, Code, and Heading) using the information highlighting dataset we collected from SO answers.

**Results:** Our models achieve a precision ranging from 0.50 to 0.72 for different formatting types. It is easier to build a model to recommend

---

*Email addresses:* ahmeds27@myumanitoba.ca (Shahla Shaan Ahmed), shaowei.wang@umanitoba.ca (Shaowei Wang), y.tian@queensu.ca (Yuan Tian), peterc@encs.concordia.ca (Tse-Hsun (Peter) Chen), haoxiang.zhang@acm.org (Haoxiang Zhang)

Code than other types. Models for text formatting types (i.e., Heading, Bold, and Italic) suffer low recall. Our 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 (i.g., long-tail knowledge). **Conclusion:** Our findings suggest that it is possible to develop recommendation models for highlighting information for answers with different formatting styles on Stack Overflow.

*Keywords:*

Stack Overflow, Information highlighting, Named entity recognition, Deep learning

---

## 1. Introduction

Technical question and answer (Q&A) sites such as Stack Overflow (SO) have become increasingly important for software developers to share knowledge and contribute to communities. Despite Stack Overflow’s success and prevalence, navigating the knowledge on it remains challenging [12, 26, 51, 56]. The previous study shows that finding answers in long posts remains one of the challenges [26, 51]. 37% of all questions on Stack Overflow have more than one answer, and the average length of an answer is 789 characters [26]. To make the posts vivid to users, Stack Overflow platform allows users to edit their posts with Markdown and HTML [29, 41], 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.

Information highlighting has demonstrated its effectiveness in various domains [27, 34, 42, 49, 50], including in Software Engineering [37] (e.g., it saves the reading time of humans and helps programming novices with code comprehension with syntax highlighting). 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 important to developers.

**Previous study:** In our previous study [2], we performed the first large-

scale study on 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<sup>1</sup>. Our 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 31,169,429) 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, as well as 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.

**Extention:** Since information highlighting is prevalent on SO, however, identifying the appropriate content to highlight can be a challenging task for SO users (typically for new users), as they must properly pinpoint the focal points and select an appropriate formatting style, which can be time-consuming and require expertise. 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. Therefore, in this paper, we extend our recent study [2] by investigating the potential of recommending highlighted content automatically. To do so, we adapt two models (i.e., CNN and BERT) to automatically identify highlighted content for each type of formatting. More specially, we train NER models on each formatting (i.e., Heading, Bold, Italic, and Code) type and recommend the content to highlight. CNN-based models outperform BERT-based models. CNN models achieve precision values ranging from 0.50 to 0.72. The CNN models for automatic source code content highlighting achieve a recall score of 0.66 and an F1 score of 0.69, outperforming the models for other formatting types. In addition, CNN-based models outperform BERT-based models. By analyzing the failure cases, we observed that the majority of the failure cases are due to missing identification (i.e., the model misses the content that is supposed

---

<sup>1</sup>Note that an SO answer may contain both text and code block, we focus on the information highlighting in text description.

to be highlighted). Our analysis shows that the models tend to learn the frequent highlighted words while struggling to learn less frequent words (i.g., long-tail knowledge). We make our replication package public at [https://github.com/shaoweiwang2010/REP\\_2022\\_Information\\_highlight\\_S0](https://github.com/shaoweiwang2010/REP_2022_Information_highlight_S0).

**Paper Organization.** Section 2 presents the background and motivation of the work. Section 3 explains our research questions and how we prepare our data to answer research questions. Section 4 introduces the methodology of our study. In Section 5, we present the results of our research questions. Section 6 discusses the implications of our study and threats to validity. Section 7 presents the related work. Finally, Section 8 concludes our study.

## 2. Background

### 2.1. Background

Stack Overflow allows users to use Markdown and HTML to write and edit posts [29, 41]. Certain formatting types are used to highlight the information in text description with Markdown and HTML tags. In this study, we focus on the five most commonly used formatting types, which are *Bold*, *Italic*, *Delete*, *Code*, and *Heading*. In Table 1, we present the HTML tags and their equivalent Markdown syntax for each formatting type. We group HTML tags and Markdown formatting based on their rendering effect. For example, we group `<b>` and `<strong>` as *Bold* since they render the same effect in the browser when users read the posts on Stack Overflow. In this study, we consider 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<sup>2</sup>.

In the rest of this paper, we refer to the formatting types *Bold*, *Italic*, *Heading*, and *Delete* as *Text* formatting for simplicity’s sake. We use *Code* and *Code* formatting, *Text* and *Text* formatting exchangeably.

### 2.2. Motivation

As discussed in Section 2, Stack Overflow allows users to apply different types of formatting on content to highlight the information. By comprehending the usage practices of those formatting types, we can provide insights into what information is important from users’ perspectives and provide insights

---

<sup>2</sup><https://stackoverflow.com/questions/32402475>



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

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.

| Type    | HTML tags                                                                                                                                                       | Equivalent Markdown                                                                                                                                                                                 |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Code    | <code>&lt;code&gt;</code>                                                                                                                                       | 'example'                                                                                                                                                                                           |
| Bold    | <code>&lt;b&gt;</code> , <code>&lt;strong&gt;</code>                                                                                                            | <b>**example**</b> , <b>__example__</b>                                                                                                                                                             |
| Italic  | <code>&lt;i&gt;</code> , <code>&lt;em&gt;</code>                                                                                                                | <i>*example*</i>                                                                                                                                                                                    |
| Delete  | <code>&lt;del&gt;</code> , <code>&lt;s&gt;</code>                                                                                                               | None                                                                                                                                                                                                |
| Heading | <code>&lt;h1&gt;</code> , <code>&lt;h2&gt;</code> ,<br><code>&lt;h3&gt;</code> , <code>&lt;h4&gt;</code> ,<br><code>&lt;h5&gt;</code> , <code>&lt;h6&gt;</code> | <code># example</code> , <code>## example</code> , <code>### ex-</code><br><code>ample</code> , <code>#### example</code> , <code>#####</code><br><code>example</code> , <code>##### example</code> |

for future research. For instance, we can provide insight into the downstream research that leverages SO information to facilitate software engineering tasks (e.g., API documentation enrichment[20, 44]). Furthermore, identifying the appropriate content to highlight can be a challenging task for SO users, as they must properly pinpoint the focal points and select an appropriate formatting style, which can be time-consuming and require expertise. For instance, we observe that, in many SO answer posts, certain contents were not initially highlighted in the first version of the post but were highlighted in subsequent versions, such as the second or third. For instance, as an example shown in Figure 2, in the second and third revisions of a post, a user highlighted the code-related content (e.g., “Integer”, “Exception”) to improve the presentation of the post. In addition, we observed (see details in Sections 5.2 and 5.3). Therefore, in this study, we also aim to investigate the feasibility of building automatic approaches to recommend content to be highlighted

The first line `declare` a variable named `num`, but, it does not contain a primitive value. Instead it contains a pointer (because the type is `Integer` which is a reference type). Since you did not say as yet what to point to Java sets it to null, meaning "I am pointing at nothing".

In the second line the `new` keyword is used to instantiate (or create) an object of type `Integer` and the pointer variable `num` is assigned this object. You can now reference the object using the dereferencing operator `.` (a dot).

The `Exception` that you asked about occurs when you declare a variable but did not create an object. If you attempt to dereference `num` BEFORE creating the object you get a `NullPointerException`. In the most trivial cases the compiler will catch the problem and let you know that "num may not have been initialized" but sometime you write code that does not directly create the object.

The first line `declares` a variable named `num`, but, it does not contain a primitive value. Instead it contains a pointer (because the type is `Integer` which is a reference type). Since you did not say as yet what to point to Java sets it to null, meaning "I am pointing at nothing".

In the second line the `new` keyword is used to instantiate (or create) an object of type `Integer` and the pointer variable `num` is assigned this object. You can now reference the object using the dereferencing operator `.` (a dot).

The `Exception` that you asked about occurs when you declare a variable but did not create an object. If you attempt to dereference `num` BEFORE creating the object you get a `NullPointerException`. In the most trivial cases the compiler will catch the problem and let you know that "num may not have been initialized" but sometime you write code that does not directly create the object.

Figure 2: An example of revising highlight to improve the presentation of the post.

with formatting types.

### 3. Experimental settings

#### 3.1. Research questions

This study aims to understand how information highlighting is used in SO answers and what content is highlighted using different formatting types. Hence, we formulate our study by answering the following three research questions:

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

In RQ1, we aim to understand the prevalence of the usage of the studied highlighting formatting and their characteristics. We 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. We study

the information highlighted by different formatting types. In RQ4, we explore the feasibility of utilizing machine learning models to recommend which information to highlight.

### 3.2. Data preparation

To answer the four RQs, we downloaded a data dump of Stack Overflow from the Stack Exchange data dump dated March 2021<sup>3</sup>. The data dump contains details information about posts (i.e., questions and answers), as well as their revision history. In this study, we include all the answers and we ended up with 31,169,429 answer posts.

As the dataset was large for processing, we imported it into MySQL. For each answer post, we first extract the textual content from its body and exclude code block(s). Note that we extracted code block(s) using tag “<pre>”. Next, we apply the regular expressions defined in Table 1 to identify information highlighting in each answer post, using the Python library *Regex*. In the end, we extract the content that is highlighted by the studied formatting types.

Table 2 presents the basic statistics of our dataset for different formatting types.

## 4. Methodology

Figure 3 presents an overview of the methodology for our study. Particularly, our study could be decomposed into two parts. In part 1, we perform an empirical study on the information highlighting of SO posts. In part 2, we investigate the feasibility of building a recommendation system for information highlighting. We elaborate on the details of the approach to answering RQs below.

### 4.1. Approach for RQ1

To understand the prevalence of information highlighting in SO answers, we compute 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. In addition, we compute the basic statistics of information highlighting instances to understand the characteristics of each formatting

---

<sup>3</sup><https://archive.org/details/stackexchange>

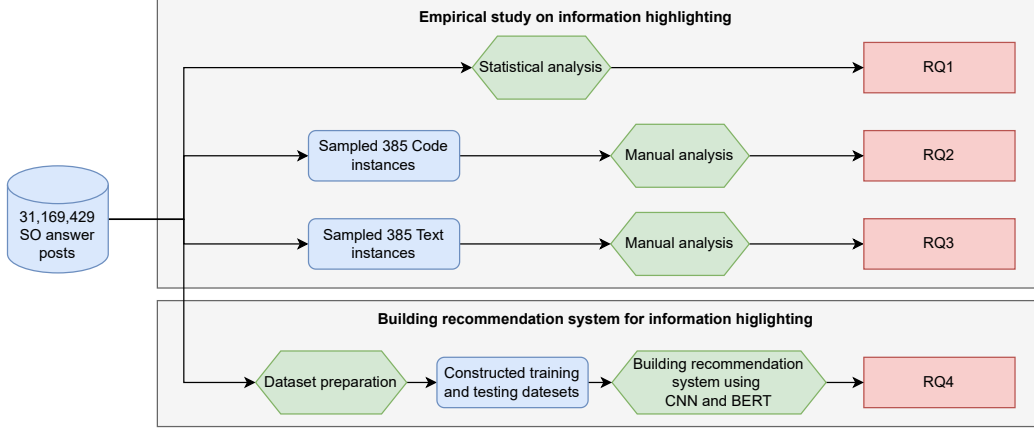


Figure 3: Framework of our study.

type. More specifically, we calculate the distribution of instances for each formatting and the number of words highlighted with them.

#### 4.2. Approach for RQ2

In this RQ, we aim to understand what content is highlighted with the *Code*. First, we randomly sampled 385 *Code* instances with a 5% interval and 95% confidence level. Since there is no existing terminologies we could reuse for this purpose, we manually performed a lightweight open coding-like process [38, 55] to identify the type of content highlighted with *Code*. The process involves three phases and is performed by the first authors (A1 and A2) of the paper. 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 [45]. Table 5 presents the definition and the corresponding example of the derived types for *Code*.

#### 4.3. Approach for RQ3

In this RQ, we aim to understand what content is highlighted with the studied *Text* formatting (i.e., *Bold*, *Italic*, *Delete*, and *Heading*). Similar

to RQ2, we performed an manual study. We randomly sampled 385 *Text* instances with a 5% interval and 95% confidence level. We 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 [45]. Table 6 presents the definition and the corresponding example of the derived types for *Text* formatting.

#### 4.4. Approach for RQ4

In the previous RQs, we focused on analyzing the highlighted content on Stack Overflow and we noticed that users misuse formatting types. For instance, users used *Bold* and *Italic* to highlight code. Therefore, we aim to explore the feasibility of implementing an automated machine learning-based method for content highlighting. 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, we aim to create a model that can highlight information in an input answer post using appropriate formatting types (*Bold*, *Italic*, *Delete*, *Code*, and *Heading*). To achieve this, we use neural network architectures initially designed for Named Entity Recognition (NER) without actually training NER models. Our task shares similarities with NER, such as the short length of labeled (highlighted) tokens and a limited number of text labeling (formatting) options. Moreover, both NER and automated information highlighting can be formed as a sequence labeling task. Note that we train the neural networks from scratch on our information highlighting dataset.

In this RQ, we have used two different structured models to investigate. One model is based on the Convolutional Neural Network (CNN) model and another is based on the transformer model, BERT [8]. We selected CNN because it has been widely used in NER tasks to capture the context information for predicting NER labels and achieved good performance [7, 58, 16]. Compared to Recurrent Neural Network-based approaches (RNN), CNN-based approaches are more efficient. In recent years, the emergence of pre-trained models, such as BERT [8], T5 [32], GPT [5] have re-evolute the landscape of the natural language processing (NLP) domain and demonstrated their effectiveness in NLP tasks, including NER tasks. Pre-trained models encapsulate knowledge gleaned from a massive amount of training data, bypassing the

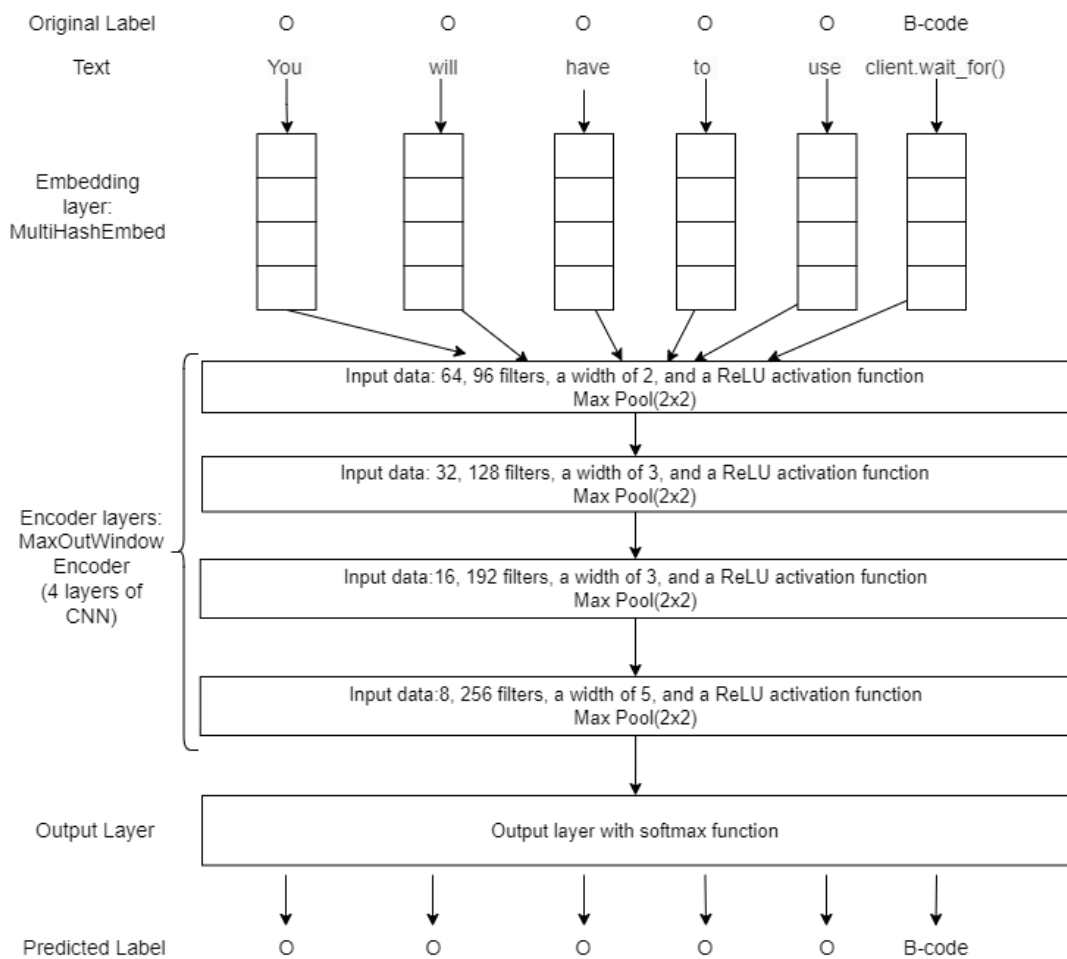


Figure 4: Layers of the CNN model

need for training from scratch, and could be easily adapted to downstream tasks by fine-tuning. BERT is an encoder-based model and considers the contexts of both preceding and subsequent words. BERT has been proven effective in embedding the context for NER task [39, 6, 13]. Therefore, we also investigate BERT in this RQ.

Figure 4 presents the architecture of our CNN model. In this paper, we use the lib spaCy [40] as the implementation of CNN model. We opt to utilize spaCy since it has emerged as the de facto standard for practical NLP due to its speed, robustness, and performance [19]. 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*. It features an embedding layer that embeds a number of lexical attributes separately. Then the results are concatenated as a vector semantic representation of a token for passing through a feed-forward network. 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.

Figure 5 presents the architecture of BERT model, which consists of multiple transformer blocks, each containing a self-attention mechanism for getting the context of the input sequence. In this work, we fine-tuned BERT for our prediction task. We used the most recent checkpoint of BERT model from HuggingFace platform [11].

As our HTML highlighting tags were categorized into five distinct types (see Section 2.1 for details), we initially attempted to train a universal model that would accommodate all five types. However, this approach yielded unsatisfactory results, as the model tended to highlight words as code by default while ignoring other formatting types. Consequently, we trained separate models for each highlighting type. Our findings in RQ1 showed that the Delete type was scarcely used in comparison to the other types, leading us to omit it from our analysis in RQ4. Therefore, we train four models, i.e., *Model<sub>Bold</sub>*, *Model<sub>Italic</sub>*, *Model<sub>Heading</sub>*, and *Model<sub>Code</sub>* with each of CNN model and BERT models.

Dataset preparation: Prior to constructing the training and testing datasets, we carefully cleaned the dataset to eliminate misuse instances (e.g., using

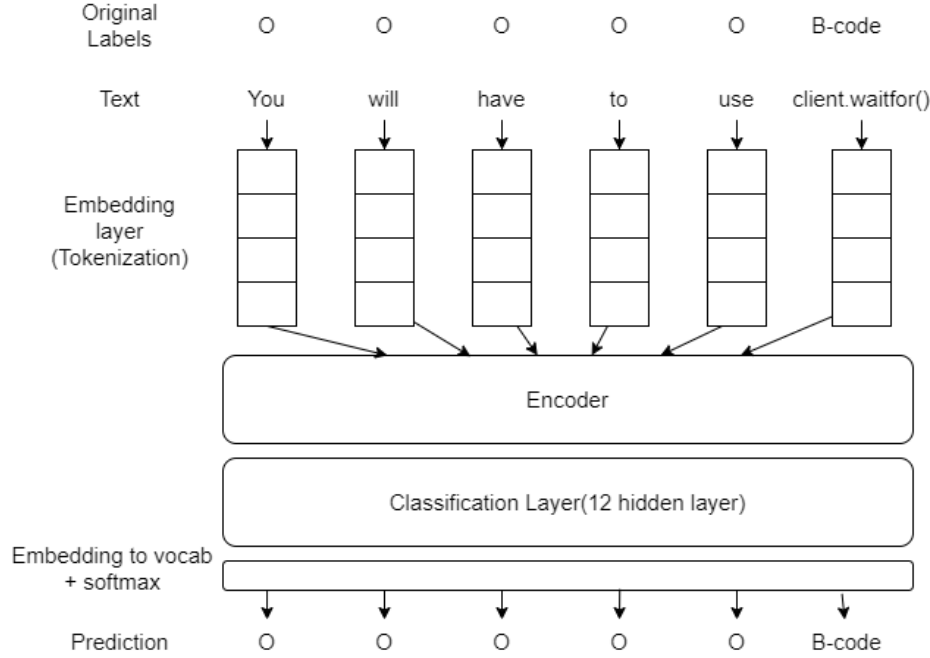


Figure 5: Layers of the NER BERT model

bold formatting to highlight code) to ensure that the dataset used for model training would accurately represent the intended formatting of information. To achieve this, we implemented the following pre-processing steps to clean the data.

- **Cleaning up code dataset.** According to Stack Overflow guidelines, users are encouraged to highlight code content using the *code* tag exclusively. However, our analysis of RQ3 results revealed instances where *code* tag was improperly used for Path, software, Terminology, and Equation. Therefore, our first task is to remove such cases from our code dataset. We used regular expressions (e.g.,  $\sim ([^/ ]^*)+/?\$$ ) to identify all paths and equations in content highlighted with *code*. For identifying software and terminologies, we collected all SO tags (e.g., “mysql”, “andriod”, “django”, and “hashtable”)<sup>4</sup> as our comprehensive dictionary. We then used fuzzy matching to identify such software and terminologies in the code dataset based on the dictionary. We used

<sup>4</sup><https://stackoverflow.com/tags>

*fuzzywuzzy* library<sup>5</sup> for our implementation. The reason we selected SO tags as our dictionary is that they contain a diverse and wide range of software names and technologies [53]. we discarded those cases from our code dataset.

- **Cleaning up text dataset.** Users sometimes highlight code data with text formatting tags. As it goes against the SO guidelines, we need to identify the code that is highlighted with text formatting tags (i.e., *Bold*, *Italic*, *Heading*). To identify the code content in text dataset, we cross-checked the code dataset and text dataset and examined if any content was cross-highlighted in both code and text formatting. We removed such instances from text data. For instance, if we find “`BufferedReader.close()`” are both highlighted with Code and Bold formatting, we only keep the instances in code data and remove the instances from text data.
- **Removing other HTML tags.** We used regular expressions to remove other HTML tags rather than the targeting tags. For example, for the *Bold* model, we only kept the `<b>` and `<strong>` tags to identify the bold highlighting in sentences and removed other highlighting tags.

Now we can construct our training and testing set. We divided the post into sentences and extracted the corresponding formatting tags described in Section 3. 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, we consider the content highlighted with different formatting types as the entities in the NER model. We annotate the tagged words with their formatting tags, considering their position in the phrase. For example, in the sample highlighted sentence shown in Figure 4, i.e., “You will have to use `client_wait_for()`.”, “`client_wait_for()`” is formatted with a code tag. To label this sentence, we assign the label “B-code” 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, we used I (Interior) and E (Ending) to label the highlighted content’s internal words and the ending word. For example,

---

<sup>5</sup><https://pypi.org/project/fuzzywuzzy/>

“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 we train separate models for each highlighting type. When preparing the dataset for a particular highlighting type, if a sentence has other highlighting types rather than the one we target, we omit other types of highlighting types from the sentence.

Table 2: Number of sentences in train and test datasets.

| Type    | Training set | Testing set |
|---------|--------------|-------------|
| Bold    | 521308       | 140539      |
| Italic  | 477690       | 104128      |
| Code    | 1410890      | 406960      |
| Heading | 131976       | 21028       |

Following prior studies, we partitioned the dataset into an 80:20 ratio for training and testing purposes [31, 3, 52]. We trained our models in 3 epochs where the learning rate is 0.001. We employ two training settings.

We use 32 as the size of the batch in the fixed-size setting (i.e., *fixed*). Models based on BERT architecture are trained in 0.05 learning rate using 16 batches(fixed) in 3 epochs.

Table 3: Models parameters.

| Parameter     | CNN-based | BERT-based |
|---------------|-----------|------------|
| Epochs        | 3         | 3          |
| Learning rate | 0.001     | 0.05       |
| Batch size    | 32        | 16         |

We ran our experiments on a Linux server with four Nvidia RTX 3090 GPUs, AMD Ryzen 48-Core CPU with 256 GB Ram. For simplicity, we call the models that are trained using the incremental size of batches *incremental models* and those using the fixed size of batches *fixed models*.

Evaluation: We use precision, recall, and F1-score to measure the performance of models, following prior studies evaluating NER models [10, 17]. 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),

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                           |

we decide to use a partial match [25, 43], in which we consider 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 specifically, we calculate the precision and recall as follows:

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

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

For instance, 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.

## 5. Results

### 5.1. RQ1: How prevalent is the information highlighting in SO answers?

**Results: Overall, 47.6% (14,845,929 out of 31,169,429) 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. We observe 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 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. *RQ2: What types of information are highlighted with Code formatting in SO answers?*

**Results:** *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. We also observe 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, we observe 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*<sup>6</sup>. *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<sup>7</sup>.

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%)      |

<sup>6</sup><https://stackoverflow.com/questions/17117375>

<sup>7</sup><https://stackoverflow.com/questions/24586043>

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%).

### 5.3. RQ3: What types of information are highlighted with Bold, Italic, Delete, and Heading formatting in SO answers?

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., “ <b>Update:</b> According to your comments, I think you will want to try <code>gzfile()</code> in <code>read.table()</code> ”.         | 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 <b>startupOrder</b> 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., “ <b>It won’t work in that shell though, you need to open a new one</b> ”. | 33 (18.6%) | 43 (25.4%) | 0        | 1 (2.8%)   |
| Reference   | Reference to internal or external resource, e.g., “<a href=“http://api.jquery.com/not/”>.not(</a>”.                                                                                | 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., “ <b>WORKING EXAMPLE</b> ”.                                                                                                                          | 27 (15.3%) | 2 (1.2%)   | 0        | 33 (91.6%) |
| Options     | Options in software, e.g., “you have to enable Less <b>Secure Sign-In</b> 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 <i>all</i> 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 <b>KVQ</b> KVC”.                                                                                                               | 0          | 0          | 3 (100%) | 0          |
| Other       | Other than the above defined types.                                                                                                                                                | 6 (3.4%)   | 10 (5.9%)  | 0        | 1          |

**Results:** 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 [41]. Interestingly, we also observe 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 function “strncmp()” from *Italic* to *Code*<sup>8</sup>.

**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 jmp.dll files (that comes up with jmp download) as a native library for more info see my answer on ...” using *Bold*.<sup>9</sup> *Bold* and *Italic* are used to highlight Reference and Terminology, although *Italic* is used 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 [33, 55]. 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 the 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.

---

<sup>8</sup><https://stackoverflow.com/posts/18437465/revisions>

<sup>9</sup><https://stackoverflow.com/questions/6498179>

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

5.4. *RQ4: Can we recommend the highlighted content in SO answers automatically?*

**Out of the four trained models, the Code model obtained the highest F1 score of 0.69 (CNN).** Table 7 presents the results of our trained models in terms of precision, recall, and F1 using CNN and BERT sizes of batches. In general, in terms of F1, we observe that the Code models (F1 score is 0.162 and 0.69 using BERT and CNN models) perform better than other models. This is reasonable, since first for code formatting, we have 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. Our observation indicates that it is easier to build a model to identify code and highlight them on SO posts than other types of formatting.

We also notice that for code data, CNN-based models outperform BERT-based models by a large margin, typically in terms of recall. The CNN-based model for code achieves a recall of 0.66, while the BERT-based model only has a recall of 0.093. In other words, the CNN-based model recognizes and covers more real code-related content than the BERT-based model. One possible explanation is that the CNN-based model is smaller than BERT and makes it easier to train and learn the patterns to recognize the code-related content to highlight. Another possibility is that BERT is pre-trained on natural language and not for code, which makes it hard to recognize code-related content.

Table 7: The performance of models on all formatting types *Bold*, *Italic*, *Heading*, *Code* with different training settings. *CNN* denotes the CNN-based models and *BERT* denotes the BERT-based models.

| Type                           | Precision<br>( <i>CNN</i> ) | Precision<br>( <i>BERT</i> ) | Recall<br>( <i>CNN</i> ) | Recall<br>( <i>BERT</i> ) | F1<br>( <i>CNN</i> ) | F1<br>( <i>BERT</i> ) |
|--------------------------------|-----------------------------|------------------------------|--------------------------|---------------------------|----------------------|-----------------------|
| <i>Model<sub>Bold</sub></i>    | 0.57                        | 0.63                         | 0.05                     | 0.006                     | 0.09                 | 0.011                 |
| <i>Model<sub>Italic</sub></i>  | 0.64                        | 0.53                         | 0.01                     | 0.0025                    | 0.02                 | 0.005                 |
| <i>Model<sub>Heading</sub></i> | 0.50                        | 0.49                         | 0.04                     | 0.0073                    | 0.08                 | 0.0143                |
| <i>Model<sub>Code</sub></i>    | 0.72                        | 0.65                         | 0.66                     | 0.093                     | 0.69                 | 0.162                 |

Our CNN-based models achieve an acceptable precision ranging from 0.50 to 0.72 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, Bold, and Italic models can achieve at least a precision of 0.57 from the 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). One possible reason is the training data for those three types are much smaller than code and the patterns for text formatting types are more diverse, which makes it challenging to learn models for text formatting.

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

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

In Table 8, we present several successful examples, 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.

Case Study of Failure Cases: We also investigate the failure cases of the models to provide some insights for future research. The failure cases could be categorized into the following three families:

- **Misidentification:** Although the content that needs highlighting 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 be highlighted.
- **Missing Identification:** The model misses the content that is supposed to be highlighted.

We calculate the distribution of three failure cases over each model using the confusion matrix. For instance, to compute the number of missing identification cases from *Model<sub>Bold</sub>*, we count the number of false negative instances in the confusion matrix on the testing data generated by *Model<sub>Bold</sub>*.

Table 9: The distribution of different failure types among CNN models of different formatting types.

| Failure Type           | Formatting Type |        |        |         |
|------------------------|-----------------|--------|--------|---------|
|                        | Bold            | Code   | Italic | Heading |
| Misidentification      | 0.86%           | 4.09%  | 2.65%  | 15.19%  |
| Missing Identification | 95.6%           | 52.77% | 96.67% | 81.1%   |
| False Identification   | 3.55%           | 43.13% | 0.67%  | 3.63%   |

Table 9 presents the distribution of those three types of failures across different models. Overall, the majority of the failure cases are missing identification. All models suffer from high missing identification except Code models, which explains the low recall in Table 7, typically for text formatting tag (Bold, Italic, and Heading) models. 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) [18, 24]. To test this assumption, we count the frequency of the words that are correctly predicted and the words that models in the training data do not identify. We examine the recommendation results produced by each CNN-based model. Table 10 presents the mean/median frequency of words that are correctly predicted and those that are not identified by models. On top of it, Figure 6 compares their distributions. We observe 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.

Table 10: 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     | 127.7/7                               | 441.5/37.0                        |
| Italic   | 2.39/2.52                             | 959.9/332.0                       |
| Heading  | 1.86/1.94                             | 241.24/89.0                       |
| Code 50% | 0.699/0.602                           | 1931.04/8.0                       |

We also observe a small portion of misidentification cases and False identification. For instance, in example 1 in Table 11, the original text is highlighted with Heading, while our 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. We actually observed such a phenomenon in Section 5.3. Users sometimes highlight the same content with both Bold Italic tags. For instance, there are many cases in which if there is any update on the posts, the keyword “update” will be highlighted by different types of formatting, i.e., “update” was highlighted the most by Bold tag (0.028%) followed by other tags (Heading 0.0054%, Code 0.00269% and Italic 0.00249%). Therefore, when recommending the information to highlight, if multiple models recommend the same content (e.g., both the Bold model and Italic model recommend “update” to be highlighted), we use the formatting tag that is returned by the model with the highest confidence as the final tag. This decision is based on the assumption that the recommendation from the model with the highest confidence aligns with the predominant choice of the community. By prioritizing the recommendation from the most confident model, we aim to ensure consistency and

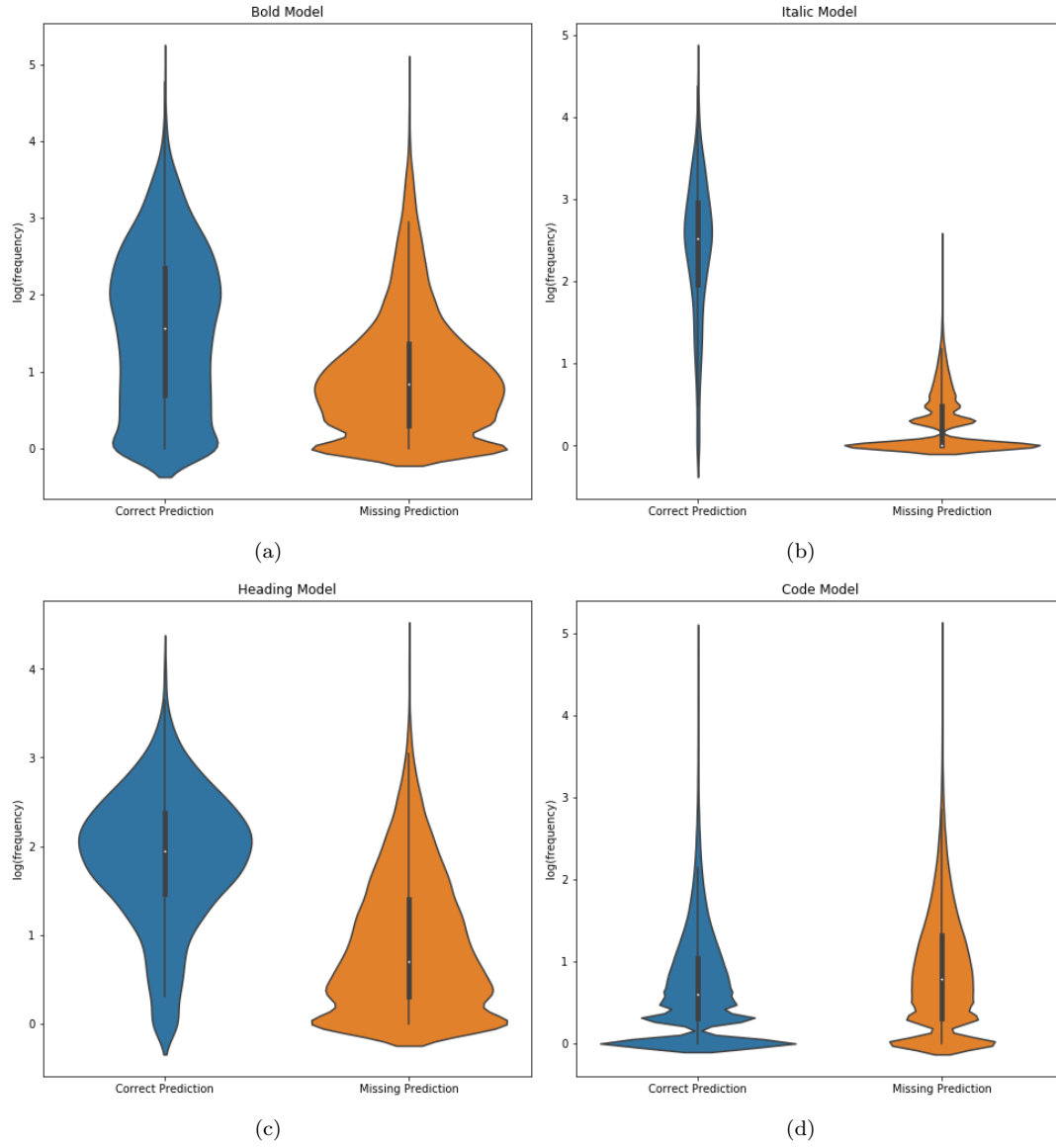


Figure 6: Comparison of the frequency distribution of words that are not identified (missing identification) and those that are correctly predicted (correct prediction) on different models.

accuracy in the formatting of highlighted information, reflecting the collective preferences of the user community. Another option is that we can return all recommendation results with confidence scores returned by each model and allow users to select, like Google Query Suggestion [1].

Table 11: Failure cases predicted by CNN-based models for all the types *Bold*, *Italic*, *Heading*, and *code*.

| ID | Type    | Original text                                                                                                                                                                                                                                                                                                                                                                      | Text highlighted with our models (Incremental).                                                                                                                                                                                                                                                                                                                                    | Failure type                                                                                  |
|----|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| 1  | Bold    | Swift 5.2.x<br>First of all, you need to declare an "easy to use" typealias for your block:                                                                                                                                                                                                                                                                                        | <b>Swift 5.2.x</b><br>First of all, you need to declare an "easy to use" typealias for your block:                                                                                                                                                                                                                                                                                 | <b>Misidentification:</b> 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 <b>WILL THROW THE EXCEPTION!</b>                                                                                                                                                                                                                                                                                                    | <b>False Identification:</b> detected normal texts as bold.                                   |
| 3  | Bold    | <b>From now, your frontend application will use access token in the Authorization header for every request.</b>                                                                                                                                                                                                                                                                    | From now, your frontend application will use access token in the Authorization header for every request.                                                                                                                                                                                                                                                                           | <b>Missing Identification:</b> missed Bold.                                                   |
| 4  | Italic  | Disclaimer: I currently work for Mapbox.                                                                                                                                                                                                                                                                                                                                           | Disclaimer: <i>I currently</i> work for Mapbox.                                                                                                                                                                                                                                                                                                                                    | <b>False Identification:</b> 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                                                                                                                                                                                                                                                                                 | <b>False Identification:</b> 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. | <b>Missing Identification:</b> 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 :                                                                                                                                                                                                                               | <code>setTimeout</code> 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 :                                                                                                                                                                                                                  | <b>False Identification:</b> 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 .                                                                                                                                                                                               | <b>Missing Identification:</b> Missed the Code tag.                                           |

Our models achieve a precision ranging from 0.50 to 0.72 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. Our 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 (i.g., long-tail knowledge).

## 6. Discussion

### 6.1. Usefulness of recommendation models

SO encourages the community to improve the quality of posts by editing them and proper edits will be approved by trusted community members[28]. To demonstrate the usefulness of the proposed recommendation models, we randomly sampled 20 new SO posts that are outside of our dataset, and applied our CNN-based model on them to recommend the information to highlight. We found that in 9 posts, the content recommended by our models was already highlighted. For the rest 11 posts, we highlight the content recommended by the model. All our recommended edits in those 11 posts were accepted by the community. For instance, in Figure 7, our model recommends highlighting the subsection title (e.g., Introduction and The Problem) in *Bold* and highlighting the code with *Code*. We posted the edits and all of our edits were approved by the community<sup>10</sup>.

### 6.2. Implication of our findings

**Future search should consider the highlighted content for the downstream tasks that leverage information from the SO answers.** In RQ3, we observe that users tend to highlight important information for a provided answer using *Text* formatting, such as a reminder or warn of in which context or condition the provided solution works or does not work (i.e., Caveat), an update and new edit in answers (i.e., Update). Such content is informative and important to users when learning and applying knowledge from the answers. Therefore, it is substantial to consider such information for the downstream tasks that leverage information extracted from SO answers,

---

<sup>10</sup><https://stackoverflow.com/posts/76886399/revisions>

#### Introduction:

Are you grappling with a baffling MongoDB connection problem after upgrading to Node.js version 18? Don't fret – there's a quick solution that can save you time and frustration. Resolving MongoDB Connection Issues After Node.js Version 18.

#### The Problem:

After upgrading to Node.js v18, some developers encounter a vexing issue while connecting to their local MongoDB server. Despite seemingly accurate configurations, they face the dreaded ECONNREFUSED error.

#### The Solution:

The key lies in the way you structure your MongoDB connection string. Previously, using localhost was common practice:

```
mongoose.connect('mongodb://localhost:27017/your_database');
```

However, with Node.js v18, opting for the explicit IP address 127.0.0.1 in the connection string does the trick:

```
mongoose.connect('mongodb://127.0.0.1:27017/your_database');
```

#### Conclusion:

When MongoDB connections hiccup post-Node.js v18 upgrade, the remedy is simpler than anticipated. By choosing 127.0.0.1 over localhost in your connection string, you can effortlessly resolve the ECONNREFUSED error and restore seamless MongoDB connectivity.

#### Introduction:

Are you grappling with a baffling MongoDB connection problem after upgrading to Node.js version 18? Don't fret – there's a quick solution that can save you time and frustration. Resolving MongoDB Connection Issues After Node.js Version 18.

#### The Problem:

After upgrading to Node.js v18, some developers encounter a vexing issue while connecting to their local MongoDB server. Despite seemingly accurate configurations, they face the dreaded ECONNREFUSED error.

#### The Solution:

The key lies in the way you structure your MongoDB connection string. Previously, using localhost was common practice:

```
mongoose.connect('mongodb://localhost:27017/your_database');
```

However, with Node.js v18, opting for the explicit IP address 127.0.0.1 in the connection string does the trick:

```
mongoose.connect('mongodb://127.0.0.1:27017/your_database');
```

#### Conclusion:

When MongoDB connections hiccup post-Node.js v18 upgrade, the remedy is simpler than anticipated. By choosing 127.0.0.1 over localhost in your connection string, you can effortlessly resolve the ECONNREFUSED error and restore seamless MongoDB connectivity.

Figure 7: An example of a post where our formatting improvements were approved by the community.

such as answer summarization [36], and API documentation enrichment [20, 44].

**Our findings provide insights to improve future research in developing approaches for automatic information highlight.** In RQ4, we investigated the potential of deep learning NER models in recommending information to highlight on SO posts. Code is easier to identify than other types of formatting. Furthermore, all text-related models (i.e., Italic, Bold, and Heading) suffer from low recall. We reveal one reason is long-tail knowledge. The model tends to memorize frequently highlighted words and miss the recognition of the less-frequent (e.g., tail knowledge). Future research is encouraged to leverage techniques [57], e.g., data augmentation and class rebalancing for tail knowledge.

### 6.3. Threats to validity

**Internal Validity:** Our study involved qualitative analysis in RQs. To reduce the bias, each instance was labeled by two of the authors, 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 RQ4, we evaluate our models using precision, recall, and F-1 score with partial match. Although, there might be other metrics that could be used for

our task, those metrics are commonly used in NER models, and we follow previous studies [10, 17]. In RQ4, we investigated the potential of a CNN-based model and trained it from scratch based on our own dataset. We also investigated a pre-trained model BERT and fine-tuned it for our task. However, our finding might not be generalized to other models. We encourage future research to investigate more models for the task.

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

**Content Validity:** Content validity refers to the extent to which a measure represents all facets of a given construct. In this study, we focus on *Bold*, *Italic*, *Heading*, *Delete*, and *Code*. Stack Overflow also provides other types of formatting to highlight information, such as lists and links. To mitigate the threat, we focus on the most commonly used ones. Future research is encouraged to study more types of formatting.

**Construct Validity:** One threat to the construct validity is related to how we collected instances for training recommendation models for different formatting types. Although we made diligent efforts to clean up the dataset and eliminate inconsistencies from both code and text data, as discussed in Section 4.4, it is important to acknowledge that it may not have been possible to remove all inconsistencies. This could potentially pose a threat to the validity of our results. However, our failure cases analysis in Section 5.4 reveals that the proportion of misidentification is low, consistently less than 15.17% across all models. This suggests that following the data cleaning process, the models are not easily confused among different types of formatting. This provides evidence regarding the effectiveness of our data cleaning efforts and increases the reliability of our results. Another threat to the construct validity relates to the suitability of our evaluation metrics: Precision, Recall, and F1-measure based on partial match. Other metrics could be used for evaluating the recommendation system. We selected those metrics because these metrics have been used in many NER studies [10, 17]. This threat is limited by the research.

## 7. Related work

### 7.1. Information highlighting

Previous research has explored the benefits of highlighting information in various domains over the last decades [27, 34, 42, 49, 50]. Wu and Yuan have shown that highlighting could reduce the cognitive load and thus reduce reading time [50]. Jorge et al. investigated the impact of highlighting text in the text classification tasks in the machine learning area and found that highlighting is effective in reducing classification effort for humans [34]. Similarly, Nguyen et al. developed a tool to explain the output ML models by highlighting the portion of text and demonstrated its effectiveness in model explanation [27]. Various studies have been done to investigate the impact of information highlighting in software engineering, such as code comprehension [4, 14, 30, 37], source code evolution [9]. For instance, Advait Sarkar investigated the effect of syntax highlighting on program comprehension and its interaction with programming experience [37], and found that syntax highlighting significantly improves task completion time. Different from prior studies that focus on understanding the influence of information highlighting, we focus on understanding the practice of information highlighting on SO.

### 7.2. Knowledge retrieval on Stack Overflow

Several studies have investigated developers' challenges and solutions in retrieving knowledge from technical Q&A sites and provided some tools to facilitate the process [12, 26, 51, 56]. For instance, 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 [12], 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 [51]. Zhang et al. pointed out an issue of 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 [56]. Sarah and Treude investigated the possibility of using two existing techniques, wordpartten and lexicrank, to identify the essential sentences from a long SO answer [26]. Different from prior studies that focus on retrieving relevant or important answers from SO, we investigate what

and how information is highlighted on SO. Our study could provide insights to enhance existing techniques.

### 7.3. Recommendation system for Stack Overflow

Previous studies developed different recommendation tools to ease the use of the Q&A sites, such as recommending hyperlinks [21, 22], tags [15, 23, 35, 46, 48], and similar questions [47, 54]. Li et al. developed LinkLiv to recommend hyperlinks [22], which uses multiple features, including hyperlink co-occurrences in Q&A discussions, and locations (e.g., question, answer, or comment). Wang et al. developed EnTagRec, which combines two complementary lines in the statistics community Bayesian and frequentist [48]. Wang et al. developed SOTagRec combining a convolutional neural network and collaborative filtering model to infer tags for new postings by studying the historical postings and their tags [46]. Maity et al. developed Deep-TagRec [23], which learns from the content representation of question title and body, and recommends appropriate question tags on Stack Overflow. Wang et al. developed an approach that combines topical interest, topical expertise, and activeness to recommend answers for new questions [47]. Similar to these studies, we investigate the potential of developing models for recommending information to highlight in SO answers. We adapted Named Entity Recognition (NER) models to identify the content that needs to be highlighted.

## 8. Conclusion

This paper is the first large-scale study of information highlighting in the text description of SO answers. We find that information highlighting is prevalent, with 47.6% of the 31,169,42 studied answers having text highlighted. We propose a terminology to categorize highlighted text in SO answers and find 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. Further studies could put more effort into investigating how to use the highlighted content for downstream tasks (e.g., answer summarizing) that leverage information from SO answers, and provide tools that can suggest highlighted text for SO users. In addition, we also investigate the

potential of recommending the content to be highlighted automatically by adopting named entity recognition (NER) models. Our experimental results highlight that our CNN-based models achieve precision scores between 0.5 and 0.72 across different formatting types, although they exhibit lower recall rates. On the other hand, while BERT demonstrates superior precision, it struggles with even lower recall. It’s worth noting that CNN-based Code model performs exceptionally well, with an impressive F1 score of 0.71, outperforming other models in our study. Although BERT usually has better performance in other tasks, in our case the CNN-based model worked better. In future research, we plan to investigate other more advanced models (e.g., LLaMA) and work on improving the recall rates for both models.

## References

- [1] (2024). Google query suggestion. [https://www.google.com/support/enterprise/static/gsa/docs/admin/current/gsa\\_doc\\_set/xml\\_reference/query\\_suggestion.html](https://www.google.com/support/enterprise/static/gsa/docs/admin/current/gsa_doc_set/xml_reference/query_suggestion.html). Accessed: 2023-04-10.
- [2] Ahmed, S. S., Wang, S., Zhang, H., Chen, T.-H., and Tian, Y. (2022). 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.
- [3] Alrashedy, K., Dharmaretnam, D., German, D. M., Srinivasan, V., and Gulliver, T. A. (2020). Scc++: Predicting the programming language of questions and snippets of stack overflow. *Journal of Systems and Software*, **162**, 110505.
- [4] Beelders, T. R. and du Plessis, J.-P. L. (2016). Syntax highlighting as an influencing factor when reading and comprehending source code. *Journal of Eye Movement Research*, **9**(1).
- [5] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

- [6] Chang, Y., Kong, L., Jia, K., and Meng, Q. (2021). Chinese named entity recognition method based on bert. In *2021 IEEE International Conference on Data Science and Computer Application (ICDSCA)*, pages 294–299. IEEE.
- [7] Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional lstm-cnns. *Transactions of the association for computational linguistics*, **4**, 357–370.
- [8] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.
- [9] Escobar, R., Alcocer, J. P. S., Tarner, H., Beck, F., and Bergel, A. (2022). Spike—a code editor plugin highlighting fine-grained changes. In *2022 Working Conference on Software Visualization (VISSOFT)*, pages 167–171. IEEE.
- [10] Esuli, A. and Sebastiani, F. (2010). 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.
- [11] Face, H. (2023). BERT. [https://huggingface.co/docs/transformers/model\\_doc/bert](https://huggingface.co/docs/transformers/model_doc/bert). Accessed: 2023-08-23.
- [12] Gottipati, S., Lo, D., and Jiang, J. (2011). Finding relevant answers in software forums. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pages 323–332. IEEE.
- [13] Hakala, K. and Pyysalo, S. (2019). Biomedical named entity recognition with multilingual bert. In *Proceedings of the 5th workshop on BioNLP open shared tasks*, pages 56–61.
- [14] Hannebauer, C., Hesenius, M., and Gruhn, V. (2018). Does syntax highlighting help programming novices? *Empirical Software Engineering*, **23**, 2795–2828.
- [15] He, J., Xu, B., Yang, Z., Han, D., Yang, C., and Lo, D. (2022). 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.
- [16] Jehangir, B., Radhakrishnan, S., and Agarwal, R. (2023). A survey on named entity recognition—datasets, tools, and methodologies. *Natural Language Processing Journal*, **3**, 100017.
  - [17] Jiang, R., Banchs, R. E., and Li, H. (2016). Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*, pages 21–27.
  - [18] Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. (2022). Large language models struggle to learn long-tail knowledge. *arXiv preprint arXiv:2211.08411*.
  - [19] Le Guillarme, N. and Thuiller, W. (2022). 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.
  - [20] Li, H., Li, S., Sun, J., Xing, Z., Peng, X., Liu, M., and Zhao, X. (2018a). 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.
  - [21] Li, J., Xing, Z., Ye, D., and Zhao, X. (2016). 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. Association for Computing Machinery.
  - [22] Li, J., Xing, Z., and Sun, A. (2018b). Linklive: discovering web learning resources for developers from q&a discussions. *World Wide Web*, **22**, 1699–1725.
  - [23] Maity, S. K., Panigrahi, A., Ghosh, S., Banerjee, A., Goyal, P., and Mukherjee, A. (2019). 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. Springer International Publishing.

- [24] Miresghallah, F., Uniyal, A., Wang, T., Evans, D., and Berg-Kirkpatrick, T. (2022). Memorization in nlp fine-tuning methods. *arXiv preprint arXiv:2205.12506*.
- [25] Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, **30**(1), 3–26.
- [26] Nadi, S. and Treude, C. (2020). Essential sentences for navigating stack overflow answers. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 229–239. IEEE.
- [27] Nguyen, A. T., Wallace, B. C., and Lease, M. (2015). Combining crowd and expert labels using decision theoretic active learning. In *Third AAAI conference on human computation and crowdsourcing*.
- [28] Overflow, S. (????). Edit a Question or Answer. <https://stackoverflowteams.help/en/articles/8858605-edit-a-question-or-answer>. Accessed: 2024-04-01.
- [29] Overflow, S. (2022). Markdown help. <https://stackoverflow.com/editing-help>. Accessed: 2023-01-30.
- [30] Palma, M. E., Salza, P., and Gall, H. C. (2022). 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.
- [31] Qiao, L., Li, X., Umer, Q., and Guo, P. (2020). Deep learning based software defect prediction. *Neurocomputing*, **385**, 100–110.
- [32] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, **21**(1), 5485–5551.
- [33] Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., and Oliveto, R. (2019). Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, **47**(3), 560–581.
- [34] Ramírez, J., Baez, M., Casati, F., and Benatallah, B. (2019). 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.

- [35] Rekha, V. S., Divya, N., and Bagavathi, P. S. (2014). A hybrid auto-tagging system for stackoverflow forum questions. In *Proceedings of the 2014 International Conference on Interdisciplinary Advances in Applied Computing*, New York, NY, USA. Association for Computing Machinery.
- [36] Ren, X., Xing, Z., Xia, X., Li, G., and Sun, J. (2019). Discovering, explaining and summarizing controversial discussions in community q&a sites. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 151–162. IEEE.
- [37] Sarkar, A. (2015). The impact of syntax colouring on program comprehension. In *PPIG*, page 8.
- [38] Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering*, **25**(4), 557–572.
- [39] Souza, F., Nogueira, R., and Lotufo, R. (2019). Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*.
- [40] SPACY (2023). Linguistic Features. <https://spacy.io/usage/linguistic-features#named-entities>. Accessed: 2023-01-30.
- [41] StackExchange (2023). How do I format my posts using Markdown or HTML? <https://meta.stackexchange.com/help/formatting>. Accessed: 2023-01-30.
- [42] Strobel, H., Oelke, D., Kwon, B. C., Schreck, T., and Pfister, H. (2015). Guidelines for effective usage of text highlighting techniques. *IEEE transactions on visualization and computer graphics*, **22**(1), 489–498.
- [43] Tanabe, L., Xie, N., Thom, L. H., Matten, W., and Wilbur, W. J. (2005). Genetag: a tagged corpus for gene/protein named entity recognition. *BMC bioinformatics*, **6**, 1–7.
- [44] Treude, C. and Robillard, M. P. (2016). Augmenting api documentation with insights from stack overflow. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 392–403. IEEE.

- [45] Viera, A. J., Garrett, J. M., *et al.* (2005). Understanding interobserver agreement: the kappa statistic. *Fam med*, **37**(5), 360–363.
- [46] Wang, H., Wang, B., Li, C., Xu, L., He, J., and Yang, M. (2019a). So-tagrec: A combined tag recommendation approach for stack overflow. In *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, page 146–152. Association for Computing Machinery.
- [47] Wang, L., Zhang, L., and Jiang, J. (2019b). Iea: an answerer recommendation approach on stack overflow. *Science China Information Sciences*, **62**(11), 212103.
- [48] Wang, S., Lo, D., Vasilescu, B., and Serebrenik, A. (2018). Entagrec++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, **23**, 800–832.
- [49] Wilson, S., Schaub, F., Ramanath, R., Sadeh, N., Liu, F., Smith, N. A., and Liu, F. (2016). Crowdsourcing annotations for websites’ privacy policies: Can it really work? In *Proceedings of the 25th International Conference on World Wide Web*, pages 133–143.
- [50] Wu, J.-H. and Yuan, Y. (2003). Improving searching and reading performance: the effect of highlighting and text color coding. *Information & Management*, **40**(7), 617–637.
- [51] Xu, B., Xing, Z., Xia, X., and Lo, D. (2017). 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.
- [52] Yang, X., Wang, S., Li, Y., and Wang, S. (2023). Does data sampling improve deep learning-based vulnerability detection? yeas! and nays! In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 2287–2298. IEEE.
- [53] Ye, D., Xing, Z., Li, J., and Kapre, N. (2016). Software-specific part-of-speech tagging: An experimental study on stack overflow. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 1378–1385.

- [54] Yin, H., Sun, Z., Sun, Y., and Jiao, W. (2019). 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.
- [55] Zhang, H., Wang, S., Chen, T.-H., Zou, Y., and Hassan, A. E. (2019). An empirical study of obsolete answers on stack overflow. *IEEE Transactions on Software Engineering*, **47**(4), 850–862.
- [56] Zhang, H., Wang, S., Chen, T.-H., and Hassan, A. E. (2021a). Are comments on stack overflow well organized for easy retrieval by developers? *ACM Transactions on Software Engineering and Methodology (TOSEM)*, **30**(2), 1–31.
- [57] Zhang, Y., Kang, B., Hooi, B., Yan, S., and Feng, J. (2021b). Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*.
- [58] Zhu, Q., Li, X., Conesa, A., and Pereira, C. (2018). Gram-cnn: a deep learning approach with local context for named entity recognition in biomedical text. *Bioinformatics*, **34**(9), 1547–1554.