

# Data-Driven Proactive Policy Assurance of Post Quality in Community Q&A Sites

CHUNYANG CHEN, Monash University, Australia

XI CHEN, JIAMOU SUN, and ZHENCHANG XING, Australian National University, Australia

GUOQIANG LI, Shanghai Jiao Tong University, China

To ensure the post quality, Q&A sites usually develop a list of quality assurance guidelines for “dos and don’ts”, and adopt the collaborative editing mechanism to fix violations of community norms. Guidelines are mostly high-level principles, and many tacit and context-sensitive aspects of the expected community norms cannot be easily enforced by a set of explicit rules. Collaborative editing is a reactive mechanism after low-quality posts have been posted. Our study of collaborative editing data on Stack Overflow suggests that tacit and context-sensitive norm-meeting knowledge is manifested in the editing patterns of large numbers of collaborative edits. Inspired by this observation, we develop and evaluate a Convolutional Neural Network based approach to learn mid-level editing patterns from historical post edits for predicting the need of editing a post. Our approach provides a proactive policy assurance mechanism that warns users potential issues in a post before it is posted.

CCS Concepts: • **Applied computing** → **Text editing**; • **Human-centered computing** → *Collaborative and social computing*;

Additional Key Words and Phrases: Q&A Sites; Quality assurance; Deep learning; Collaborative editing

## ACM Reference Format:

Chunyang Chen, Xi Chen, Jiamou Sun, Zhenchang Xing, and Guoqiang Li. 2018. Data-Driven Proactive Policy Assurance of Post Quality in Community Q&A Sites. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 33 (November 2018), 22 pages. <https://doi.org/10.1145/3274302>

## 1 INTRODUCTION

Online question and answer (Q&A) sites are platforms for participants to ask and answer questions. Recent years has witnessed the popularity of both the general Q&A sites like Quora<sup>1</sup> and domain-specific Q&A sites like Stack Overflow<sup>2</sup>. As time flows, these Q&A sites have accumulated a large pool of valuable knowledge which serves millions of users around the world. However, the dramatic growth of posts and users on a Q&A site poses a severe challenge to the quality assurance of the site. For example, some less experienced users may post questions and answers with misspellings,

<sup>1</sup><https://www.quora.com/>

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

---

Authors’ addresses: Chunyang Chen, Faculty of Information Technology, Monash University, Melbourne, Australia, [chunyang.chen@monash.edu](mailto:chunyang.chen@monash.edu); Xi Chen; Jiamou Sun; Zhenchang Xing, Research School of Computer Science, Australian National University, Canberra, Australia, \*Xi Chen and Jiamou Sun contribute equally, [u6013686@anu.edu.au](mailto:u6013686@anu.edu.au), [u5871153@anu.edu.au](mailto:u5871153@anu.edu.au), [zhenchang.xing@anu.edu.au](mailto:zhenchang.xing@anu.edu.au); Guoqiang Li, corresponding author, School of Software, Shanghai Jiao Tong University, Shanghai, China, [li.g@sjtu.edu.cn](mailto:li.g@sjtu.edu.cn).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2018 Association for Computing Machinery.

2573-0142/2018/11-ART33 \$15.00

<https://doi.org/10.1145/3274302>

grammar errors, inappropriate code and text formatting, and the lack of important information (e.g., data visualization images, URLs of already-studied posts or referenced resources)

Such quality decay negatively influences the readability and understandability of posts, which may further discourage the participation of users [33]. To avoid the quality decay of the sites, Q&A sites like Stack Overflow provide official recommendations for effective question writing [7] and answer writing [8], by incorporating the suggestions [1, 2] from the highest reputation community member in Stack Overflow, Jon Skeet [9] who answered more than 34,000 questions in the last decade. These guidelines include different quality aspects such as adding source code, containing as much detail as possible, providing links to related resources, checking post formatting and spellings, being polite, etc. All of these guidelines can be regarded as site-specific **quality assurance policies**. By following these policies, the questions and answers created by users will be clear and easy to understand, leading to the boost of site quality.

Although all users are encouraged to obey the quality assurance policies, many users may still violate it carelessly or unintentionally. New users may not even be aware of the existence of these policies initially until other users point out the issues in their posts and recommend relevant policies to them. That is, quality assurance policies are a kind of tacit knowledge and implicit community norms in a Q&A site. Such community norms include a set of behaviours expected in a community, based on the community's values, traditions, policies, etc [4]. Although there are guidelines for users to read, they may still not clearly understand what they should or should not do in different circumstances, because the guidelines are mostly high-level principles and descriptions which lack detailed examples of patterns and anti-patterns. Furthermore, many community norms are context sensitive, which lead to different rules in different context. For example, users should use Markdown list to highlight text structure, while using proper code indents for code structure on Stack Overflow. Therefore, it is not only difficult for policy makers to enumerate all policy rules, but also for users, even experienced users to decide what rules they should use in different contexts.

To ensure the site quality [36], Stack Overflow encourages users, especially experienced users<sup>3</sup> to collaboratively edit the posts to make them comply with the site quality standards. Among 36,943,972 posts including questions and answers (as of August 2017), 14,639,359 (39.6%) of them have been edited at least once. Post edits involve not only minor corrections of misspellings and grammar errors, but also improving text and code formatting, removing unnecessary contents, and adding related resources (e.g., screenshots) or hyperlinks.

Although collaborative editing is beneficial for the community [33], there are still three problems with such mechanism. First, it requires significant community effort, especially from high-reputation users to edit the posts directly and/or approve edits by other users. Second, some violations of community norms, especially relatively complicated ones such as whether some code, images or hyperlinks are needed or not, are difficult to spot, as they may require a good understanding of the question or answer content. Third, all these collaborative edits are reactive to existing policy violations which may have already harmed the readers of the posts before edits, or made it difficult for those who want to help to answer the questions.

In addition to collaborative editing for reactive policy assurance, we also need a more proactive mechanism of policy assurance which could check a post before it is posted, spot the potential issues in the post, and remind the post owner to fix issues if any. If the post owner publishes the post regardless of the proactive warnings of potential issues, this policy assurance checker could also attract other users' attention to such potentially problematic posts so that these users could either edit the posts directly or recommend the post owner to edit the post timely. The goal of our work is to develop such a proactive policy assurance mechanism which can complement the collaborative

<sup>3</sup><https://stackoverflow.com/help/editing>

editing mechanism. Our work is data driven and built on top of existing collaborative editing patterns by users. Therefore, our research questions are two-fold: 1) *what kinds of community norms are manifested in collaborative edits?* 2) *what techniques can effectively learn collaborative editing patterns by human users to assist users in following community norms proactively?*

To answer the first research question, we conduct an empirical study of millions of collaborative edits on Stack Overflow to understand common types of post edits and editing details within these types. By analysing the topics in the comments of post edits, besides minor sentence-level edits, four middle-level editing types emerge, including code formatting, text formatting, hyperlink modification and image revision. Chen et al.'s sentence correction method [20, 21] can correct only minor mistakes like misspellings or grammar errors, but it cannot recommend more complex post edits such as formatting code or text, adding hyperlinks, or adding images. Considering the wide range of post contents and formats involved in post edits (see Fig. 1 for examples), it would require significant manual effort to develop a complete set of rules for representing editing patterns.

This challenge motivates us to develop a deep-learning based post policy assurance janitor to recommend to post owners or other users potential mid-level edits that may be needed to make a post meet community norms in Stack Overflow, such as post edits shown in Fig. 1. This janitor can also justify its prediction by pointing out the post content relevant to the predicted edits. In particular, we formulate post edit recommendation as a text classification problem, in which a post can be predicted as "need certain type of edit" or "no need for certain type of edit". In this work, we consider four types of mid-level post edits, i.e., code format, text format, link modification, image revisions identified in our formative study of collaborative edits. Inspired by the recent progress of deep learning for text classification [26, 35, 51], we adopt Convolutional Neural Network (CNN) to predict if a post needs or not need any of these four types of edits. Furthermore, we exploit the computation structure of CNN to identify key phrases in the post that contribute most to a classification result. This helps users understand why CNN recommends certain edits to the post.

To evaluate our approach, we collect a large dataset of edited posts (as positive data) and non-edited posts (as negative data) from Stack Overflow post edit history for model training and testing. Our results show that our approach outperforms other machine learning based and deep learning based baselines for post edit prediction. We then explain the detailed comparison of our model with the baselines, and also analyze reasons for common prediction errors by our model. Finally, we visualize and check what phrases in the posts contribute to the post edit predictions which uncover the black box of our deep learning model. We make the following contributions in this paper:

- We conduct an empirical study to investigate the Stack Overflow's collaborative editing practices, including editing types, users' attention on different types of edits, scale of changes, and concrete editing patterns. This study identifies four types of post edits that an automatic tool can predict and the requirements for the tool.
- We develop a CNN-based approach that learns from historical post editing data to predict the type(s) of edits that a post may need and help users understand why the edit is recommended.
- We conduct large-scale experiments to evaluate the performance and limitations of our approach. Our approach achieves good precision, recall, and F1-score (at least 0.7) on a large dataset of collaborative editing data. Qualitative analysis of the CNN-learned key phrases confirms their intuitiveness for understanding the CNN's prediction results.

## 2 RELATED WORK

There is a growing body of research on the quality of user-generated content, which can be broadly divided into two groups: 1) research on the importance of the quality of user-generated content to the community, and 2) research that assists the quality assurance of user-generated content.

The general consensus is that the quality of user-generated content is a key factor to attract users to visit knowledge-sharing websites such as Wikipedia, Quora, Stack Overflow. Much research effort has been put into the investigation of the content quality of such websites. For example, Stvilla et al. [44, 45] carry out an empirical study to examine the information quality aspects of Wikipedia which can then guide the construction of context-specific information quality models. Allahbakhsh et al. [12] propose a framework for characterizing various dimensions of quality control in crowdsourcing systems, and also potential issues and research directions. Studies [33, 36] show that collaborative editing can improve the content quality of Stack Overflow. Different from these works, our empirical study distills common post editing types and patterns that manifest implicit quality assurance policies on Stack Overflow by analysing large-scale historical collaborative editing data. Analyzing common post editing types and patterns helps us determine the feasibility of proactive post edit recommendation and inspires the design of our CNN-based approach.

As the solely manual collaborative editing is time-consuming and labor extensive, machine learning technique has been developed to assist collaborative editing in online communities. Hu et al. [29] and Anderka et al. [13] train a classifier to predict the quality issues of articles on Wikipedia. Wikipedia itself also develops the Objective Revision Evaluation Service (ORES) [3] to separate blatant vandalism from well-intentioned changes in the edit review process. Ponzanelli et al. [40] present an approach to detect low-quality posts in Stack Overflow based on simple textual features and readability metrics. Our edit recommendation approach is different in that it works at a more fine-grained level. Our approach tells not only if the post is of good or poor quality, but also predict what type(s) of edits may be needed. In addition, we adopt more advanced deep learning (CNN) algorithm to improve the prediction performance and justify the model's prediction.

Some researchers also adopt deep learning methods to assist the quality assurance of user-generated content. Chen et al. [20] use a RNN encoder-decoder to predict if minor sentence-level revisions are needed such as grammar errors, misspellings. Xie et al. [47] add attention to the RNN encoder-decoder to assist grammatical error correction. Both works target at minor revisions, while our work involves prediction of more complicated editing types such as adding code or text format, hyperlinks or images. Compared with their sentence-level prediction, our task is more difficult with modeling more contextual information at post-level which involves multiple sentences.

We formulate the post edit prediction in this work as a text classification problem. Many approaches have been proposed to tackle that problem, ranging from machine learning [25, 30] to deep learning [26, 35, 51]. Among deep learning based methods, CNN [51] and RNN [35] are widely adopted for text classification. As RNN especially LSTM is capable at capturing the overall semantic of text, it works well in a broad range of tasks except when the task is essentially a keyphrase recognition task [14, 49]. According to our observation, most post edits depend on some key phrases or words in the post, hence we adopt the CNN model in this work. In addition, by leveraging key phrase recognition capability of CNN, we can further extract those key phrases which can help users understand why certain type of post edits are recommended.

### 3 EMPIRICAL STUDY OF COLLABORATIVE EDITING IN STACK OVERFLOW

Stack Overflow is selected as our study subject, not only because of its popularity and large user base [17–19], but also because it is a well-known online Q&A site which supports collaborative editing [33, 36]. We download the latest data dump<sup>4</sup> of Stack Overflow which contains 36,943,972 posts (including 14,237,281 questions and 22,618,594 answers). It was released by Stack Overflow at 27th August 2017 and it was a snapshot of all post edits since the launch of Stack Overflow in 2007 to 27th August 2017. Based on this large dataset, we carry out an empirical study of post edits in

<sup>4</sup><https://archive.org/download/stackexchange>

### 1.1 Minor Revision (Spelling)

I've found SVN to be **extremely useful** for documentation, personal files, among other non-source code uses. What other practical uses have you found to version control systems in general?

I've found SVN to be **extremely useful** for documentation, personal files, among other non-source code uses. What other practical uses have you found to version control systems in general?

### 1.2 Minor Revision (Grammar)

Why should I move away from them as long as **it works** on my site?

Why should I move away from them as long as **they work** on my site?

### 2. Code Edit (Format)

```
$safe_variable = mysql_real_escape_string($_POST["user-input"]);
mysql_query("INSERT INTO table (column) VALUES ('" . $safe_var
```

```
$safe_variable = mysql_real_escape_string($_POST["user-input"]);
mysql_query("INSERT INTO table (column) VALUES ('" . $safe_variab
```

### 3. Text Edit (Format)

However, you may start a standalone instance as a replica set by using the command, **1) Start Mongo server in replica mode: mongod --dbPath --replSet rs0 2) Initiate the replica set (Execute from mongo shell) rs.initiate();**

However, you may start a standalone instance as a replica set by using the command,

1. **Start Mongo server in replica mode.**
  - `mongod --dbPath <path_to_data_file> --replSet rs0`
2. **Initiate the replica set (Execute from mongo shell)**
  - `rs.initiate();`

### 4. Link Modification

Get Bruce Schneier's book **Applied Cryptography** and read it carefully.

Get Bruce Schneier's book **Applied Cryptography** and read it carefully.

### 5. Image Revision

You can just rearrange your keys on your current keyboard and change the layout.

Here is the key layout: **alt-text http://rfr.de/images/dvorak.jpg**

You can just rearrange your keys on your current keyboard and change the layout.



Fig. 1. Examples of five types of post edits

Stack Overflow to understand the characteristics of post editing and to motivate the required tool support.

### 3.1 What has been edited?

In Stack Overflow, there are three kinds of post information which can be edited, i.e., question tags, question title, and post body [33]. As of August 27, 2017, there have been in total 26,025,446 post edits. Among them, 2,106,079 (8.1%) are question-title edits, 2,930,379 (11.3%) are question-tag edits, and the majority of post edits (20,988,988 (80.6%)) are post-body edits<sup>5</sup>. The tags of 2,498,568 (17.6%) questions, the titles of 1,832,054 (12.9%) questions, and the body of 13,341,488 (36.2%) posts have been edited at least once. Overall, post-body edits make up the majority of post edits. And compared with adding/removing question tags, revising question titles, post-body editing is more complex (further studied in the next question). Therefore, we focus on post-body edits in this work.

### 3.2 What kinds of edits have been made?

We analyze the comments of post edits to understand what post edits are about. In Stack Overflow, when users finish editing a post, they can add a short comment to summarize the post edit. We

<sup>5</sup>Some posts that need some edits may not be actually edited (false negatives), while some posts that have been edited may not need the edits (false positives). We assume that the majority of Stack Overflow post editing data is reliable due to the community curation of the site content.

Table 1. Frequent phrases from post-edit comments

Unigram		Bigram		Trigram	
Phrases	Frequency	Phrases	Frequency	Phrases	Frequency
http	1585947	improve format	439122	insert duplicate link	45842
format	1481740	code format	235428	fix code format	28170
code	1102378	add code	110441	improve code format	22391
grammar	359107	fix grammar	100943	fix trivial typos	11617
link	344236	correct spell	100805	add code format	8983
spell	329189	add link	69231	add syntax highlight	6383
typo	197242	fix typo	68143	add code example	6279
example	150384	fix format	64678	add code block	6191
rollback	103205	add example	38924	fix code indentation	6015
image	77574	add detail	37293	add backticks around	5409
readability	73245	syntax highlight	23504	add code snippet	5403
syntax	70763	code indentation	21247	fix code block	5373
indentation	68273	add image	19264	format code block	4911
highlight	62129	edit code	17495	add sample code	4830
reference	46467	update link	17289	add example code	4693

collect all post-edit comments and apply standard text processing step to post-edit comment such as removing punctuations, lowercasing all characters, and excluding stop words.

**3.2.1 Frequent words or phrases.** We extract unique unigrams, bigrams and trigrams from post-edit comments and count their frequencies in the corpus of post-edit comments. n-gram is a contiguous sequence of n words from text. The most frequent unigrams, bigrams and trigrams can be seen in Table 1. Some frequent words and phrases are about grammar problems such as “grammar”, “correct spell”, “fix trivial typos”. In addition, post-edit comments also indicate many other types of post edits, such as code formatting (“code format”, “fix code indentation”), text formatting (“readability”, “syntax highlight”), hyperlink modification (“http”, “update link”, “insert duplicate link”), and image revision (“add image”).

**3.2.2 Editing topics and types.** To extract common editing types, we adopt the Latent Dirichlet Allocation (LDA) model [16] to analyze the post-edit comments. LDA is a statistical model for discovering abstract topics that occur in a collection of documents in which each topic consists of a set of keywords. A significant limitation of LDA is that it considers only single words (i.e., unigrams). However, as seen in Table 1, a single word may not capture the exact semantics of the post-edit comments. In contrast, phrases that are composed of several words are more intuitive to understand the intention behind post edits, such as *add syntax highlight* instead of *highlight*, *fix code format* rather than *code*.

Therefore, these multi-word phrases must be recognized and treated as a whole in LDA model. We adopt a simple data-driven and memory-efficient approach [38] to detect multi-word phrases in post-edit comments. In this approach, phrases are formed iteratively based on the unigram and bigram counts, using the following formula:

$$score(w_i, w_{i+1}) = \frac{count(w_i w_{i+1}) - \delta}{count(w_i) \times count(w_{i+1})} \times N \quad (1)$$

The  $w_i$  and  $w_{i+1}$  are two consecutive words.  $\delta$  is a discounting coefficient to prevent infrequent bigrams to be formed. That is, the two consecutive words will not form a bigram phrase if they appear as a phrase less than  $\delta$  times in the corpus.  $N$  is the vocabulary size of the corpus. In this work, we experimentally set  $\delta$  at 10 and the threshold for *score* at 15 to achieve a good balance between the coverage and accuracy of the detected multi-words phrases.

Our method can find bigram phrases that appear frequently enough in post-edit comments compared with the frequency of each unigram, such as “*improve format*”. But the bigram phrases like “*it is*” will not be formed because each unigram also appears very frequently in the text. Once the bigram phrases are formed, we repeat the process to detect trigrams. All these phrases are then



Table 2. Topics of post-edit comments

ID	Topic Name	Keywords
1	fixing grammar	minor grammar, improve grammar, syntax, fix grammar
2	spelling correction	typos, fix typo, spell mistake
3	code modification	code snippet, indent code, code block
4	format improvement	improve format, reformatted, layout, remove fluff
5	body edits	body, character, clarity, improve answer, example, info, highlight
6	links modification	fix break link, link, update link
7	image revision	image, add image, fix image
8	title edits	edit title, change title
9	tag edits	edit tag, remove obsolete tag, add tag, fix tag

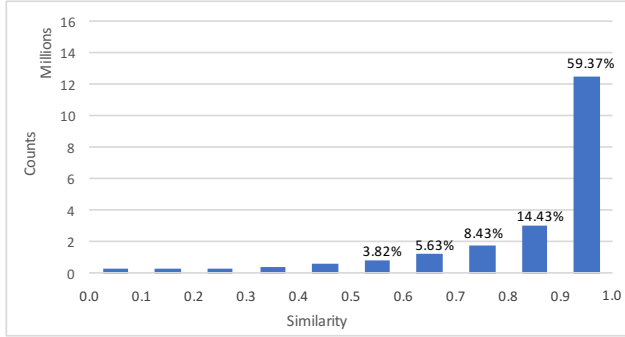


Fig. 2. The count of original-edited post body in different similarity range

concatenated with underline like “*improve\_format*” in the corpus of post-edit comments, and then we use the LDA model to extract the topics.

Table 2 lists 9 topics with corresponding keywords of each topic. Note that we annotate the topic name with our own summarization based on the topic keywords. According to these topics in post-edit comments, there are five major post-body editing types (except #8 title edits and #9 tag edits), including 1) minor revisions (#1 fixing grammar and #2 spelling correction), 2) code revision (#3 code modification and #4 format improvement), 3) text revision (#4 format improvement and #5 body edits), 4) link modification (#6 link modification), and 5) image revision (#7 image revision). For each of these five editing types, we also display real-world examples in Fig. 1. These five editing types represent the community norms that Stack Overflow commits its effort to maintain.

### 3.3 What is the scale of changes that post edits involve?

User can decide to carry out five different types of edits to post body according to different goals or context. To understand the scale of changes that post edits involve, we measure the similarity between the post body before an edit and the post body after the edit. Given a post edit, let *original* and *edited* be the text content of the original and edited post body. We use the text-matching algorithm [15] to find the char-level Longest Common Subsequence (LCS) between the *original* and *edited* content. We measure the similarity between the original and edited post body as:

$$\text{similarity}(\text{original}, \text{edited}) = \frac{2 * N_{\text{match}}}{N_{\text{total}}} \quad (2)$$

where  $N_{\text{match}}$  is the number of characters in the LCS and the  $N_{\text{total}}$  is the total number of all characters in both the *original* and *edited* content. The similarity score is in the range of 0 to 1, and the higher the similarity score, the fewer changes between the original and edited post body.

As shown in Fig. 2, among the 20,988,988 post-body edits, the original and edited post body of 59.37% edits has the similarity score between 0.9 and 1. That is, the majority of post-body edits involve small scale of changes. We randomly sample 100 edits with similarity score in 0.9 and 1

and another 100 edits with similarity score smaller than 0.9. We manually check these edits and summarize three different scales of changes.

First, many edits are minor changes such as correcting misspellings (e.g., “javascrip” to “javascript”), grammar errors (e.g., “it slow down the whole program” to “it slows down the whole program”), keyword highlight (“click OK button” to “click ‘OK’ button”), etc. These revisions involve only sentence-level edits without the needs for the whole-post consideration.

Second, some edits are medium-level changes such as adding some Markdown elements to organize the text like adding “header” to make the subtitle bold, adding “<li>” to make the bullet points clearer. Formatting is also frequently applied to the code, for example, adding code tag “<code>” or adding code indents to discriminate code from plain text. Users may also add external resources to their posts such as the hyperlinks or image links. Although these medium-level edits involve larger scale of changes than minor revisions, most posts after medium-level changes are still very similar to the original posts, i.e., their similarity score is mostly higher than 0.9. But different from minor changes, decisions on such medium-level changes can only be made by considering the content of the whole post (see Fig. 6 for examples).

Third, other edits are major changes such as clarifying a post by adding an example code snippet, a new paragraph of text description, or replacing the current code with more efficient code. Such major changes always involve significant changes of post content so that most edited posts are not very similar to their original versions, leading to similarity score smaller than 0.9.

The minor changes like correcting grammar and spellings have already been well solved by Chen et al.’s work [20]. In this work, we are moving one step further by focusing on the medium-level changes including code format edits, text format edits, link edits, and image edits. Major text or code changes are much more complicated than these four types of edits. So they need deeper understanding of not only a post content, but also other posts in the discussion thread, such as editing an answer by considering its questions, editing a question’s body by considering its comments. We leave such major changes as the future work.

### 3.4 What are the detailed editing operations?

In Stack Overflow, all posts are written in Markdown language<sup>6</sup>. For each of the four types of post edits (code formatting, text formatting, hyperlink modification and image revision) summarized in the last section, we observe the corresponding detailed Markdown changes for detecting instances of a particular type of edits as follows:

- To distinguish the code from the text, Stack Overflow requires users to annotate their code with `<pre>` and `<code>` tags (Fig. 1.2). Users also have to use proper code indents for improving code readability. We call such revision as *code formatting*.
- To improve *text format*, users are encouraged to add headers and use list to make their post as clear as possible (Fig. 1.3). Such change is reflected in the post edits by adding HTML tags like `<h1>`, `<h2>` and `<li>`.
- For *link modification*, Stack Overflow provides two different ways. First, users can directly add bracket (i.e. `<` and `>`) to enclose the original link text to make it clickable (Fig. 1.4). Second, user can add the link reference to a piece of text in the post in which the link will be listed at the end of the post.
- In Stack Overflow posts, embedding an image requires a special link ending with postfix including `.png`, `.jpg`, `.gif`, `.bmp` and `.tiff` (Fig. 1.5). We refer to this kind of revision of image links as *image revision*.

<sup>6</sup><https://stackoverflow.com/editing-help>



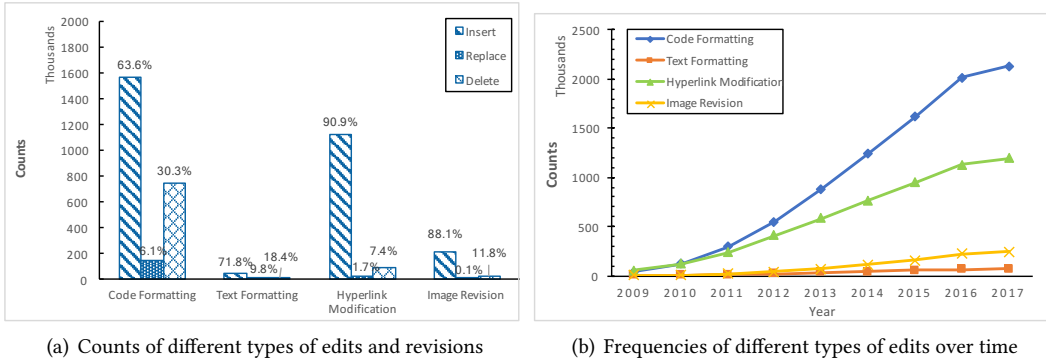


Fig. 3. Statistics of different types of edits and different kinds of revisions

For each editing type, there are three kinds of revisions i.e., add, delete and replace. If a post undergoes an *add* revision, users add new contents that belong to some type(s) of edits. On the contrary, *delete* revision means that some contents related to some editing types are deleted from the initial posts. *replace* revision means that some contents related to some editing types are replaced by other same-type content. For example, if the size of the header in a post is changed (e.g., from h1 to h2), then that post is considered having a *replace text format* edit. However, if the header is replaced by a hyperlink, then the post undergoes a *delete text format* edit and an *add hyperlink* edit.

By differencing the original post body and the edited post body, we count the number of different types of edits and different kinds of revisions, as summarized in Fig 3(a). Overall, most revisions are about *add revisions*, followed by *delete* revisions and then *replace* revisions.

- 2462919 (61.20%) post-body edits include *code format* edits. Among them, 1567272 (63.63%) edits are *adding code format*, 150597 (6.11%) edits are *replacing code format*, and 745050 (30.25%) edits are *deleting code format*;
- For *text format* editing, 73708 (1.83%) post-body edits include some edits of header or list. Among them, 52945 (71.83%) posts have been added headers or lists, 7198 (9.77%) posts have been replaced headers or lists, and 13565 (18.40%) posts have been deleted headers or lists;
- For *hyperlink modification*, 1238935 (30.79%) post-body edits include edits of hyperlinks. 1126252 (90.90%) edits are *adding hyperlinks*, 20612 (1.66%) edits are *replacing hyperlinks*, and 92071 (7.43%) edits are *deleting hyperlinks*;
- For *image revision*, 248795 (6.18%) post-body edits include edits of images. 219215 (88.11%) edits are *adding image*, 302 (0.12%) edits are *replacing image*, and 29278 (11.77%) edits are *deleting image*.

In Fig. 3(a), we observe that the frequency of *deleting code format* is much higher than other types of *delete* edits. Analyzing *deleting code format* edits reveals two main reasons. First, users want to improve code formatting by deleting unnecessary code indents, e.g., changing from eight-space indents to four-space indents. Second, some text contents are sometimes presented in code format. Users fix the misuse of code format by deleting code format.

We also calculate the frequency of the four types of edits over time, which is shown in Figure 3(b). Overall, all the four types of post edits increase along the time. Nevertheless, the *code format* edits rise up the fastest, followed by *link modification* edits and *text format* edits. The *image revision* edits only account for a small proportion of edits and the number of *image revision* edits goes up relatively slowly, compared with the other three types of edits.

**Summary:** Post-body edits account for 80.6% of all post edits in Stack Overflow. They involve a variety of editing types, including minor grammar and spelling correction, the four types of medium-scale edits (code formatting, text formatting, link modification and image revision), and the large changes of post contents. In this work, we focus on assisting the four types of medium-scale edits. Although different types of edits differ in their occurrence frequencies, all of them attract substantial users' attention over time. As such, a post-body edit recommendation algorithm should be able to recommend all the four types of edits to the posts. Each of these four types of edits may involve three kinds of revisions: *add*, *replace* and *delete*. *Add* revision occurs much more frequently than the other two kinds of revisions. Therefore, we focus on assisting *add* revision in this work.

## 4 RECOMMENDING POST-BODY EDITS BY CONVOLUTIONAL NEURAL NETWORK

The four types of post-body edits identified in our empirical study manifest Stack Overflow policies and norms for curating the post body. Unfortunately, these policies and norms are implicit knowledge in millions of post-body edits. Considering the diversity of post-body editing types and contexts, it would require significant effort to manually build a complete set of policy assurance rules to deal with all different situations. Therefore, we propose a deep-learning based approach which can automatically learn editing patterns from historical post-body edits, and recommend post-body edits and justify its recommendation based on the learned post-body editing patterns.

### 4.1 Approach overview

The core of our approach is a Convolutional Neural Network (CNN). To train the CNN for post-body edit prediction, we first collect all original posts which are then edited by the four editing types identified in our empirical study (Section 4.2). Note that a post may undergo one or more types of edits. We then formulate the post-body edit recommendation task as a text classification problem. The input to the classification problem is the text content of a post's body, and the output is a decision whether the post body needs a particular type of edit or not (one decision for each of the four editing types). We train the CNN model with a large dataset of *< original – post, post – body – edit – type >* pairs (Section 4.3). The trained CNN model can be used to not only predict the types of edits needed for a given post, but also point out the key phrases in the post content that are relevant to the recommended edits (Section 4.4).

### 4.2 Collecting the dataset of *< original – post, post – body – edit – type >* pairs

In Stack Overflow, a post can be edited several times. Assume a post has  $N$  versions, i.e., undergoing  $N - 1$  post edits. For each post edit  $i$  ( $1 \leq i \leq N - 1$ ), we collect a pair of the original post-body content and the edited content. The original content is from the version  $i$  of the post before the edit, and the edited content is from the version  $i + 1$  of the post after the edit. As seen in Fig. 3(a), most edits are *additive* revisions. Therefore, in this work we focus on predicting whether a post needs adding code format, adding text format, adding hyperlink and/or adding image. Correspondingly, we collect only the addition of these four types of edits from historical post-body edits.

- For adding code format, we select posts that include inserting code indents. That is, the added contents must contain only indents such as tabs and spaces within a sentence line. Besides, the Markdown tag `<code>` and `<pre>` are also considered as the recognition marks for the edits of code formats. Thus, the edits that include the addition of `<code>` and `<pre>` are also regarded as adding code format.
- For adding text format, we find the addition of Markdown elements of list and headers. The list addition includes adding line breaks or list markdown characters (+, –, \* and numbers). The header addition includes adding line breaks or one of the –, = or # marks.

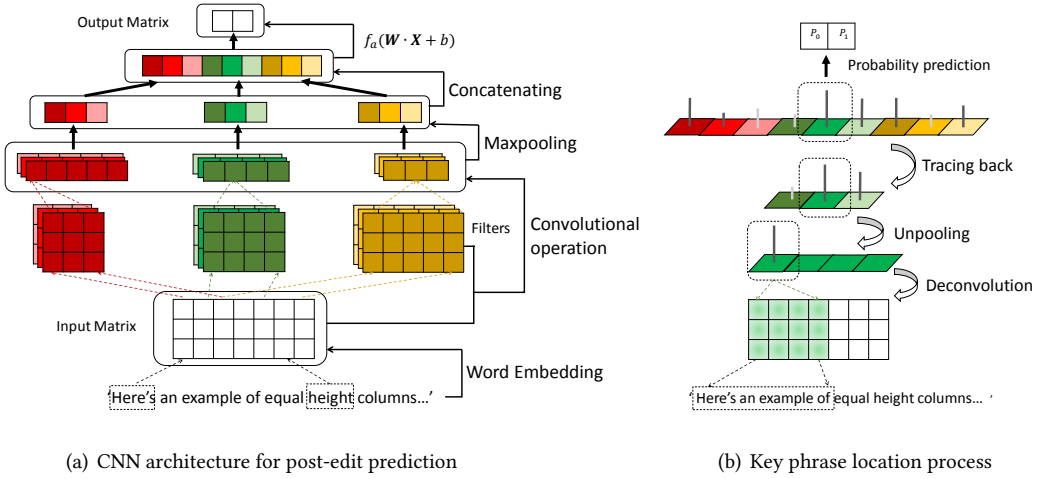


Fig. 4. Approach

- For adding hyperlinks, we find the addition of hyperlink Markdown elements in the edits. The links are not image links like those ending with *.jpg*, *.png*, etc.. Moreover, there must be the angle symbols “< >” that contain a hyperlink (e.g., <https://example.com>) or the square brackets “[ ]” before a hyperlink (e.g., [Google] (http://www.google.com/)).
- For adding images, there must be link addition with image postfix including *.jpg*, *.png*, *.tiff*, *.bmp* and *.gif*. To recognize adding images, the edits must also contain the angle symbols “< >” or the square brackets “[ ]”.

As we formulate post-body edit recommendation as a text classification problem, the collected  $\langle \text{original} - \text{post}, \text{post} - \text{body} - \text{edit} - \text{type} \rangle$  pairs are positive data for the classification problem, i.e., posts that need certain types of edits. We also need to collect posts which do not need certain types of edits, i.e., negative data. Therefore, for each type of edits, we collect the same number<sup>7</sup> of posts that do not need this type of edits. However, a post that does not undergo a type of edit does not necessarily mean that it does not need this type of edit. To mitigate this potential threat, we collect negative data only from posts that have been created more than a year ago, have received more than 3 score points (upvote - downvote), and do not undergo a particular type of edit. The underlying rationale is that a post that has been posted long enough and attracted enough attention in the community but still does not undergo a type of edit would be unlikely to need that edit.

Finally, from the post-body edits before Aug 27, 2017, we collect 1,305,593 posts for adding code format, 50,405 posts for adding text format, 1,086,635 posts for adding hyperlinks, 211,397 posts for adding images, and the same numbers of negative data for each type of edits.

### 4.3 CNN for Edit Prediction

Convolutional Neural Networks (CNNs) is a class of deep, feed-forward neural networks, most commonly applied to analyzing visual imagery. Apart from images, some recent works have also successfully applied CNNs to model sentence- and document-level semantics for NLP tasks such as text classification [22, 48], sentiment analysis [24, 42], and paraphrase detection [28, 50].

Figure 4(a) shows the model architecture of the CNN for our post-edit prediction task. Instead of image pixels, the input to NLP tasks are sentences or documents which need to be represented

<sup>7</sup>To make a balanced dataset of positive and negative data for model training and testing [46].

as a matrix. To apply the CNN to text, words of a sentence need to be converted into vectors by word embeddings [37, 38] and these vectors comprise the matrix. Let  $x_i \in \mathbb{R}^k$  be the  $k$ -dimensional word vector corresponding to the  $i$ -th word in the sentence. A sentence of length  $n$  is represented as  $x_{1:n} = x_1 \oplus x_2 \oplus \dots \oplus x_n$ , where  $\oplus$  is the vector concatenation operator.

We treat the matrix (sentence vectors) as an “image”, and perform convolution with filters that slide over each row of the matrix. In text applications, because each word is represented as a  $k$ -dimensional vector, it is reasonable to use filters with widths equal to the dimensionality of the word vectors (i.e.,  $k$ ). Thus we simply vary the *window size* (or *height*) of the filter, i.e., the number of adjacent words considered jointly.

One may also specify multiple kinds of filters with different window sizes, or use multiple filters for the same window size to learn complementary features from the same word windows. In Figure 4(a), we illustrate 3 filters for the window size of  $h = 3, 4, 5$ . The dimensionality of the feature map generated by each filter will vary as a function of the sentence length and the filter’s window size. Thus, a pooling function is then applied to each feature map to induce a fixed-length vector. A common strategy is *1-max pooling*, which extracts a scalar (i.e., a feature vector of length 1) with the maximum value for each filter. The idea behind it is to capture the most important feature, one with the highest value for each feature map.

Together, the outputs from each filter can be concatenated into a fixed-length, “top-level” feature vector which is then fed through a fully connected layer with Softmax as its activation function to generate the final classification. Our task is a binary classification task and hence has two possible outputs for each edit type (need this type of edit or no need for this type of edit). All the four editing types are independent, and it means that one post can receive one or several of the four types of edits. We train four CNN models separately for the four editing types respectively. For a post, the four models will individually determine if the corresponding editing type is needed or not.

The training objective to minimize the categorical cross-entropy loss with calculating parameters such as the weight matrix, the bias term of the filters, the weight vector and the bias term of the classifier. Optimization is performed using Stochastic Gradient Descent and back-propagation[41].

#### 4.4 Locating the Key Phrases in Posts to Explain the Edit Prediction

As existing CNN de-convolution methods generally focus on image classification [39], we propose a different method for locating the key phrases in document-level texts. A significant feature of CNN model in text classification is that each element of output matrices in intermediate layers (convolutional layer - fully connected layer) represents a  $n$ -gram phrase where  $n$  is determined by the filter window sizes in the convolution operations. Therefore, we can extract prominent weighted outputs representing key phrases.

As seen in Fig. 4(b), our key phrase extracting method includes two steps: 1) tracing back through the model to locating the filtered phrases in the input layer; and 2) predicting the contribution score of the phrases’ corresponding features in the fully connected layer to the prediction class. Sorting the phrases by the prediction scores helps to locate key phrases in the post content that altogether determine the predicted editing type for the given post.

Locating phrases is actually a process to find by which continuous word indices each output value in the concatenate layer is influenced through a series of feature mapping, unpooling and deconvolution operation. After mapping the feature in the concatenate layer to the feature map in the max-pooling layer that the feature belongs to, we detect the location of this maximum feature value in the feature map. Unpooling means that generating a new feature map of the same size as the original feature map by only keeping the maximum value but setting all other values in the feature map to 0. Finally, a phrase in the input text with word vectors in the input sentence matrix can be extracted after applying deconvolution to the feature map with corresponding filters.

The predicted class of our CNN model is calculated by applying Softmax function on the sum of dot product of the weight matrix in the fully connected layer and the output feature vector of the concatenate layer. To gain the probability to which class a phrase in the input text belongs, we can perform the same Softmax on partial output feature vector of the concatenate layer due to their linearity and discreteness. The formula is as:  $P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^2 e^{x^T w_k}}$  ( $1 \leq j \leq 2$ ) where  $x$  is a vector comprising the output features in the concatenate layer corresponding to the same phrase. This is because several filters may extract the same phrase more than once. Each value in vector  $w_j$  is a fully connected layer weight of class  $j$  for its corresponding output.

If the most prominent phrase is needed, this method should only be applied once for the maximum feature in the concatenate layer. However, as a document-level text usually have several key phrases altogether determining whether the text should undergo a particular type of edit, the probability computation for all extracted phrases followed by a probability sorting step is necessary to obtain the top- $N$  key phrases that contribute most to the prediction of a editing type.

#### 4.5 Implementation

Different from the illustrative model in Fig. 4(a), our implementation is more complicated. The input word embedding dimension is 128. Each convolutional layer uses filters of window sizes 3, 4, 5 and for each window size, we use 256 filters. We implement our model in Keras [23] with Tensorflow [11] as the backend. Model training is performed on a Nvidia P40 GPU (24G memory) with 10 epochs. The training takes about 4 days in total.

### 5 EXPERIMENT

Our post edit recommendation tool aims to assist post owners or post editors in identifying quality issues in posts and highlighting key post contents relevant to the identified issues. The quality of recommended edits will affect the adoption of the tool by the users. In this section, we use randomly selected historical post editing data to evaluate the quality of recommended post edits by our tool.

#### 5.1 Dataset

From archival post editing history in Stack Overflow data dump, we collect 264,372 posts as positive data for adding code format, 38,168 posts for adding text format, 228,191 posts for adding links, and 189,884 posts for adding images (see Section 4.2 for data collection details). The same amount of negative data (i.e., the posts that do not need certain type of edits) are also collected. Following the deep learning practice guide [10], we split the whole dataset into three parts. For each edit type, we randomly take 80% of them as the training data, 10% as the validation data to tune model hyperparameters (e.g., the number of CNN layers, word embedding dimension), and the rest 10% as the testing data to evaluate the quality of editing types predicted by our tool [6].

#### 5.2 Baselines

Apart from our own CNN model, we take another four methods as baselines for comparison: the traditional machine learning methods including 1) logistic regression [25] and 2) SVM (Support Vector Machine) [30], and deep learning based methods including 3) FastText [26] and 4) attention-based LSTM (Long Short Term Memory) model. As the traditional machine learning methods are based on human-engineered text features, we take the widely-used TF-IDF (Term Frequency times Inverse Document Frequency) [43] as posts features for training logistic regression and SVM model. FastText is an open-source, light-weight library released by Facebook for text classification. It incorporates hierarchical softmax and  $n$ -gram features into a one-layer fully-connected neural network for fast text classification. Recurrent Neural Network (RNN) is a class of artificial neural

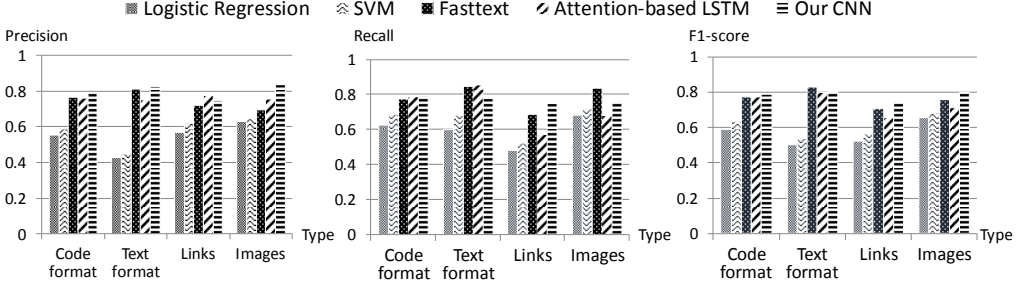


Fig. 5. Precision, recall and F1-score of five post-additive-edit recommendation methods

network where connections between units form a directed graph along a sequence, and it is widely used for text classification [31, 34]. To make the RNN baseline more competitive, we exploit its variant LSTM [27] which can capture long-distance information.

### 5.3 Evaluation metrics

As our task is a binary classification problem, the quality of our model's recommendation can be measured by three widely-used metrics including precision, recall, and F1-score. For all the three metrics, higher value will lead to better recommendation performance.

Precision is the proportion of posts that are correctly predicted as needing one certain type of editing among all posts predicted as needing that type of editing:

$$precision = \frac{\#Posts \text{ correctly predicted as needing a type of edit}}{\#All \text{ posts predicted as needing a type of edit}}$$

Recall is the proportion of posts that are correctly predicted as needing one certain type of editing among all posts that really need that type of editing:

$$recall = \frac{\#Posts \text{ correctly predicted as needing a type of edit}}{\#All \text{ posts really needing a type of edit}}$$

F1-score (F-score or F-measure) is the harmonic mean of precision and recall, which combine both of the two metrics above.

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}$$

F1-score will increase when an increase in precision (recall) outweighs a reduction in recall (precision). That is, a good balance of precision and recall will produce good F1-score.

### 5.4 Evaluation Results

We report the evaluation results of our CNN model in two aspects. First, we show how accurate our model can predict if a post needs certain type(s) of edits. Second, we further show how well our model can locate key phrases that contribute most to the prediction.

**5.4.1 Comparison of post-edit prediction accuracy by different methods.** Fig. 5 presents the precision, recall and F1-score of five different methods. Compared with three deep learning-based models (FastText, attentional-based LSTM, and our CNN model), the performance of traditional machine learning methods (logistic regression and SVM) is much worse than that of deep-learning model. It shows that the traditional machine-learning methods are not capable to capture the semantic of the post. In the following discussion, we only compare three deep-learning methods in detail. For precision, our CNN model has higher precision in all comparisons except for predicting



*adding hyperlinks* compared with attention-based LSTM. High precision is the most important metric for our post-edit recommendation task, as users are sensitive to false alarms. Furthermore, erroneous post-edit recommendations may mislead novice users' post edits which may increase the approval working load of trusted users (see Section 6.1 for further discussion on this point). For recall, there is no obvious winner among the three methods. The three methods' recalls are almost the same for predicting adding code format. Our model has higher recall than the two baseline methods for predicting adding hyperlinks, but has lower recall than the two baseline methods for predicting *adding text format*. For predicting *adding images*, our model is better in recall than attention-based LSTM but is worse than FastText. For F1-score, the three methods perform almost the same for predicting *adding code format* and *adding text format*, but our model has obvious advantages in predicting *adding hyperlinks* and *adding images*, compared with FastText and attention-based LSTM.

To qualitatively understand the strengths and weaknesses of different methods, we randomly sample 100 posts in testing data for manual inspection. Different from FastText which regards a post as a bag of words without order, our model takes into account sequential information which is important to natural languages. For example, one post contains sentence "*You can find more in this page*", but the post owner forgets to add the hyperlink to "*this page*". Although any single word in this sentence is not so meaningful, they as a whole indicate the need for adding a hyperlink. Such sentence sequential information is successfully captured by our model leading to an accurate prediction, while FastText fails to recommend *adding hyperlink*.

Compared with attention-based LSTM which can also capture sentence sequential information, our model is more effective in gathering separated information that is far away from each other in the post. Although attention-based LSTM is designed to capture long-distance information, our experimental results show that once the post is very long, LSTM may "forget" prior information when modelling the more recent text, as it reads text sequentially. For example, one of the testing post begins with "*Here is a compilation of verified information from various answers and cited references ...*" and concludes lots of functionalities of several command processors with incomplete reference links. Attention-based LSTM fails to predict the need for hyperlinks, because the relevant information is separated at the beginning and the end of the post. Instead, our model can extract important n-gram key phrases as the features which are fully connected in the last layer for the final prediction. Therefore, our model can overcome the problem of distant vital information which leads to better performance than attention-based LSTM.

**5.4.2 Error analysis of our model.** We also analyse the erroneous predictions by our model for the four editing types, and identify three main reasons for erroneous predictions:

First, some posts are edited to add more information which is beyond the context of a post. For example, one post edit adds not only a new hyperlink, but also some new content like "There is another method..." or "You can also try this...". The added hyperlink is highly dependent on the new content. However, as our model can only "see" the post content before editing, it cannot predict the addition of the hyperlink based on the existing context.

Second, different users may have different opinions regarding what should or should not be edited, especially for code format and text format. For example, some users prefer to add a header "Update" or "Edit" to updated content, while others may only add a horizontal line to separate the updated content from the original content. This often results in some edits being reverted back later, for example, from initial horizontal separation line to "Update" header and then back to horizontal separation line. Such back-and-forth edits lead to non-obvious editing patterns, which machine learning techniques cannot effectively encode.

Third, some negative data is not really negative data. Our way to collect negative data assumes that a post that did not undergo a type of edit does not need that type of edit. Although we use several heuristics to ensure that this assumption holds (see Section 4.2), the collected negative data still contains some “noisy” posts, i.e., posts that did not undergo a type of edit but actually needs that type of edit. For example, some posts actually need a type of edit, but because it did not receive enough attention from the community, it did not undergo that type of edit. For such “noisy” negative posts, our model may actually make the right prediction, but its prediction does not match the negative label (i.e. no need to edit) of the posts.

**5.4.3 Key phrase location.** To understand the key phrases that contribute most to our model’s prediction results, we develop a key phrase visualization method (see Fig. 6(a) to Fig. 6(d) for examples). In the visualization of key phrases, we concatenate the overlapping phrases and take the average of the prediction probability of individual phrases as the probability of the concatenated phrase. The reason for this concatenation step is that the filter window size in our CNN model has the upper bound, but the key post content that affects the prediction of needing a type of edits is likely longer than the maximum filter window size. For example, the maximum filter window size in our current implementation is 5. That is, the CNN “sees” only 5 consecutive words at most. But as shown in Fig. 6(a) to Fig. 6(d), the post content that is relevant to a type of edits is usually longer than 5. After the concatenation processing, we visualize the background color of a phrase proportional to its prediction probability. The darker the background color, the higher the probability.

We use this visualization to examine the correspondence between the key phrases that our CNN model pays attention to in a post to predict the type of needed post edit and the actual post edit made by users. Fig. 6(a) to Fig. 6(d) present one example for each type of post edit. The left part of the figure shows the attended key phrases that contribute most to predict a type of edit needed for a post and the right part shows the actual post edit. In Fig. 6(a), our CNN pays attention to a code-like sentence and predicts the need of adding code format. This code-like sentence is actually enclosed in `< code >` Markdown element in the edited post by users. In Fig. 6(b), our CNN pays attention to the sentence “*the following version of ...*” followed by names of several software tools and predicts the need of adding text format, which is actually made by users. In Fig. 6(c), our CNN pays attention to the sentence “*User the project converter which is designed for*” and some follow-up description of the mentioned tool. It predicts the need of adding hyperlinks, and the actual post edit does include a hyperlink to the mentioned tool. In Fig. 6(d), our CNN pays attention to a comment in the code snippet “*Display the resulting label matrix*” and some other relevant information in the code snippet and the text description. It predicts the need of adding image, and the post edit actually includes a test run of the code and the resulting image output.

As these examples demonstrate, our CNN model can identify the most prominent features in a wide range of information in posts, based on which it can make reliable prediction of the needed post edits. Furthermore, our key phrase identification and visualization method can be used to explain the CNN’s prediction results. Compared with just a simple prediction, this explainability can help post owners or post editors judge the validity of the recommended post edits and may encourage them to accept the tool’s recommendation. Making the prediction criteria more explicit is crucial for the acceptance of our deep learning based approach for post-edit recommendation.

## 6 DISCUSSION

In this section, we discuss the potential impact of our post-edit recommendation approach on collaborative editing and the generalizability of our approach.

**WordMergeFields.Add(wrdSelection.Range, ProductName)** →

The code above basically displays all the **productName** in different pages in word Document during merge.

Please help me how to put the data inside a table. I have to write multiple codes of this for my ProductName, AccountNo, OutBalance, AccountName, etc. My problem here is that I don't know how to put them in a table.

```
wordMergeFields.Add(wrdSelection.Range, "ProductName")
```

The code above basically displays all the productName in different pages in word Document during merge.

Please help me how to put the data inside a table. I have to write multiple codes of this for my ProductName, AccountNo, OutBalance, AccountName, etc. My problem here is that I don't know how to put them in a table.

(a) Adding code format

I had to revert back and explicitly specify the following version of

```
pip install docker
docker
docker-compose
pip install ansible
name install certain python modules for docker
apt
name { { item . name } }
version { { item . version } }
state present
with items
{ name docker , version 2 . 0 . 0 }
{ name docker py , version 1 . 10 . 6 }
```

All my play books work fine since then .

This is because new versions of python modules 'docker' and 'docker-py' that ansible uses are incompatible. I had to revert back and explicitly specify the following versions of PIP packages:

- docker: 2.0.0
- docker-py: 1.10.6

Sample playbook task for these:

```
- name: Install certain python modules for docker
  pip:
    name: "{{ item.name }}"
    version: "{{ item.version }}"
  state: present
  with_items:
    - { name: docker, version: 2.0.0 }
    - { name: docker-py, version: 1.10.6 }
```

All my play books work fine since then.

(b) Adding text format

Use the **Project Converter** which is designed for this . It is located under the **Windows Start Menu as Developer Express V2010 vol1 Components Tools Project Converter** . This will automatically change all of your references to the current version .

Use the project converter which is designed for this. It is located under the Windows Start Menu at "Developer Express V2010 vol1 -> Components -> Tools -> Project Converter". This will automatically change all of your references to the current version.

[Project Converter Documentation](#)

(c) Adding hyperlink

You can use 'ismember' to find where each label exists in your cell array . The second output will provide the index of the label . You can then use 'imagesc' with a custom colormap to display the result .

```
% Create a copy of Grid where the empty cells are replaced with ''
tmp = Grid;
tmp = cellfun(@(x) ['' x], Grid, 'UniformOutput', false);

% Locate all of the 's' and 'i' cells and assign values of 1 and 2 respectively
[~, labels] = ismember(tmp, {'s', 'i'});

% Display the resulting label matrix
imagesc(labels)

% Use a custom colormap where empty cells are black, 's' are blue and 'i' are red
cmap = [0 0 0; 0 0 1; 1 0 0];
colormap(cmap)
```

And if we test this with 'Grid = {'s', []; 'i', []}'

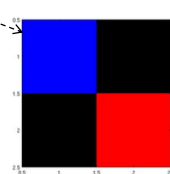
You can use 'ismember' to find where each label exists in your cell array . The second output will provide the index of the label . You can then use 'imagesc' with a custom colormap to display the result.

```
% Create a copy of Grid where the empty cells are replaced with ''
tmp = Grid;
tmp = cellfun(@(x) ['' x], Grid, 'UniformOutput', false);

% Locate all of the 's' and 'i' cells and assign values of 1 and 2 respectively
[~, labels] = ismember(tmp, {'s', 'i'});

% Display the resulting label matrix
imagesc(labels)

% Use a custom colormap where empty cells are black, 's' are blue and 'i' are red
cmap = [0 0 0; 0 0 1; 1 0 0];
colormap(cmap)
```



(d) Adding image

Fig. 6. Left: key phrases for predicting a type of edit; Right: actual post edit

## 6.1 The Impact on Collaborative Editing

Among 20,988,988 post-body edits in Stack Overflow, 13,487,086 (64.26%) of them are self edits, i.e., the post is edited by the post owner. This includes edits suggested by post viewers using post comments and edits coming from double checking post content by the post owners themselves. Fig. 7 shows an example of post comment which asks the post owner to add a screenshot for a clearer understanding of his question, and the post owner did so accordingly. The significant amount of self edits and users' interactions triggering self edits suggest that post owners could

2 What file it cannot find? Show a screenshot of how it looks (that message -- to see the context). After what operation this messages shows up? – [LazyOne](#) Aug 2 '16 at 16:55

---

I have added screenshots of the popup. I hope this will help. Thanks – [sanders](#) Aug 3 '16 at 7:02

Fig. 7. An example of the users' interaction triggering post self editing

appreciate the proactive post-edit recommendations which our approach supports. Because this proactive support can warn the post owners some potential quality issues in their posts before the posts are published, they have a better chance to get the posts right in the first place, rather than fixing the issues in an afterthought way. After all, the high-quality questions and answers could be better received by the community [33].

When a post with some quality issues has been posted anyway, our post-edit prediction can also be used to attract other users' attention to the problematic post. As our approach currently predicts only four types of edits, it may impact to which posts users allocate their attention. However, this is more like a limitation of our current approach implementation, rather than a methodological limitation. Furthermore, new editing patterns are likely to emerge over time. It takes time to accumulate sufficient instances for the deep learning model to learn the new editing patterns. Before that model update, our approach may impact whether the community should adopt an existing editing pattern or the emerging editing pattern.

Collaborative editing often provides a mechanism of legitimate peripheral participant for novice users [32]. In this context, our post-edit recommendation could provide a learning opportunity for novice users to "observe" the community's implicit quality assurance practices. This observation is rather impossible from the raw collaborative editing data or it would take a long time to learn through trial-and-error style of participation in collaborative editing activities. Having that said, there could also be some unintended consequences that are worth monitoring. For example, as our recommendation is not perfect (which is actually impossible for any machine learning based approach), novice users may accept some erroneous edit recommendations made by the tool due to the lack of experience and knowledge. This may result in low-quality post edits, which may in turn increase the approval working load of trusted users.

In any cases, our approach only recommends post edits, and human users always have final say on what do to with the recommendations. We do not envision that our approach will replace human decisions on the need of editing a post and how to edit the post. This is especially the case for the post edits that require deep natural language understanding of the post contents, which is beyond the capability of current machine learning techniques. But machine-learning based post-edit recommendation would assist the allocation of user attention and their editing decisions. On the other hand, as our evaluation shows, machine-learning based post-edit recommendation is not perfect. They could be erroneous or missed recommendations. Collaborative editing would be able to deal with machine's errors. Therefore, machine-learning based post-edit recommendation and human collaborative editing are complementary. The effective mechanism to blend machine-learning based post-edit recommendations and human collaborative editing is an interesting research direction to explore.

## 6.2 The Generalization of Our Approach and Findings

**6.2.1 The Generalization to Other Edits.** In Section 5.4, we report and analyze the performance of our CNN-based model for predict mid-level additive edits. However, it is important to note that our approach is not limited to additive edits. In fact, we conduct similar performance evaluation for other types of edits including replacing and deleting edits as those reported in Section 5.4. Table 3 shows that our model can achieve high precision (0.7977 ~ 0.9088), recall (0.7830 ~ 0.9739), and

type	#Posts	Precision	Recall	F1-score
replacing link	14,580	0.7977	0.7830	0.7903
replacing text format	15,671	0.8193	0.8383	0.8287
replacing code format	24,466	0.9088	0.9739	0.9402
deleting link	50,354	0.7828	0.7833	0.7830
deleting image	17,361	0.8942	0.8611	0.8773
deleting text format	35,244	0.8059	0.7898	0.7978
deleting code format	171,525	0.8576	0.8289	0.8430

Table 3. Performance of our model in replacing and deleting edits



Fig. 8. Examples of markdown applications in other sites

F1-score (0.7830 ~ 0.9402). These performance metrics have no significant difference from those for predicting addition edits. For replacing code format and deleting image, the performance is actually better than that of adding code format and adding image. Such results demonstrate the generalization of our model to other kinds of edits<sup>8</sup>. As our approach is a deep-learning based method, it is not applicable when the amount of certain type of post edit is too small. For example, there are only 595 replacing edits of images, and that is why we do not have the performance metric for replacing-image edits in Table 3.

**6.2.2 The Generalization to Other Sites.** There are many Q&A sites in the world, and some popular ones are summarized in an Wikipedia page [5]. Following this list, we manually check if some Q&A sites may adopt markup format similar to that in Stack Overflow. After filtering non-Q&A or closed Q&A sites, there are seven Q&A sites left which are founded in the recent decade i.e., the ones after 2008. These seven Q&A sites include *Jobstr*, *Zhihu*, *Ask.fm*, *Brainly*, *Quora*, *Sharecare*, *Stack Exchange*. According to our observation, 5 of these seven Q&A sites contain markup style formatting which allow users to attach list, header, image, hyperlinks in the posts. The other 2 sites (*Jobstr*, *Ask.fm*) do not adopt markup format.

This work examines collaborative editing data of Stack Overflow, and is based on the observed collaborative editing characteristics, we develop a deep learning based approach for predicting whether or not a post may need four types of edits. However, it is important to note that our data analysis method and deep learning approach are totally data driven, not tied to any specific collaborative editing or quality control process in community Q&A sites. Therefore, we would expect that our data analysis method and deep learning approach can be applied to other Q&A sites, where historical collaborative editing data is available.

Of course, the collaborative editing characteristics and the types of edits to predict are very likely to differ from one site to another. For sites which belong to Stack Exchange network, as these sites obey the same rule as Stack Overflow, our analysis and method can be directly extended to them. For other sites with also markup style formatting like Quora, the differences between these sites with Stack Overflow can be relatively small. Our analysis and approach may be easily adapted to these Q&A sites. But the differences can be considerably big between Stack Overflow and those

<sup>8</sup>We do not report the details of experiments results for replacing and deleting edits because the results and analysis are very similar to those reported for additive edits.

sites without markup formatting like *Jobstr* or *Ask.fm*. In such cases, a detailed empirical study of the collaborative editing characteristics is required to determine what edits can be predicted and what methods will be effective.

In this work, we confirm the prediction performance of our approach using historical collaborative editing data. But it is still an open question whether our approach can perform well in a large-scale, live deployment on Stack Overflow. Several issues require further investigation, such as how to integrate our recommendation in the Q&A and collaborative editing process, whether the recommended post edits will be accepted by the community in practice, and how they may impact the post owners and post editors' behavior. We leave these questions as our future work.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we conduct an empirical study of historical collaborative editing data on Stack Overflow to investigate the need for proactive quality assurance on Q&A sites. Our study analyzes various aspects of collaborating editing behaviors and results, including common editing types, users' attention on different types of edits over time, scale of changes involved in post edits, and patterns of concrete editing operations. Based on our empirical observation, we identify four types of post edits (adding code format, adding text format, adding hyperlinks and adding images) that occur frequently by collaborative editing, and at the same time, involve medium-scale of changes that are feasible to predict using machine learning techniques. A CNN-based post-edit recommendation approach has been developed and the CNN model has been trained using large-scale post editing data. Our evaluation through historical post editing data demonstrates the quality of the recommended post edits by our approach.

We discuss the potential benefits of our post-edit recommendation approach for post owners, post editors and novice users. However, deploying our approach on Stack Overflow may have complicated impacts on social process and collaborative editing, which deserve further studies in the future. We will ask novice and trusted users on Stack Overflow to try our tool in practice and analyze the feedback to gauge its design. We will also extend our approach to other online Q&A sites, similar to or different from Stack Overflow, to understand the generalizability of our data analysis method and our deep learning approach.

## ACKNOWLEDGMENTS

We appreciate the valuable comments from the anonymous reviewers. This project is partially supported by the FIT ECR seed grant in Monash University.

## REFERENCES

- [1] 2009. Answer technical questions helpfully. <https://codeblog.jonskeet.uk/2009/02/17/answering-technical-questions-helpfully/>. (2009). Accessed: 2018-03-01.
- [2] 2010. Write the perfect question. <https://codeblog.jonskeet.uk/2010/08/29/writing-the-perfect-question/>. (2010). Accessed: 2018-03-01.
- [3] 2017. The Objective Revision Evaluation Service. <https://ores.wikimedia.org/>. (2017).
- [4] 2018. Community norm. [http://communitymgt.wikia.com/wiki/Community\\_norm](http://communitymgt.wikia.com/wiki/Community_norm). (2018). Accessed: 2018-06-20.
- [5] 2018. Comparison of Q&A sites. [https://en.wikipedia.org/wiki/Comparison\\_of\\_Q&A\\_sites](https://en.wikipedia.org/wiki/Comparison_of_Q&A_sites). (2018). Accessed: 2018-06-20.
- [6] 2018. Deep Learning Tutorial. <http://deeplearning.net/tutorial/deeplearning.pdf>. (2018). Accessed: 2018-06-20.
- [7] 2018. How do I ask a good question? <https://stackoverflow.com/help/how-to-ask>. (2018). Accessed: 2018-03-01.
- [8] 2018. How do I write a good answer? <https://stackoverflow.com/help/how-to-answer>. (2018). Accessed: 2018-03-01.
- [9] 2018. Thanks a Million, Jon Skeet! <https://stackoverflow.blog/2018/01/15/thanks-million-jon-skeet/>. (2018). Accessed: 2018-03-01.
- [10] 2018. Training, test, and validation sets. [https://en.wikipedia.org/wiki/Training,\\_test,\\_and\\_validation\\_sets](https://en.wikipedia.org/wiki/Training,_test,_and_validation_sets). (2018). Accessed: 2018-06-20.



- [11] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning.. In *OSDI*, Vol. 16. 265–283.
- [12] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. 2013. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing* 17, 2 (2013), 76–81.
- [13] Maik Anderka, Benno Stein, and Nedim Lipka. 2012. Predicting quality flaws in user-generated content: the case of wikipedia. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 981–990.
- [14] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).
- [15] Lasse Bergroth, Harri Hakonen, and Timo Raita. 2000. A survey of longest common subsequence algorithms. In *String Processing and Information Retrieval, 2000. SPIRE 2000. Proceedings. Seventh International Symposium on*. IEEE, 39–48.
- [16] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
- [17] Chunyang Chen and Zhenchang Xing. 2016. Mining technology landscape from stack overflow. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 14.
- [18] Chunyang Chen and Zhenchang Xing. 2016. Towards correlating search on google and asking on stack overflow. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, Vol. 1. IEEE, 83–92.
- [19] Chunyang Chen, Zhenchang Xing, and Lei Han. 2016. Techland: Assisting technology landscape inquiries with insights from stack overflow. In *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. IEEE, 356–366.
- [20] Chunyang Chen, Zhenchang Xing, and Yang Liu. 2017. By the Community & For the Community: A Deep Learning Approach to Assist Collaborative Editing in Q&A Sites. *PACMHCI* 1, CSCW (2017), 32:1–32:21.
- [21] Chunyang Chen, Zhenchang Xing, and Ximing Wang. 2017. Unsupervised software-specific morphological forms inference from informal discussions. In *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 450–461.
- [22] Guibin Chen, Chunyang Chen, Zhenchang Xing, and Bowen Xu. 2016. Learning a dual-language vector space for domain-specific cross-lingual question retrieval. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 744–755.
- [23] François Chollet et al. 2015. Keras. (2015).
- [24] Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 69–78.
- [25] Alexander Genkin, David D Lewis, and David Madigan. 2007. Large-scale Bayesian logistic regression for text categorization. *Technometrics* 49, 3 (2007), 291–304.
- [26] Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*. 427–431.
- [27] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [28] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. 2042–2050.
- [29] Meiqun Hu, Ee-Peng Lim, Aixin Sun, Hady Wirawan Lauw, and Ba-Quy Vuong. 2007. Measuring article quality in wikipedia: models and evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM, 243–252.
- [30] Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *ICML*, Vol. 99. 200–209.
- [31] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification.. In *AAAI*, Vol. 333. 2267–2273.
- [32] Jean Lave and Etienne Wenger. 1991. *Situated learning: Legitimate peripheral participation*. Cambridge university press.
- [33] Guo Li, Haiyi Zhu, Tun Lu, Xianghua Ding, and Ning Gu. 2015. Is It Good to Be Like Wikipedia?: Exploring the Trade-offs of Introducing Collaborative Editing Model to Q&A Sites. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 1080–1091.
- [34] Yi Lian, Pengfei Liu, Hongyuan Huo, Hu Zhang, Tiejun Cui, and Peng Du. 2016. Inversion of FeO and TiO<sub>2</sub> content using microwave radiance simulation based on Chang-E2 passive microwave radiometer data. In *2016 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2016, Beijing, China, July 10-15, 2016*. 4319–4322.

- [35] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-task Learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 2873–2879.
- [36] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2857–2866.
- [37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
- [38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
- [39] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature Visualization. *Distill* 2, 11 (2017), e7.
- [40] Luca Ponzanelli, Andrea Mocci, Alberto Bacchelli, Michele Lanza, and David Fullerton. 2014. Improving low quality stack overflow post detection. In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 541–544.
- [41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533.
- [42] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 959–962.
- [43] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.
- [44] Besiki Stvilia, Michael B Twidale, Les Gasser, and Linda C Smith. 2005. Information quality discussions in Wikipedia. In *Proceedings of the 2005 international conference on knowledge management*. Citeseer, 101–113.
- [45] Besiki Stvilia, Michael B Twidale, Linda C Smith, and Les Gasser. 2008. Information quality work organization in Wikipedia. *Journal of the Association for Information Science and Technology* 59, 6 (2008), 983–1001.
- [46] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. 2009. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence* 23, 04 (2009), 687–719.
- [47] Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727* (2016).
- [48] Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. 2016. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 51–62.
- [49] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923* (2017).
- [50] Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 901–911.
- [51] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

Received April 2018; revised July 2018; accepted September 2018