# By the Community & For the Community: A Deep Learning Approach to Assist Collaborative Editing in Q&A Sites

CHUNYANG CHEN, Nanyang Technological University, Singapore
ZHENCHANG XING, Australian National University, Australia
YANG LIU, Nanyang Technological University, Singapore

Community edits to questions and answers (called post edits) plays an important role in improving content quality in Stack Overflow. Our study of post edits in Stack Overflow shows that a large number of edits are about formatting, grammar and spelling. These post edits usually involve small-scale sentence edits and our survey of trusted contributors suggests that most of them care much or very much about such small sentence edits. To assist users in making small sentence edits, we develop an edit-assistance tool for identifying minor textual issues in posts and recommending sentence edits for correction. We formulate the sentence editing task as a machine translation problem, in which an original sentence is "translated" into an edited sentence. Our tool implements a character-level Recurrent Neural Network (RNN) encoder-decoder model, trained with about 6.8 millions original-edited sentence pairs from Stack Overflow post edits. We evaluate our edit assistance tool using a large-scale archival post edits, a field study of assisting a novice post editor, and a survey of trusted contributors. Our evaluation demonstrates the feasibility of training a deep learning model with post edits by the community and then using the trained model to assist post editing for the community.

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

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

**ACM Reference Format:**
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. *Proc. ACM Hum.-Comput. Interact.* 1, 2, Article 32 (November 2017), 21 pages.
https://doi.org/10.1145/3134667

## 1 INTRODUCTION

Stack Overflow is the most popular Question and Answering (Q&A) site for software programming. It hosts a community of millions of developers who share and learn software programming knowledge. An important reason for the popularity of Stack Overflow is that the content of Stack Overflow posts has been well maintained by the community. After a question or answer (referred to as a post in Stack Overflow) is posted, it can be self-edited by the post owner (i.e., the question asker or

Authors' addresses: Chunyang Chen (chen0966@e.ntu.edu.sg), School of Computer Science and Engineering, Nanyang Technological University, Singapore; Zhenchang Xing (zhenchang.xing@anu.edu.au), Research School of Computer Science, Australian National University, Canberra, Australia; Yang Liu (yangliu@ntu.edu.sg), School of Computer Science and Engineering, Nanyang Technological University, Singapore.

2573-0142/2017/11-ART32 $15.00
https://doi.org/10.1145/3134667

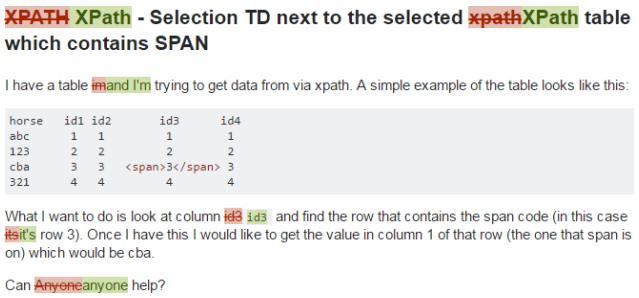


Fig. 1. An example of post edit in Stack Overflow

answerer) and/or collaboratively edited by some post editors[1] (i.e., users other than the post owner). In Stack Overflow, users with 2000+ reputation scores are considered as trusted contributors. Other users are considered as novice contributors. Edits by post owners or trusted contributors will be directly accepted. While Edits by novice contributors will be accepted only if they are approved by three trusted contributors to guarantee the site quality. As of December 11, 2016, 12.3 million posts (i.e., about 37% of all 33.5 million posts) have been edited at least once, and there have been more than 21.8 million accepted post edits[2] (see Figure 1 for an example).

Studies [21, 24] show that post editing improves the content quality. By analyzing the Stack Overflow's archival post edits, we find that some post editing involves complex revisions such as adding or removing sentences, code snippets or web resources. But there are also large numbers of post edits which involve small sentence revisions, such as correcting misspellings or grammar errors in a sentence, or changing the word or sentence format according to the website- or domain-specific convention. For example, Figure 1 shows a post edit with corrections of misspellings, grammar errors and formatting. Table 1 lists more examples of different kinds of small sentence edits. Among all 8.5 million post edits annotated with comments in Stack Overflow, 2.1 millions (24.7%)[3] of them contain keywords like "spell", "grammar" or "format" in the comments of post edits (See Section 3.3). Our survey of trusted contributors in Stack Overflow who has made large numbers of post edits to their own or others' posts further confirms the core users' attention to small sentence edits to improve the content quality (See Section 7).

The presence of a large number of small sentence edits and the attention of trusted contributors to such edits motivate us to investigate a post edit assistance tool for identifying minor textual issues in posts and recommending proper corrections, such as those shown in Figure 1 and Table 1. Our analysis of "who edited posts" (see Section 3.2) reveals that 65.77% of all accepted post edits are self-edits by the post owners and 34.23% are collaborative edits by some post editors. Among collaborative edits, 89% are done by trusted contributors and 11% are done by novice contributors. Therefore, an edit assistance tool will not only help post owners reduce minor textual issues before posting their questions and answers, but also help post editors improve their editing efficiency. Furthermore, the identified issues together with the recommended corrections will help novice post editors learn community-adopted editing patterns.

To determine the required tool support, we conduct a formative study to understand the categories, editing reasons and scale of changes of post edits in Stack Overflow. As shown in Table 1, some sentence edits (row 1-5) is to fix general language issues, such as misspellings, grammar errors, sentence formatting issues. Some general language issues can be resolved by spell checking tools

[1]https://stackoverflow.com/help/editing
[2]Hereafter, post edits refer to accepted post edits, unless otherwise stated.
[3]This number could be highly underestimated as there are many similar words which are not taken into account, such as "typo", "wording", "indentation", etc. See Figure 3.

| | Original Sentence | Edited Sentence | Editing Reason |
|---|---|---|---|
| 1 | I need to get the last char of a srting. | I need to get the last char of a string. | Spelling |
| 2 | Is it possible to this? | Is it possible to do this? | Grammar |
| 3 | Can you suggest me | Can you suggest me? | Punctuation |
| 4 | Any ideas how to fix it ? | Any ideas how to fix it? | Space |
| 5 | how can I accomplish this? | How can I accomplish this? | Capitalization |
| 6 | My problem is the when I click on the OK button, nothing happens. | My problem is the when I click on the ` OK ` button, nothing happens. | Annotation |
| 7 | EDIT: Sorry, I should have inserted the term "cross browser" somewhere. | **EDIT**: Sorry, I should have inserted the term "cross browser" somewhere. | Annotation |
| 8 | 1) How to connect to SVN server from java? | <li>How to connect to SVN server from java?</li> | HTML |
| 9 | I want an Apple script that refreshes a certain song in iTunes from file. | I want an AppleScript that refreshes a certain song in iTunes from a file. | Spelling |
| 10 | I am trying to parse a set of xml files. | I am trying to parse a set of XML files. | Capitalization |
| 11 | Use javascript function isNaN. | Use JavaScript function isNaN. | Capitalization |

Table 1. Examples of small sentence edits in Stack Overflow

like LanguageTool[4]. However, due to the lack of domain knowledge, general spell checkers often make mistakes, such as changing *css* (acronym for Cascading Style Sheets) to *CBS*, *json* (acronym for JavaScript Object Notation) to *son*, or *li* (an HTML tag) to *Ali*. Furthermore, there are many sentence edits beyond general language issues, such as site-specific formatting style (row 6-8) and domain-specific naming convention (row 9-11). For example, when mentioning a User Interface (UI) component (e.g., the *OK* button), the community prefers to quote the component name. When listing items, the community prefers to use Markdown language[5] (e.g., HTML <li>). Software-specific names should be mentioned according to the domain-specific convention, like *AppleScript* rather than *Apple script*.

Considering the devisity of words and formats involved in sentence edits, it would require significant manual effort to develop and validate a complete set of rules for representing editing patterns. For example, OK should be quoted when it refers to an UI component, but it will not be quoted in many other contexts. It is impractical to enumerate all such cases.

Alternatively, machine learning techniques can be used to learn sentence editing patterns from archival post edits. In this work, we formulate sentence editing as a machine translation problem, in which an original sentence is "translated" into an edited sentence. We solve the problem using the RNN encoder-decoder model [9]. As we observe that the majority of post edits involves only character-level changes of post content, we decide to use a character-level RNN model. Furthermore, we develop a normalization method for preprocessing and postprocessing domain-specific rare words (e.g., URLs, API calls, variable names) in posts before and after the RNN model training. To train the RNN model, we develop a text differencing algorithm to collect a large dataset of original-edited sentence pairs from post edits, such as those shown in Table 1[6]. The trained RNN model detects and encodes editing patterns from original-edited sentence pairs, thus removing the need for prior manual editing rule development.

We evaluate the quality of sentence editing recommendations by our approach with large-scale archival post edits. Our approach outperforms the LanguageTool (a rule-based proof-reading tool that has been developed for over 10 years) and a phrase-based SMT technique [31] specially designed for sentence correction. We also conduct a field study in which the first author acts as a novice post editor that has little post editing experience. Based on the sentence editing recommendations by our tool, he edits 39 posts, 36 of which have been accepted. That is, for each accepted post edit, at least three trusted contributors consider that the edit has significantly improved the post quality. Finally, we collect 58 replies in a survey of trusted contributors who has made large numbers of post edits. These replies shed the light on the trusted contributors' opinions and suggestions for our tool.

[4]https://languagetool.org/

[5]http://stackoverflow.com/editing-help

[6]Edits in Table 1 are underlined for the explanation purpose. The RNN model does not know such sentence edits in advance.

We make the following contributions in this paper:

- We conduct a formative study of post editing in Stack Overflow and a survey of trusted contributors' attention to small sentence edits, which help us understand the need for an edit assistance tool and who can benefit from the tool in which scenarios.
- We develop a RNN-based edit assistance tool for identifying spelling, grammar and formatting issues in Stack Overflow posts and recommending corrections. The tool learns editing patterns from big data of accepted archival post edits and makes special considerations of the data characteristics of post edits. To the best of our knowledge, our dataset is the largest dataset that has ever been collected for sentence editing task.
- We evaluate our approach from three perspectives, including the quality of sentence editing recommendations using large-scale archival post edits, the ability to assist a novice post editor in editing unfamiliar posts, and the feedbacks of 58 trusted contributors on our tool.

## 2 RELATED WORK

Wikipedia-like collaborative editing has been adopted in Stack Overflow to ensure the content quality on the site [24]. Existing studies investigate the general implications of collaborative editing in large-scale social computing system. For example, Vargo and Matsubara [35] investigates how different users behave in a system that contains gamified motivations for contributing edits. Li et al. [21] examines the trade-offs between managing quality and encouraging contributions. They show that collaborative editing significantly and robustly improves the votes received, the number of answers received for questions, the likelihood of questions getting accepted answers, and the likelihood of answers being accepted, but it barely decreases the users' subsequent contributions. Different from existing studies, our formative study of post editing in Stack Overflow examines who edits posts and the categories, editing reasons and scale of changes of post edits, which reveals the need for and the benefits of a specific tool support to assist post editing.

Machine learning techniques have been developed to assist collaborative editing in online communities. For example, Li et al. [20] trained a classifier to predict whether a post needs an edit. Wikipedia develops the Objective Revision Evaluation Service (ORES) [1] to separate blatant vandalism from well-intentioned changes in the edit review process. Our edit assistance tool is different as it works at a much finer-grained level. Instead of classifying posts or post edits, our tool identifies spelling, grammar and formatting issues in sentences and recommends sentence edits to fix the identified issues.

Much research has been carried out to correct spelling and grammar errors using machine learning techniques. Junczys-Dowmunt and Grundkiewicz [16] adopt phrase-based Statistical Machine Translation (SMT) method for automatic grammar error correction. Mizumoto and Matsumoto [26] and Yuan et al. [41] propose a ranking method to rank the SMT's recommendations of grammar error corrections. However, SMT focuses on source-target phrase pairs without effective modeling sentence context. Furthermore, SMT consists of components that are trained separately and then combined [18]. Compared with traditional SMT methods, Neural Machine Translation (NMT), such as RNN-based methods [25, 38], models sentence context and all the NMT components are trained jointly to maximize the performance. Especially, NMT is appealing for Grammatical Error Correction (GEC) tasks as it may correct erroneous sentences that have not been seen in the training data. Therefore, our edit assistance tool adopts a RNN encoder-decorder model [9].

Other researchers also adopt deep learning methods for grammatical error detection [22, 34] and sentence correction [10, 39, 40]. Although these methods obtain better performance than traditional SMT methods, they cannot effectively deal with the three data characteristics of Stack Overflow post edits. First, existing methods are designed for general English text. They cannot

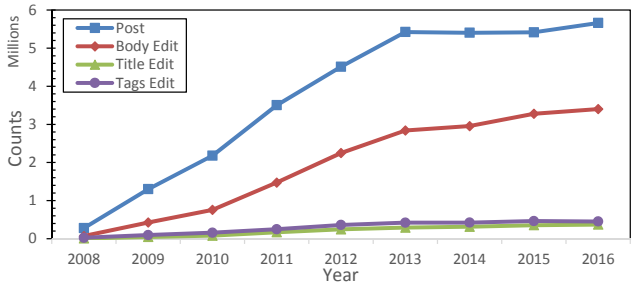


Fig. 2. The numbers of posts and three kinds of post edits in each year in Stack Overflow

handle domain-specific rare words, such as URLs, API calls and variable names in Stack Overflow posts, due to the rareness of such domain-specific words and the much noise these words introduce to the sentence correction model. But simply throwing away these domains-specific rare words will negatively influence the quality of sentence editing model because it breaks the sentence integrity. Second, existing methods consider only general language errors that are only part of post edits in Stack Overflow. They cannot handle the format change such as HTML markdown and domain-specific naming convention. Third, existing methods deal with word- or phrase-level correction, but the majority of post edits involves only minor changes of post content at character level.

Another big limitation of existing NMT methods is the lack of editing data for training deep learning methods. The Helping Our Own task [12] contains 1264 edits for model training and 1057 edits for testing. The CoNLL-2014 task [30] on GEC contains 57151 edited sentence pairs for training and 1312 for testing which are collected from essays written by students at National University of Singapore. To mitigate the lack of data, Liu et al. [13, 22] propose different ways of artificial error generation, but such generated edits may differ from the real data. We are the first to leverage the big data of post edits in Q&A sites to train an edit assistance tool for sentence correction. And our training data is 100 times larger than the largest dataset of existing work [39].

## 3 FORMATIVE STUDY OF POST EDITING IN STACK OVERFLOW

Stack Overflow is selected as our study subject, not only because of its popularity and large user base [7, 8], but also because it is a well-known online Q&A site which supports collaborative editing [21, 24]. We download the latest data dump[7] of Stack Overflow which contains 33,402,449 posts (including 12,865,526 questions and 20,536,823 answers) and all post edits since the launch of Stack Overflow in 2007 to December 11, 2016. Based on this large dataset, we carry out an empirical study of post edits in Stack Overflow to understand the characteristics of post editing in Stack Overflow 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 [21]. Question-title and post-body editing are of the same nature (i.e., sentence editing), while question-tags editing is to add and/or remove the set of tags of a question.

As of December 11, 2016, there have been in total 21,759,565 post edits. Among them, 1,857,568 (9%) are question-title edits, 2,622,955 (12%) are question-tag edits, and the majority of post edits (17,279,042 (79%)) are post-body edits. The tags of 2,246,658 (17.5%) questions, the titles of 1,630,933 (12.7%) questions, and the body of 11,205,822 (33.5%) posts have been edited at least once. Figure 2

[7]https://archive.org/download/stackexchange

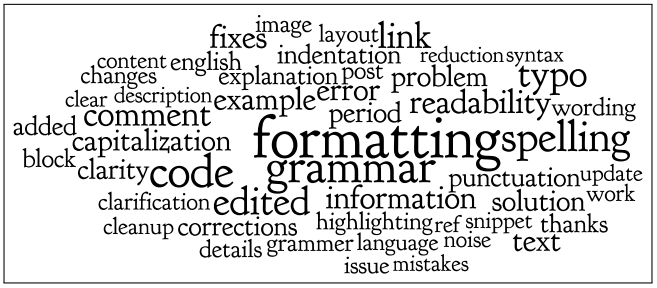


Fig. 3. The word cloud of the top 50 most frequent words appearing in edit comments (the words are aligned alphabetically from left to right)

shows that the number of post edits increases as the number of posts increases over time. The number of question-title and question-tag edits increase slowly, while the number of post-body edits increase in a similar rate as posts increase.

Overall, post-body and question-title edits make up the majority of post edits. Compared with adding/removing question tags, post-body and question-title editing are more complex (further studied in the next question). Therefore, we focus on post-body and question-title edits in this work. Hereafter, post edits refer to post-body and question-title edits, unless otherwise stated.

### 3.2 Who edited posts?

Among all 19,136,610 post-body and question-title edits, 12,586,199 (65.77%) are self-edits by the post owners, 5,806,880 (30.34%) are collaborative edits by trusted contributors, and 743,531 (3.89%) are collaborative edits by novice contributors. This data suggests that an edit assistance tool may benefit the Stack Overflow community from three perspectives.

First, the tool can highlight the textual issues in the posts that the post owners are creating and remind them fixing the issues. This helps to get the posts right in the first place and reduce the need for after-creation editing. Second, trusted contributors make up only 9% of Stack Overflow users but they take up 30.34% of post editing tasks. An edit assistance tool that recommends sentence edits can improve the editing efficiency of trusted contributors. Third, the approved edits by novice editors is rather low. This could be because novice editors do not have enough experience and courage to edit others' posts, or their edits are incorrect and rejected by trusted contributors. It also suggest that an edit assistant tool would be beneficial if novice editors would like to use small edits as a mechanism for legitimate peripheral participation, because an edit assistance tool's edit recommendations can serve as a "tutor" to teach novice contributors the community-adopted editing patterns. This may help to on-board novice contributors in Stack Overflow and improve the quality of their post edits.

### 3.3 What are post edits about?

According to the Stack Overflow's edit guidelines[8], there are four common types of edits: 1) to fix grammatical or spelling mistakes, 2) to clarify the meaning of a post without changing it, 3) to correct minor mistakes or add addendums/updates as the post ages, 4) to add related resources or hyperlinks.

We analyze the comments of post edits to understand what post edits are about and whether they align with the community guideline. In Stack Overflow, when users finish editing a post, they can add a short comment to summarize the post edit. We collect all post-edit comments and apply

[8]http://stackoverflow.com/help/privileges/edit

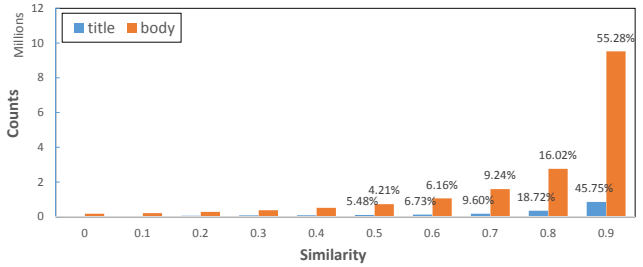


Fig. 4. The count of original-edited post title and body in different similarity range

standard text processing step to post-edit comment such as removing punctuation, lowercasing all characters, excluding stop words, stemming. Then we count the word frequencies and we display the top 50 most frequent words in a word cloud [36] in Figure 3. The font size of a word depends on the word frequency[9] in our dataset.

According to these comments, it can be seen that post edits have covered four common edit types in the guideline, such as "spelling", "typo", "grammar" for the type (1), "clarification", "details", "explanation' for type (2), "fixes", "errors", "changes" for type (3), and "links", "information", "image" for type (4). Apart from them, there are also some other keywords, such as "formatting", "indentation", "highlighting", "capitalization", and "readability". Although formatting, grammar and spelling types of edits are not about post mistakes or additional/updated resources, they are still crucial for readers as these edits can make the posts easier to read and understand. Table 1 lists some examples of formatting, grammar and spelling edits. In fact, the word cloud shows that formatting, grammar and spelling types of edits happen more frequently than other types of edits.

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

When editing a post, users may change some words, delete a sentence, add some sentences or code snippets according to different goals or context. To understand the scale of changes that post edits involve, we measure the similarity between the original post before a post edit and the edited post after the edit. Given a post edit, let *original* and *edited* be the text content (question title or post body) of the original and edited post. We use the text-matching algorithm [5] to find the character-level Longest Common Subsequence (LCS) between the *original* and *edited* content. We measure the similarity between the original and edited post as:

$$similarity(original, edited) = \frac{2 * N_{match}}{N_{total}} \quad (1)$$

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

As shown in Figure 4, among the 17,279,042 post-body edits, the original and edited post body of 55.28% edits are very similar with the similarity score between 0.9 and 1. 16.01% of them are between 0.8 to 0.9. Similarly, among 1,857,568 question-title edits, 64.47% of them are between 0.8 and 1. This indicates that most of the post edits involve only minor scale of changes of question titles and post bodies.

**Summary:** Our study shows that there is a large number of formatting, grammar and spelling types of post edits that involve minor scale of changes of post content. Assisting these types of post edits would benefit the post owners, trusted contributors and novice contributors from different

[9]We normalize the frequency in logarithm to avoid the extreme large word size in the figure.

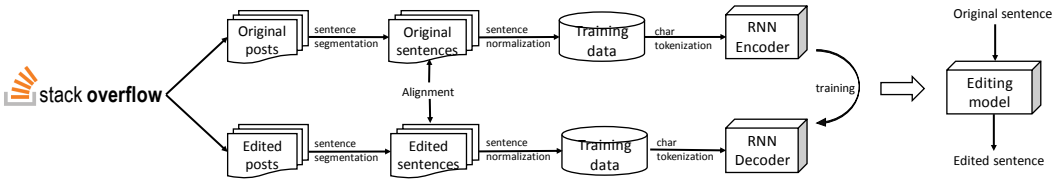


Fig. 5. The overall workflow of our approach

perspectives. To be effective, the edit assistance tool must be able to handle the diversity of post editing patterns and the character-level changes that post edits often involve.

## 4 ASSISTING SENTENCE EDITING

Based on the empirical observation of post edits in Stack Overflow, we focus our effort in this work on sentence edits that correct minor textual issues in a sentence, such as those shown in Figure 1 and Table 1. Considering the diversity of post editing patterns, it would require significant effort to manually develop a complete set of editing patterns which is time-consuming and error-prone. Therefore, we propose a deep-learning based approach which can automatically detect and encode sentence editing patterns from large numbers of archival post edits using a RNN encoder-decoder model [9].

### 4.1 Approach Overview

The overall workflow of our approach is shown in Figure 5. Our approach aligns the sentences of the original and edited posts for preparing a large corpus of original-edited sentence pairs for model training (Section 4.2). To reduce the negative effects of domain-specific rare words on the model, our approach normalizes the sentences by replacing domain-specific rare words (such as URLs, APIs, variable names) by special symbols (Section 4.4). To model character-level changes of post edits like formatting, grammar, spelling, our approach trains a character-level RNN encoder-decoder model with a large parallel corpus of original-edited sentence pairs (Section 4.3). The trained sentence editing model can identify minor textual issues (both general and domain-specific) in an original sentence and recommend corrections of these issues. Next, we will describe the core steps of our approach.

### 4.2 Collecting the Corpus of Original-Edited Sentence Pairs

A post may have been 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 and 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. The edited part can be question title or post body. As this work focuses on sentence edits, we remove code snippets by HTML tags "$< code >$" from the collected content. Then, we split the content into sentences by punctuation such as ".", "?", "!" and ";".

The Algorithm 1 aligns the sentence list *oList* from the original content and the sentence list *eList* from the edited content. It finds the LCS of matched (lines 4-10) and unmatched-but-similar-enough (lines 11-23) sentences between the two input sentence lists. For the two unmatched sentences, $computeSimilarity()$ computes the char-level LCS of the two sentences [5] and measures the sentence similarity using the Eq. 1. For a sentence in the *oList*, if the similarity score $largestScore$ of the most similar sentence in the *eList* is above a threshold $sim\_threshold$, the two sentences are aligned as a pair of original-edited sentences. Similarity threshold should be set to achieve a

**ALGORITHM 1:** Collect original-edited sentence pairs from post edits

**Input:** Two sentence lists *oList* (**o**riginal) and *eList* (**e**dited)
**Output:** A list of original-edited sentence **p**airs *pList*

```
Init oIndex ← 0, eIndex ← 0;
while oIndex < oList.length && eIndex < eList.length do
    Init largestScore ← −1, topPosition ← −1;
    for i ∈ [eIndex, eList.length-1] do
        if oList[oIndex] == eList[i] then
            eIndex = i + 1;
            largestScore = 1;
            break;
        end
    end
    if largestScore! = 1 then
        for i ∈ [eIndex, eList.length-1] do
            similarity = computeSimilarity(oList[oIndex], eList[i]);
            if similarity > largestScore then
                largestScore = similarity;
                topPosition = i;
            end
        end
        if largestScore > sim_threshold then
            pList.append([oList[oIndex], eList[topPosition]]);
            eIndex = topPosition + 1;
        end
    end
    oIndex = oIndex + 1;
end
```

balanced precision and recall for sentence alignment, so we experimentally set the threshold at 0.8 in this work. The algorithm outputs all aligned original-edited sentence pairs.

From the post edits before Dec 11, 2016, we collect 13,806,188 sentence pairs. But there are two common problematic kinds of sentence pairs in the dataset. First, some sentence pairs are code snippets which are not enclosed inside < *code* > HTML tag. Such code-snippet sentences are not in the scope of our study. We exclude code-snippet sentences if sentences contain programming constructs such as "{", "}", "=", "if(", "for(", "while(". Second, sometimes a post is edited by one user, but then is edited back into its original content by another user. We cannot decide which one of the edits is more suitable. Therefore, we exclude such sentence pairs. After post-processing, 7,545,979 sentence pairs are left, which are used to train the RNN encoder-decoder model for automatic sentence editing.

### 4.3 Character-Level RNN Encoder-Decoder Model

Recurrent Neural Network (RNN) is a class of neural networks where connections between units form directed cycles. Due to its nature, it is especially useful for tasks involving sequential inputs such as speech recognition [14] and sentence completion [25]. Compared with traditional n-gram language model [6], a RNN-based language model can predict a next word by preceding words with variant distance rather than a fixed number of words. This makes it possible to model long-distance dependencies in the sentence.

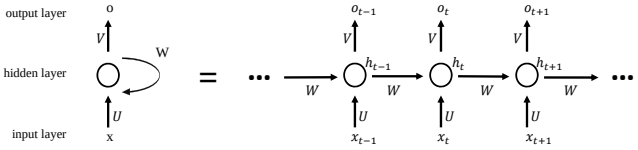


Fig. 6. Basic RNN model

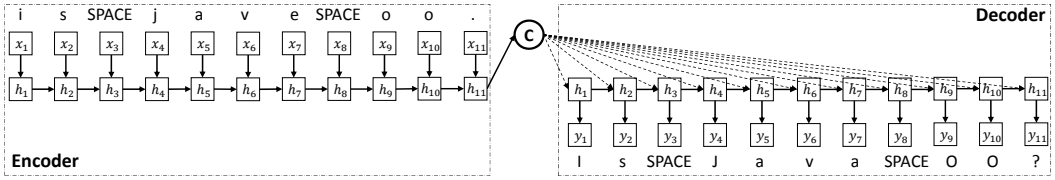


Fig. 7. The RNN encoder-decoder model

The architecture of a basic RNN model includes three layers. An input layer maps each word to a vector using word embedding or one-hot word representation. A recurrent hidden layer recurrently computes and updates a hidden state after reading each word. An output layer estimates the probabilities of the next word given the current hidden state. Figure 6 shows the unfolding in time of the RNN's forward computation. At time step $t$, it estimates the probability of the next word $P(w_{t+1}|w_1, ..., w_t)$ by three steps. First, the current word $w_t$ is mapped to a vector $x_t$ by the input layer.

$$x_t = input(w_t) \tag{2}$$

Then, the hidden layer generates the hidden state $h_t$ according to the previous hidden state $h_{t-1}$ and the current input $x_t$

$$h_t = h_{t-1} * W + x_t * U \tag{3}$$

where $W$, $U$ are parameters inside the neural network. Finally, the $P(w_{t+1}|w_1, ..., w_t)$ is predicted according to the current hidden state $h_t$:

$$P(w_{t+1}|w_1, ..., w_t) = g(h_t) \tag{4}$$

where the function $g$ produces valid probabilities. During model training, the parameters are learned by backpropagation [37] with gradient descent to minimize the error rate.

More complex RNN-based models have been developed for more complex NLP tasks. For example, the RNN encoder-decoder model [9] is commonly adopted for machine translation tasks. This model includes two RNNs as its main components: one RNN to encode a variable-length sequence into a fixed-length vector representation, and the other RNN to decode the given fixed-length vector representation into a variable-length sequence. From a probabilistic perspective, this model is a general method to learn the conditional distribution over a variable-length sequence conditioned on yet another variable-length sequence, i.e., $p(y_1, ..., y_{T'}|x_1, ..., x_T)$. The length of the input $T$ and output $T'$ may differ.

The architecture of our character-level RNN encoder-decoder model is shown in Figure 7. The example is to edit "is jave oo." to "Is Java OO?" in which "OO" is the abbreviation of "Object Oriented". The encoder is a basic RNN model that reads each character of an original sentence $x$ sequentially. This work focuses on sentence edits that involve many character-level changes such as misspellings, capitalizations, annotations. Furthermore, the character-level model can avoid the out-of-vocabulary problem [4, 23] because there are countless words, but only limited characters. Therefore, we use the character-level RNN model instead of the normal word-level one. As the model reads each character sequentially, the hidden state of the RNN encoder is updated according

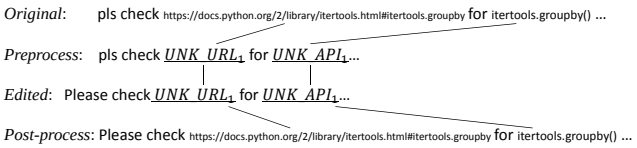


Fig. 8. The process to deal with domain-specific rare words

to Eq. 3. After reading the end of the the input, the hidden state of the RNN encoder is a vector $c$ summarizing the whole input original sentence.

The decoder of the model is another RNN which is trained to generate the output edited sentence by predicting the next word $y_t$ given the hidden state $h_{(t)}$. Unlike the basic RNN model in Figure 6, $y_t$ and $h_t$ are not only conditioned on $y_{t-1}$ but also on the summary vector $c$ of the input sentence. Hence, the hidden state of the decoder at time $t$ is computed:

$$h_t = f(h_{t-1}, y_{t-1}, \mathbf{c}) \tag{5}$$

and the conditional distribution of the next character is

$$P(y_t|(w_1, ..., w_{t-1}), \mathbf{c}) = g(h_t, y_{t-1}, \mathbf{c}) \tag{6}$$

for the given activation functions $f$ and $g$. The two RNN components of the RNN encoder-decoder model are jointly trained to maximize the conditional log-likelihood

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y_n|x_n) \tag{7}$$

where $\theta$ is the set of the model parameters and each $(x_n, y_n)$ is a pair of original-edited sentences from the training corpus.

### 4.4 Normalizing Domain-Specific Rare Words

The performance of deep learning models heavily depends on the quality of the training data. In particular, our RNN-based model relies on patterns of character sequences. However, in domain-specific Q&A text like Stack Overflow discussions, many terms are problem-specific or external resources, such as URLs of online documents (e.g., *http://support.microsoft.com/kb/299853*), API calls (e.g., *document.getElementById('whatever')*) and variable names (e.g., *pages[0]*). According to our observation, these specific terms usually have few errors because developers are more careful and sensitive when mentioning API/variable names than other general text. For the URL links, most of them are put into the text by copy-paste which rarely lead to errors.

However, when dealing with such special character sequences, the RNN encoder-decoder model cannot learn meaningful patterns effectively due to their rareness and diversity in text. Furthermore, these problem-specific and resource terms will negatively influence the quality of sentence editing model because they introduce noise to other normal words. But simply throwing away these domains-specific rare words will also negatively influences the quality of sentence editing model because it breaks the sentence integrity. Therefore, we normalize mentions of such domain-specific rare words in the training sentences to simplify the model complexity.

The normalization process first detects mentions of domain-specific rare words by regular expressions. Each detected mention in the original and edited sentence will be marked by a unique special symbol. For Stack Overflow sentences, we develop regular expressions for detecting URLs by *http://* or *https://*, API calls by API conventions like *a.b(c)* (Java/Python) or *a::b(c)* (C/C++), and array/list variables by *a[]*. As the example in the Figure 8 shows, *https://docs.python.org/2/library/itertools.html#itertools.groupby* and *itertools.groupby()* are marked as $UNK\_URL_1$ and $UNK\_API_1$ after normalizing the training

original-edited sentences. $UNK\_URL_{index}$, $UNK\_API_{index}$, or $UNK\_VARIABLE_{index}$ are special symbols where *index* is the unique index of an URL/API/variable in the training corpus.

The normalized sentence pairs are used to train the RNN encoder-decoder model. Given an original sentence to be edited, the trained model will change it into an edited sentence. The special symbols are then mapped back to the original domain-specific words in a post-processing step.

### 4.5 Implementation

Slightly different from the model in Figure 7, our implementation of the RNN encoder-decoder model consists of 3 hidden layers, i.e., deeper model structure for better learning. In each hidden layer, there are 1024 hidden units to store the hidden states. We implement our model based on the Tensorflow [2] framework and train the model in a Nvidia M40 GPU (24G memory) for about 6 days.

## 5 THE QUALITY OF RECOMMENDED SENTENCE EDITS

Our edit assistance tool aims to help post owners and editors edit posts by identifying textual issues in sentences and recommending small sentence edits for correcting these issues. The quality of recommended sentence edits will affect the adoption of the tool by the community. In this section, we use randomly selected 377,298 original-edited sentence pairs from archival post edits to evaluate the quality of recommended sentence edits by our tool.

### 5.1 Dataset

From the accepted 19,136,610 post edits, we collect 7,545,979 original-edited sentence pairs. We randomly take 6,791,381 (90%) of these sentence pairs as the training data, 377,298 (5%) as the development data to tune model hyperparameters, and 377,298 (5%) as the testing data to evaluate the quality of recommended sentence edits by our tool.

### 5.2 Baseline Methods

Apart from our own model, we take another two methods as baselines for comparison. One baseline is the LanguageTool[10], an open source proof-reading program for more than 20 languages. This tool's style and grammar checker is rule-based and has been developed for over 10 years. The other baseline is the phrase-based SMT model specifically designed for sentence correction [31]. We use the same training data to train the SMT model.

### 5.3 Evaluation metric

Our task can be regarded as a domain-specific GEC task, as it deals with the site- and domain-specific formatting, grammar and spelling knowledge. Therefore, we adopt GEC metrics for evaluating our approach.

Precision and recall have been traditionally used to evaluate the performance of GEC approaches [11, 12]. Given a sentence, precision measures the percentage of edits suggested by a tool that are correct, and recall measures the percentage of correct edits that are suggested by the tool. Precision and recall require manually-annotated gold-standard edits, such as insert, deletion, replacement, tense change, etc., in the sentences [30]. For example, the underlined text in the reference sentences in Table 2 are manually annotated gold-standard changes for the corresponding original sentences. However, the difficulty of defining error types and the disagreement between annotators often challenge the annotation validity as a gold standard [33]. This is especially the case

[10]https://languagetool.org/

| | | |
|---|---|---|
| **Original Sentence** | how do i make a file handle from a file path specified on the command line? | Would you recommend Java/J2EE, .Net / Erlang ? |
| **Edits by a tool** | How do I make a file handle from a file path specified on the command line? | Would you recommend Java/J2EE, .NET / Erlang ? |
| **Reference** | How do I make a filehandle from a file path specified on the command line? | Would you recommend Java/J2EE, .NET or Erlang? |
| **GLEU score** | 48.36 | 27.08 |

Table 2. GLEU score for different edits

for our data, considering the large number of techniques and concepts that have been discussed in Stack Overflow and the varieties of sentence edits that have been applied.

In the GEC field, recent released shared tasks have prompted the development of GLEU [27, 28] (Generalized Language Understanding Evaluation[11]) for evaluating GEC approaches. GLEU is a customized metric from the BLEU (BiLingual Evaluation Understudy) [32] score which is widely used to evaluate the machine-translation quality. It is independent of manual-annotation scheme and requires only reference sentences (without annotations of gold-standard edits). Recent studies [29, 33] show that GLEU has the strongest correlation with human judgments of GEC quality and effort, compared with precision and recall.

Therefore, we adopt GLEU in our evaluation. We regard an original sentence as the source sentence ($S$), the edited sentence by Stack Overflow user as the reference sentence ($R$), and the edited sentence generated by a GEC tool as candidate sentence ($C$). GLEU score measures how close a candidate sentence generated by the tool is to the reference sentence edited by human, with respect to the source sentence. Intuitively, GLEU awards the overlap between $C$ and $R$ but not in $S$, and penalizes n-grams in $S$ that should have been changed but are not and n-grams found in $S$ and $C$ but not in $R$. It also captures the sentence length and the fluency and adequacy of n-gram overlap. GLEU is computed as:

$$GLEU(S, R, C) = BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \tag{8}$$

where $p_n$ is the number of n-gram matches between the candidate sentence ($C$) and the reference sentence ($R$), minus the sum of positive difference of n-gram matches between candidate-source sentences ($C, S$) and candidate-reference sentences ($C, R$), divided by the number of n-grams in the candidate sentence ($C$):

$$p_n = \frac{\sum_{ng \in \{C \cap R\}} count_{C,R}(ng) - \sum_{ng \in \{C \cap S\}} max[0, count_{C,S}(ng) - count_{C,R}(ng)]}{\sum_{ng \in C} count_(ng)} \tag{9}$$

where $ng \in \{A \cap B\}$ are common n-grams in sentence $A$ and $B$, and $count_{A,B}(ng)$ is the minimum occurrence number of n-grams in $A$ and $B$. $n = 1, ..., N$ where $N$ is the maximum number of grams to be considered. $w_n$ is the weight of $p_n$. We set $N$ to 4 which is a common practice in the machine translation and grammatical error correction literature, and set all $w_n = \frac{1}{N}$. $BP$ is a brevity penalty which penalizes short candidate sentence (that may have a higher $p_n$ due to the small number of n-grams in the sentence).

$$BP = \begin{cases} 1 & c > r \\ e^{(1-\frac{r}{c})} & c \leq r \end{cases} \tag{10}$$

where $r$ is the length of the reference sentence, and $c$ is the length of the candidate sentence.

GLEU is expressed a percentage value between 0 and 100. The higher the GLEU, the closer the candidate sentence is to the reference sentence. If the candidate sentence completely matches the reference sentence, the GLEU is 100.

| Method | GLEU |
|---|---|
| RNN encoder-decoder | 57.44 |
| LanguageTools | 51.93 |
| SMT | 46.85 |

Table 3. The performance of different methods for sentence editing

| | Original Sentence | Our RNN Encoder-Decoder | LanguageTool | Phrase-based SMT |
|---|---|---|---|---|
| 1 | What I did wrong? | What did I do wrong? | What I did wrong? | What I did wrong? |
| 2 | pls help me.. | Please help me.. | Pls help me.. | Please help me.. |
| 3 | However, you show use CSS for this. | However, you should use CSS for this. | However, you show use CBS for this. | However, you use for this. |
| 4 | I'm thinking it has something to do with the json. | I'm thinking it has something to do with the JSON. | I'm thinking it has something to do with the son. | I'm thinking it has something to do with the JSON. |
| 5 | Inside the li tag we have many options to select. | Inside the `li` tag we have many options to select. | Inside the Ali tag we have many options to select. | Inside the tag we have many options to select. |
| 6 | Here selectedShape is either circle or polygon. | Here `selectedShape` is either circle or polygon. | Here selectedShape is either circle or polygon. | Here selectedShape is circle or polygon. |
| 7 | Edit: By the way, this is my first time using the community wiki; | **Edit:** By the way, this is my first time using the community wiki; | Edit: By the way, this is my first time using the community wiki; | Edit: By the way, this is my first time using the wiki; |
| 8 | It looks a s if the Large image is taking up all the space. | It looks as if the Large image is taking up all the space. | It looks an s if the Large image is taking up all the space. | It looks as if the **Large** is taking up all the space. |
| 9 | Below the code i use to validate a user login outside magento. | Below the code I use to validate a user login outside magento. | Below the code i use to validate a user logic outside magenta. | the code I use to validate user login outside magento2. |
| 10 | How to find hte library dependency? | How to find the library dependency? | How to find Rte library dependency? | How to find the library dependency_? |

Table 4. Examples of sentence edits by different methods

## 5.4 Evaluation Results

We report out evaluation results by answering the following two research questions.

*5.4.1 What is the quality of recommended sentence edits by our method? How much improvement can it achieve over the baseline methods?*

Table 3 presents the GLEU score of different methods for sentence editing tasks in the testing dataset of 377,298 sentences. Our RNN encoder-decoder model achieves the best overall result with the average GLEU score 57.44. The average GLEU of the SMT is only 46.85, and the average GLEU of the LanguageTool is 51.93. According to the literature of machine translation [38, 42] and grammar error correction [26, 41], the improvement in the GLEU score by our model represents a significant improvement over the two baseline methods.

GLEU score is a relative strict evaluation metric, i.e., any editing mistake may lead to large decay of the GLEU score. Consider the original sentence "how do i make a file handle from a file path specified on the command line?" in Table 2. There should be three edits: *how* to *How*, *i* to *I*, and *file handle* to *filehandle*, as seen in the reference sentence. Our RNN model suggests the first two edits, but misses the third edit. But the GLEU score is only 48.36. Similarly, for the other sentence "Would you recommend Java/J2EE, .Net/Erlang?", our RNN model suggests one edit, but miss the other edit. The GLEU score is only 27.08. However, compared with precision (100% in both examples) and recall (66.7% and 50% respectively), GLEU can better reflect editing quality with respect to editing effort required.

To qualitatively understand the strengths and weakness of different methods, we randomly sample about 600 sentences for manual inspection. Table 4 lists some representative categories of examples in which our method outperforms the two baseline methods. Due to the page limitation, other examples (both high-quality and low-quality) can be found online[12]. We can see that compared with the two baseline methods, our model can carry out relatively complex edits (e.g., the first example) and domain-specific word and site-specific formatting (e.g., the 4th, 5th, 6th and 7th examples). In such cases, the LanguageTool mostly preserves the original sentences because it does not have rules for them. Even worse, because many domain-specific words (e.g., *css*, *json*, *magento*) are out of its vocabulary, the LanguageTool may regarded them as typos of some general words and make erroneous edits, such as *CBS* for *css* (the 3rd example), *son* for *json* (the 4th example), *magenta* for *magento* (the 9th example).

The SMT can edit some domain-specific words (e.g., *json* to *JSON* in the 4th example). But it often preserves the original sentences that should be edited (e.g., the 1st and 7th examples), removes words that should not be removed (e.g., the 3rd, 5th and 6th examples), formats the sentence (e.g.,

[11] https://github.com/cnap/gec-ranking

[12] http://tagreorder.appspot.com/sentenceCorrection_examples.html

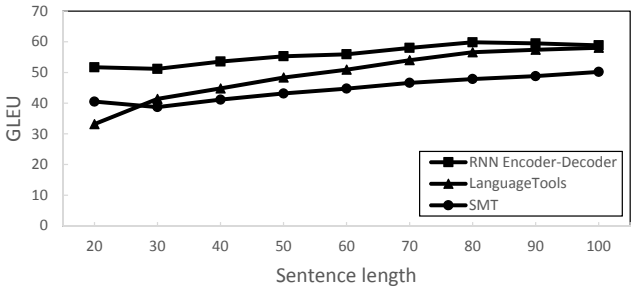


Fig. 9. The performance of different methods dealing with sentences of different length (the number of characters of the sentence)

the 8th example) that should not be formatted, or introduces some strange words (e.g., the 9th example). In general, the phrase-based SMT does not work very well for minor scale of changes involved in post edits. Therefore, it often has worse GLEU than the LanguageTool.

*5.4.2 In which cases does our model recommend low-quality sentence edits?*

By analyzing low-quality recommendations by our tool, we find four main cases in which our model does not perform well.

First, some sentences are edited to add more information which is beyond the context of a sentence, such as editing "I am currently working on a large project that heavily uses smart pointers with reference counting." to "I am currently working on a large project in C++ that heavily uses smart pointers with reference counting.". Our current model considers only the local context of a sentence. To support such complicated edits, the broader context of the sentence (i.e., previous and subsequent sentences) need to be considered in the future.

Second, sometimes the context captured by our model may not be long enough. For example, the original sentence "My db engine is MySQL I have two table 1." should be edited to "My DB engine is MySQL I have two table 1.". But our method recommends not only capitalizing "db" to "DB", but also changing "table" to "tables". However, the LanguageTool will not make such a mistake because there are no rules inside it to change singular form to plural form.

Third, different users may have different opinions regarding what should or should not be edited. For example, some users will edit the sentence "Would you recommend Java/J2EE, .Net / Erlang ?" to "Would you recommend Java/J2EE, .NET or Erlang?" by changing "/" to "or". However, many other users will not do that. Many revert-back edits we see when collecting original-edited sentences are the evidence of such different opinions. Different editing opinions often result in non-obvious editing patterns, which machine learning techniques cannot effectively encode.

Fourth, the sentence length is a crucial factor influencing the performance of the RNN model. As shown in Figure 9, with the increase of the sentence length, the GLEU of all three methods becomes higher. This does not simply mean that these methods work better on longer sentences than shorter sentence. Instead, this is because GLEU favors longer sentences as they would require more effort to read and edit. For example, although the tool misses one edit for both sentences in Table 2, the GLEU for the longer sentence is much higher than the shorter sentence. For all sentence lengths we experiment, our model performs the best. But the performance difference between our model and the baseline methods narrows as the sentence length increases. The fact that GLEU favors longer sentences and the performance difference narrows actually indicates that the performance of our RNN model decays as the sentence length increases. That is because the RNN model may "forget" some prior information when processing towards the end of a long sentence. As such, it cannot encode long-distance editing patterns very well which will lead to edit mistakes.

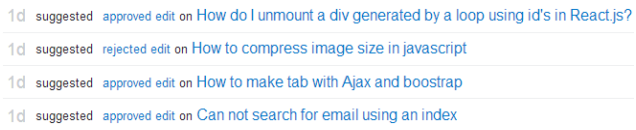


Fig. 10. Part of the results of post edits by our model

## 6 ASSISTING NOVICE POST EDITORS IN POST EDITING

Having established the confidence in the quality of sentence edits by our tool, we would like to further investigate whether the recommended sentence edits by our tool can assist novice post editors who have little experience in post editing and have little expertise in post content in successfully completing post editing tasks.

To that end, we conduct a small-scale field study, in which the first author who has 400+ reputation score in Stack Overflow acting as a novice post editor. We randomly select 50 posts from April 4 to April 6, 2017 which is of reasonable size, and the human effort to manually collect the posts and submit the post edits is manageable.. These 50 posts are not in our dataset of archival post edits. In stack Overflow, each question can have up to 5 tags to describe its topics. The 50 selected posts contain in total 123 tags (if the post is the answer, we take tags from its corresponding question) and 105 of these tags are unique. This indicates that the 50 selected posts cover a diverse set of programming topics. In fact, these posts contain many technical terms that are beyond the expertise of the first author.

In Stack Overflow, novice post editors have to submit their post edits for peer review. According to the Stack Overflow policy[13], to guarantee the quality of post edits, a post edit will be accepted only if at least three trusted contributors approve the edit otherwise it will be rejected. The first author uses our model to generate sentence editing recommendations for the selected 50 posts, based on which he edits the posts and submit the post edits to the community for approval. Our field study lasts for three days because each user in Stack Overflow can submit at most 5 post edits at the same time. We cannot submit more post edits until some submitted post edits are accepted or rejected.

Among the 50 selected posts, our model finds that 39 posts need at least one sentence edit and suggests the needed edits. For these 39 posts, there are on average 3.9 sentences edited and 5.6 edits per post (one sentence may receive several edits). Among the 39 submitted post edits, 36 (92.3%) are accepted and 3 (7.7%) are rejected. Records of some accepted and rejected edits[14] are shown in Figure 10. 3 rejected post edits contain 1, 2, 4 sentence edits respectively. For example, one post edit involves only adding a question mark to the title, and it got three reject votes. The other two rejected post edits actually got two approval votes but one reject vote. Although our tool recommends the correct sentence edit, the post edit is rejected because it is regarded as too minor which does not significantly improve the post quality[15]. In other words, for the 36 accepted post edits, at least trusted contributors believe that they contain sufficient edits that significantly improve the post quality, and thus approve them.

## 7 FEEDBACKS OF TRUSTED POST EDITORS

Finally, we conduct a survey of trusted post editors to understand three questions: 1) the trusted post editors' attention to small sentence edits. This will help us understand the need for an edit

[13]https://meta.stackexchange.com/questions/76284
[14]We cannot release the full results now as they will expose our identity which violates the double-blind polity.
[15]Predicting whether a post edit contains significant enough sentence edits is out of scope of this work.

assistance tool. 2) The trusted post editors' opinions on the potential usefulness of our tool. 3) The trusted post editors' suggestions for our tool, which may inspire future enhancement of our tool.

We design a survey with three 5-points likert scale questions and 1 free-text question. The likert scale questions are: 1) How much do yo care about spelling, grammar, formatting edits? (1 being very little and 5 being very much); 2) What percentage of your edits are spelling, grammar, formatting edits? (1 being very low and 5 being very high); 3) How much could our tool help with such edits? (1 being very little and 5 being very much). The first and third questions are accompanied with the examples in Table 1 and Table 4 for illustration purpose. The free-text question asks for "any suggestions or opinions for our tool and this research".

To find survey participants, we sort all Stack Overflow users by the number of post edits they have made in descending order. We read the profiles of the top 2000 ranked users and find the email contacts for 410 users. Each of these 410 users has at least 800 post edits to their own or others' posts. We send these 410 users an email introducing the background and goal of our work and providing access to the survey. Among these 410 users, we collect 61 valid survey replies[16].

Figure 11 summarizes the results of the three likert scale questions. 55 of 61 survey respondents care much or very much about spelling, grammar, formatting edits. 30 respondents report that high or very high percentage of their edits is spelling, grammar, formatting edits. These results confirm the trusted post editors' attention to small sentence edits. 34 respondents consider that our tool would be helpful or very helpful for assisting small sentence edits.

34 of 61 respondents provides their opinions or suggestions for the free-text question. Those considering our tool helpful comment that "having a natural language and correct grammar is important as all accepted answers will be archived for reference in future", "SO needs this tool and I hope to see it in action soon. I believe the resulting tool might be useful outside the context of SO websites", and "Very good idea, that would deserve to be integrated into StackOverflow as an assistance tool". Some mention "make your tool free software (open source)".

However, even strong supporters have concerns like "How will it get integrated with the SO site?". Similar concerns are mentioned by those holding neural opinion "I believe a tool like that would only be pretty useful if it somehow did what it did without at all getting in the way", and those considering our tool not helpful "dubious if you can create an interface that is sometimes useful and doesn't get in the way when it isn't". We will elaborate our future plan to integrate our tool in Stack Overflow in Section 8.1.

Some respondents consider our tool not helpful because they do not like spell checking tools at all "Not interested. Same reason I abhor spell and grammar checkers. Generally way too many false positives". Others prefer to use existing tools "I use a browser plugin (Pentadactyl) to do the edits in an external editor (Vim), ... Any browser-based tool therefore would be of little value to me.". It would be interesting to see if these respondents would appreciate the uniqueness and quality of our tool if they had actually use the tool.

Finally, both consider-helpful and consider-not-helpful respondents suggest that we should consider assisting code formatting "Another useful thing would be to detect when some code isn't indented enough", "Most of my edits are fixing indentation and formatting to make the code more readable", "I don't care about English formatting. Code formatting is very important for me". We believe they point out an interesting future direction.

## 8 DISCUSSION

### 8.1 Impact on Social Process and Quality Control

Our edit assistance tool can be integrated as a plugin of post editor or viewer to assist post editing or viewing. In fact, one survey replier suggests that "Maybe it would automatically highlight things in text areas on SO, or maybe it'd even optionally highlight things in posts even before I entered the edit interface, to prompt me that there are things in that post I could fix". In this way, our tool works in a similar way to existing spell checkers. It focuses on

[16] Available at http://tagreorder.appspot.com/surveyResults.html

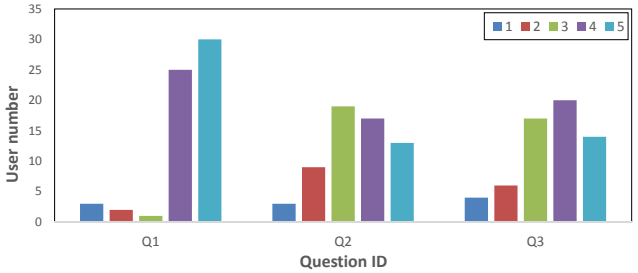


Fig. 11. The survey results

handling information without incorporating notions of role, process and social interaction [15]. However, our tool may still indirectly impact social process and quality control.

First, if the suggested highlighting feature is supported, it may impact to which posts human editors allocate their attention. Furthermore, the editing knowledge of our tool comes from archival post edits that human make. However, editing new terms emerging as technologies evolve still needs human editors. Our tool may also impact to what information (the editing needs that the tool can assist or the emerging editing needs) human editors allocate their attention. In any cases, our edit assistance tool will never replace human editors, but assist the allocation of their attention.

Second, post editors can use our tool to improve the post quality, but conflicts may still occur when collaborative editing changes the post meaning[17]. This may discourage other users' subsequent contributions. However, the risk of adopting our tool in discouraging users' contributions would be low. Our assistance tool corrects minor textual issues which would rarely change the post meaning and the recommended edits is generally of high quality. Furthermore, our tool only recommends sentence edits, while the editors decide whether to adopt the recommended edits or not. The double checking by the editors mitigates the potential effects of altering the meaning of the original posts. And Li et al [21] show that collaborative edits by human editors decrease the subsequent contributions marginally.

Third, novice post editors can use our tool to learn to correct minor textual issues in others' posts. Such small editing tasks can provide a mechanism of legitimate peripheral participation [19] for novice users. However, the increasing post edits by novice editors may impact the edit review process in Stack Overflow. On the one hand, as our field study shows, novice post editors can edit many posts in a short time with the tool's assistance. This will increase the approval working load of trusted contributors. On the other hand, novice editors may accept some erroneous edit recommendations made by the tool due to the lack of experience and knowledge. It may result in some low-quality post edits. The existing Stack Overflow policy prevents such a situation from happening because one novice contributor can submit at most 5 post edits for approval at the same time and they cannot continue until some submitted edits have been approved. Suitable policies and machine learning methods like Wikipedia's OREs [1] may be further explored to avoid such unintended consequences in the future.

### 8.2 Generalizability of Our Approach and Findings

In this work, we study only Stack Overflow post edits. However, our approach is not tied to any collaborative editing or quality control process in online communities. The input to our approach is essentially just a parallel corpus of original and edited text (see Figure 5). Therefore, we would expect that the work flow of our approach can be applied to other online communities, such as

[17]https://meta.stackexchange.com/questions/28005

other Stack Exchange sites, Wikipedia [3, 17], Quora[18] where sufficient content editing history is archived, to develop edit assistance tool.

Adopting our approach to other online communities needs to consider the content and editing characteristics of those communities. For other Q&A sites about computer programming like CodeProject[19], the model trained by Stack Overflow data could be directly adopted. It would also be straightforward to apply our approach to other Stack Exchange sites as all Stack Exchange sites follow the same general community practices. However, for non-computing related Q&A sites like mathematics, physics or chemistry, domain-specific regular expressions need to be developed to normalize domain-specific rare words. For online communities like Wikipedia or Quora whose content and editing practices are substantially different from Stack Overflow, a formative study of the content and editing practices would be necessary to determine how to collect training data and choose an appropriate deep learning method.

The other critical issue that affects the applicability of our approach is the availability of sufficient archival editing data for training the deep learning model. By sufficient, the editing data is not necessarily as large scale as Stack Overflow data in this study. For online communities like Stack Overflow, Wikipedia, or Quora where the content and editing patterns are very diverse, a large scale editing data is required and also available. For other sites that have much less editing data, it may still be possible to train a reliable model because the content would be more focused and the editing patterns would be less diverse. Future comparative studies are required to confirm the extent to which the training data size affects our approach.

Another issue is whether the findings of our approach's effectiveness and usefulness through the evaluation of archival post edits, a small field study and the survey of trusted post editors will still be valid in a large-scale, live deployment of the tool in Stack Overflow. We are confident in the quality of recommended sentenced edits, but how to integrate our tool in Stack Overflow, whether the recommended edits will be accepted in practice and how they may impact post owners and editors' behavior and the edit-review process require many further studies.

## 9 CONCLUSION AND FUTURE WORK

In this paper, we investigate the need for and the feasibility of assisting small sentence editing in Stack Overflow. Our empirical observation of post edits in Stack Overflow and the survey of trusted contributors confirms the need for an edit assistance tool and the potential benefits for the community. A deep learning based approach has been developed to learn to apply sentence editing patterns from large-scale post edits by the community. Our evaluation through large-scale archival post edits demonstrates the quality of recommended sentence edits by our tool. Our field study with a novice post editor demonstrates the tool's ability to assist novice editor to edit posts with a wide range of topics. Our survey of trusted contributors shows that trusted contributors appreciate the tool's potentials in assisting post editing, but they also raise the concerns about how to integrate the tool in Stack Overflow and how to extend the tool for complex edits such as adding/removing sentences or code formatting.

Although our tool, as a single-user application, can potentially assist post owners and editors in improving content quality or assist novice editors in learning community-adopted editing patterns, deploying our tool in Stack Overflow may have complicated impacts on social process and collaboration, which deserve further studies in the future. We will also extend our approach to other online communities to understand the generalizability of our approach, especially its reliance on big data. Extending deep learning approach to code formatting could also benefit the

[18]https://www.quora.com/What-are-some-guidelines-and-policies-for-editing-a-question-on-Quora
[19]https://www.codeproject.com/script/Answers/List.aspx

Stack Overflow community, as existing code formatting tools are all based on manually developed rules for a particular language. Adding/removing whole sentences is still an open challenge for the RNN-based model, but it could be feasible to train a classifier to decide what kinds of information (e.g., code, screenshots, web resources) need to be added or removed in a post.

## 10 ACKNOWLEDGEMENT

We thank the anonymous reviewers for their valuable reviews which help reshape this paper. We also thank Shamil Chollampatt in NUS for the evaluation metric suggestions of the sentence edits. The GPU used in this work to speed up the experiment is supported by NVIDIA AI Technology Center, Singapore.

## REFERENCES

[1] 2017. The Objective Revision Evaluation Service. https://ores.wikimedia.org/. (2017).
[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).
[3] 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.
[4] Guntis Barzdins, Steve Renals, and Didzis Gosko. 2016. Character-Level Neural Translation for Multilingual Media Monitoring in the SUMMA Project. *arXiv preprint arXiv:1604.01221* (2016).
[5] 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.
[6] Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18, 4 (1992), 467–479.
[7] 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.
[8] 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.
[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[10] Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural Network Translation Models for Grammatical Error Correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*. 2768–2774.
[11] Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, 54–62.
[12] Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*. Association for Computational Linguistics, 242–249.
[13] Mariano Felice. 2016. *Artificial error generation for translation-based grammatical error correction*. Technical Report. University of Cambridge, Computer Laboratory.
[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.
[15] Jonathan Grudin. 1994. Groupware and social dynamics: Eight challenges for developers. *Commun. ACM* 37, 1 (1994), 92–105.
[16] Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. *arXiv preprint arXiv:1605.06353* (2016).
[17] Aniket Kittur and Robert E Kraut. 2008. Harnessing the wisdom of crowds in wikipedia: quality through coordination. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. ACM, 37–46.
[18] Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.
[19] Jean Lave and Etienne Wenger. 1999. Legitimate peripheral participation. *Learners, learning and assessment, London: The Open University* (1999), 83–89.
[20] Guo Li, Tun Lu, Xianghua Ding, and Ning Gu. 2016. Predicting Collaborative Edits of Questions and Answers in Online Q&A Sites. *çűšéŽŻçűšèůřæŁĂèąŞåŋÿâĹŁ* 17, 6 (2016), 1187–1194.

[21] 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.

[22] Zhuoran Liu and Yang Liu. 2016. Exploiting Unlabeled Data for Neural Grammatical Error Detection. *arXiv preprint arXiv:1611.08987* (2016).

[23] Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. 11–19.

[24] 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.

[25] Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. *arXiv preprint arXiv:1507.01193* (2015).

[26] Tomoya Mizumoto and Yuji Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *Proceedings of NAACL-HLT*. 1133–1138.

[27] Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Vol. 2. 588–593.

[28] Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU Without Tuning. *arXiv preprint arXiv:1605.02592* (2016).

[29] Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction. *arXiv preprint arXiv:1610.02124* (2016).

[30] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction.. In *CoNLL Shared Task*. 1–14.

[31] Daniel Ortiz-Martınez, Ismael Garcıa-Varea, and Francisco Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. *Tenth Machine Translation Summit. AAMT, Phuket, Thailand, September* (2005).

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[33] Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics* 4 (2016), 169–182.

[34] Allen Schmaltz, Yoon Kim, Alexander M Rush, and Stuart M Shieber. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. *arXiv preprint arXiv:1604.04677* (2016).

[35] Andrew W Vargo and Shigeo Matsubara. 2016. Editing Unfit Questions in Q&A. In *Advanced Applied Informatics (IIAI-AAI), 2016 5th IIAI International Congress on*. IEEE, 107–112.

[36] Fernanda B Viegas, Martin Wattenberg, and Jonathan Feinberg. 2009. Participatory visualization with wordle. *IEEE transactions on visualization and computer graphics* 15, 6 (2009).

[37] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* (2016).

[39] 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).

[40] Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of NAACL-HLT*. 380–386.

[41] Zheng Yuan, Ted Briscoe, and Mariano Felice. 2016. Candidate re-ranking for SMT-based grammatical error correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. 256–266.

[42] Ying Zhang, Stephan Vogel, and Alex Waibel. 2004. Interpreting bleu/nist scores: How much improvement do we need to have a better system?. In *LREC*.

Received June 2017; Revised August 2017; Accepted November 2017