Learning to Describe Editing Activities in Collaborative Environments: A Case Study on GitHub and Wikipedia

Edison Marrese-Taylor[1], Pablo Loyola[2], Jorge A. Balazs[1] and Yutaka Matsuo[1]
Graduate School of Engineering, The University of Tokyo, Japan[1]
{emarrese, jorge, matsuo}@weblab.t.u-tokyo.ac.jp
IBM Research, Tokyo, Japan[2]
e57095@jp.ibm.com

Abstract

We propose to study the automatic generation of descriptions from content editing activities in collaborative environments. We define such task as identifying the changes associated to two consecutive versions of a document, and then producing a message in natural language that explains it, which should provide a compact description of the change while retaining its key informative elements. Our model is based on a sequence to sequence architecture that receives as input the representation of the change, and outputs a message. We propose a framework to conceptualize the problem and two instances for GitHub activity and Wikipedia contributions, two of the most important collaborative systems on the Web. Our results indicate that the proposed approach is able to generate feasible descriptions, which are on average aligned with the semantic purpose of the editing activities.

1 Introduction

One of the positive outcomes of the current pervasiveness of the Web has been the boost of collaboration across several domains. Examples of this are platforms such as Wikipedia and GitHub, where self-organized and voluntary groups of individuals gather based on the common goal of crafting documents and programs (Crowston et al., 2007).

The outcome of these collaborative activities is usually the result of a series of incremental modifications over time. For example, in the case of GitHub, incremental modifications are usually functional changes, which allow to incorporate new features or fix reported bugs. In the case of wiki-based platforms, contributors modify the content of a given article in order to reflect an update on the matter the article is dealing with. The transparency and openness of change management provides complete awareness of the state of the document being crafted, at any point in time (Dabbish et al., 2012).

As collaboration is carried out in a decentralized fashion, the coordination between contributors plays a key role (Von Krogh et al., 2003). While there exist direct ways of communication, such as bug trackers and discussion forums, we are interested in studying the indirect ways in which contributors interact and coordinate.

One of these elements are the short messages that the contributors provide at the time of submitting the change. This short message usually provides a description of the change and serves as a way of broadcasting it to the rest of the community, ideally clarifying the purpose and other technical aspects, and supporting the reviewing process (Guzman et al., 2014).

Therefore, our goal is to use this set of change-message pairs to develop a model able to explain collaborative activities by automatically generating a short passage in natural language. In that sense, we visualize this task as being in-between summarization and translation given the challenges it presents, namely, (i) the length asymmetry between changes and their messages, and (ii) the fact that documents can be written in modalities different from natural language (e.g. source code, art). Our intention is to learn the most salient elements that characterize the change, and then decode them into a description in natural language. As we will show, this duality has implications on the choices of metrics used for evaluation.

Moreover, rather than describing the content that was changed, we are generating a description of the *action* taken over it, therefore, there is an inherent temporal dimension associated to the generation. Additionally, the action can be seen as the result of an optimization problem: given the state of the file, the agent needs to find the most efficient change that allows him to satisfy the requirement. In other words, the change performed on the file is a function dependent on the current functional state of the file and the given requirement: the change performed was such, only because of the given state of the file. If such state was different, then the change would have been different too.

The usage of models based on deep neural networks in natural language processing has been successful in large part because they learn and use their own continuous numeric representational systems for word and sentences. In particular, distributed representations (Hinton, 1984) applied to words (Mikolov et al., 2013) have meant major breakthroughs allowing networks to parse and represent sentences and phrases using an effective compositional vector grammar. Recurrent neural networks now provide state-of-the-art performance in tasks such as machine translation, sentence-level sentiment analysis, text generation and automatic image captioning.

Moreover, the introduction of the encoder-decoder (Cho et al., 2014) or sequence-to-sequence (Sutskever et al., 2014) architectures presented a successful framework based on neural networks that aims to map highly structured input to highly structured output. Additional improvements on the encoder-decoder architecture came with the addition of attentional components (Bahdanau et al., 2015; Luong et al., 2015), which allowed the decoder to focus on specific information provided by the encoder at a time.

Therefore, to tackle the introduced problem we use a representation learning approach. More concretely,our approach takes inspiration in recurrent neural models, widely used in sequence-to-sequence learning (Bahdanau et al., 2015), but we introduce specific extensions to account for the structural differences in our case. While the Web offers several types of collaborative environments, in this work we focus on GitHub activity and Wikipedia contributions, based on their popularity and data availability. We perform an empirical study based on collected editing activity, and show that the introduced models are able to learn representations from the changes and produce sound descriptions in most cases.

2 Related Work

The analysis of editing activities has been tightly associated with quality assessment tasks. For example, in software engineering, version changes are the basis of regression testing and defect prediction (McIntosh and Kamei, 2017).

In the case of Wikipedia, since one of its core principles is being open for anyone to maintain it, Wikipedia cannot fully ensure the reliability of its articles, and thus sometimes had suffered criticism for containing low-quality information. It is therefore essential to assess the quality of Wikipedia articles automatically. In this context, for example, Su and Liu (2015) approach the problem by using a psycho-lexical resource. On the other hand, Kiesel et al. (2017) aim at automatically detecting vandalism utilizing change information as a primary input. Gandon et al. (2016) also validate the importance of the editing history of Wikipedia pages as a source of information, presenting a new extraction technique which produces a linked data representation for it.

More recently, Yang et al. (2017) proposed an approach for identifying semantic edit intentions from revisions in Wikipedia. Also, Sarkar et al. (2019) and Marrese-Taylor et al. (2019) have focused on the quality assessment issue and proposed approaches that directly produce an edit-level quality label for a given Wikipedia edit. While the former is concerned only with edit-level quality classification of edits, the latter also incorporates a generative part similar to ours but only as an auxiliary task.

Our work is also related to summarization on Wikipedia. Recent work includes Chisholm et al. (2017), where the authors proposed an autoencoder-based model to generate short biographies, and Zhang et al. (2017), where authors present a method to summarize the discussion surrounding a change in the content, along with a visualization tool to ease comprehension of its evolution.

When it comes to GitHub, we find several papers that perform analyses over the platform, including the work of Batista et al. (2017), who study the correlation among features that measure the strength of social coding collaboration and Nielek et al. (2016), who try to predict which developer will join which project.

In terms of specifically working on code change descriptions, we see that the paradigm is based on the distributional similarities that emerge between natural and programming languages (Hindle et al., 2012). Indeed, both are ways of communication based on sets of defined vocabularies, and their composition is based on structured and sequential instructions. Concretely, Cortes et al. (2014) and Linares et al. (2015) proposed methods based on a set of rules that consider the type and impact of the changes, and Buse and Weimer (2010) combine summarization with symbolic execution.

Moreover, mapping source code to natural language has received special attention in recent years, mainly in the form of summarization. Examples if this are the work of Allamanis et al. (2016) who use a convolutional neural network approach, and Iyer et al. (2016) who used an recurrent neural network architecture capable of learning to summarize Stack Overflow snippets.

In terms of code change description generation, the use of a representation learning paradigm has been proposed Jiang et al. (2017; 2017) and by Loyola et al. (2017; 2018). The authors train an encoder-decoder architecture on a set of commit-message pairs extracted from GitHub open source projects to generate change descriptions. We took that work as a starting point and proposed an extended architecture that considers intra-change comments with an ad-hoc attention mechanism, with the additional feature of generalizing to other data sources such as Wikipedia changes. More recent variations of this include augmenting the model with a pointer network (Liu et al., 2019a), or with abstract syntax trees (Liu et al., 2019b). In contrast, Liu et al. (2018) focused on efficiency and proposed a method that relies on nearest neighbors instead the encoder-decoder.

Finally, our work is also related to Yin et al. (2019), who proposed a general framework for learning edit representations based on a self-supervised approach similar to an auto-encoding task.

3 Proposed Approach

Generative tasks, such as summarization and translation, try to map between source and target sequences ignoring time dependencies across examples. Our main motivation for this work is to explore a task where we can generate a natural language description of a *transition* between states, i.e., adding temporal dimension into the generation by learning to represent the difference between consecutive versions of the changed document.

Web-based collaborative platforms represent a convenient source of indirectly supervised data, as each contributed change is usually required to be submitted along with a short description of its purpose and detail. For this work, we focus on source code changes on GitHub and Wikipedia contributions, based on their availability.

From a broad perspective, a *change* can be seen as the consequence of a requirement, which can be *external*: e.g., the need for a new functionality on a GitHub project, or the need reflect a recent event on someone's biographical article on Wikipedia; or *internal*: e.g., a reported bug or a functionality mismatch on a GitHub project, or the need to revert a vandalism attack on a Wikipedia article. That requirement is internalized by a contributor that identifies which portion of the document should be modified in order to satisfy the requirement. As all proposed changes are expected to be reviewed by a peer, the contributor appends a short description explaining the purpose. Such dual configuration (changes, descriptions) represents our main data source for training.

For both modalities, **GitHub** and **Wikipedia**, we assume the existence of T versions of a given project or article $\{v_1, \ldots, v_T\}$. Given a pair of consecutive versions (v_{t-1}, v_t), we define the tuple (C_t, N_t), where $C_t = \Delta_{t-1}^{t}(v)$ is a representation of the content changes associated to v in time t, and N_t is a representation of its corresponding natural language (NL) description. Let $\mathcal{C}$ be the set of content changes and $\mathcal{N}$ be the set of all descriptions in NL. We consider a training corpus with T content snippets and summary pairs (C_t, N_t), $1 \leq t \leq T$, $C_t \in \mathcal{C}$, $N_t \in \mathcal{N}$. Then, for a given content snippet $C_k \in \mathcal{C}$, the goal of our model is to produce the most likely NL description $N^{\star}$. The nature of the content snippet $C_k \in \mathcal{C}$ depends of the modality considered.

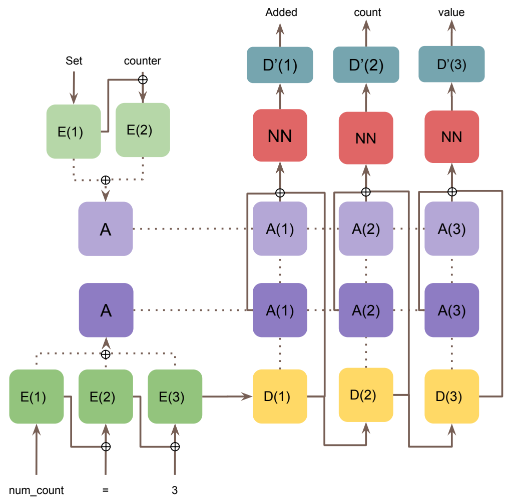

Figure 1: Model architecture with two encoders for the GitHub modality.

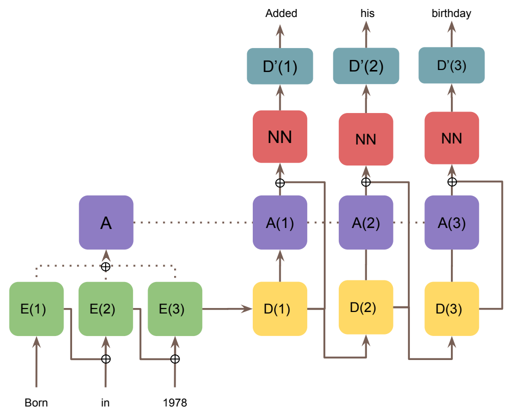

Figure 2: Model architecture based on sequence-to-sequence for the Wikipedia modality.

For both modalities, similarly to Iyer et al. and Loyola et al. (2016; 2017), we use an attention-augmented encoder-decoder architecture. On each case, we assume the existence of an ad-hoc encoder that allows us to obtain a representation for the change we intend to study. The only assumption about our encoders is that their inputs have to be represented as a sequence of tokens, $c_i \in C_t$. With this, our encoders rely on embeddings and bidirectional LSTMs. Let $X_t = x_1, \ldots x_n$ be the embedded input content sequence C_t, using embedding matrix E.

$$\vec{h}_i = LSTM(x_i, \vec{h}_{i-1}) \quad (1)$$
$$\overleftarrow{h}_i = LSTM(x_i, \overleftarrow{h}_{i+1}) \quad (2)$$
$$h_i = [\vec{h}_i; \overleftarrow{h}_i] \quad (3)$$

We add special beginning-of-sentence *BOS* and end-of-sentence *EOS* tokens to our output NL sequences, and set the decoder to be an LSTM that reads the representation given by the encoder, generating NL words one at a time based on its current hidden state and guided by a global attention model (Luong et al., 2015). We model the probability of a description as a product of the conditional next-word probabilities. We use an embedding matrix D to encode each NL token $n_i \in N_t$ into a sequence of vectors $Y_t = y_1, \ldots y_m$ and set

$$s_i = LSTM(y_{i-1}, s_{i-1}) \quad (4)$$
$$p(n_i|n_1, \ldots, n_{i-1}) \propto W \tanh(W_1 s_i + W_2 a_i) \quad (5)$$

where $\propto$ denotes a softmax operation, s_i represents the decoder hidden state and a_i is the contribution from the attention model on the input. W, W_1 and W_2 are trainable combination matrices. The decoder repeats the recurrence until a fixed number of words or a special *END* token is generated. The attention contribution a_i is defined as $a_i = \sum_{j=1}^{k} \alpha_{i,j} \cdot h_j$, where $h_j \in H$ is a hidden state associated to the input and $\alpha_{i,j}$ is:

$$t_i = \sum_{j=1}^{k} \alpha_{i,j} \cdot h_j \quad (6)$$
$$\alpha_{i,j} = \frac{\exp\left(h_i^{\top} s_i\right)}{\sum_{h_j \in H} \exp\left(h_j^{\top} s_i\right)} \quad (7)$$

In this way, the decoder is trained as a conditioned language model over the NL vocabulary and on each generation step we let it have full access to the representation of the input as provided by the encoder using the attentional component.

Wikipedia: We set $C_k = x_1, ..., x_L$, as a sequence of L text tokens associated with a change. To encode this sequence we use a bidirectional LSTM.

GitHub: We build C_k based on both code and documentation changes, as extracted from $\Delta^t_{t-1}(v)$.

We define the change in source code C_k as having two components: a sequence of source code tokens $SC_k = x_1, ..., x_{L_{SC}}$, and a sequence of documentation tokens $SD_k = z_1, ..., z_{L_{SD}}$. To obtain a vector representation for $\Delta_{t-1}^{t}(v)$, as we model it as two different sequences, we use two bidirectional LSTMs as encoders, one for the source code sequence and one for the documentation sequence. We aggregate each representation using mean pooling and concatenate the resulting vectors. The resulting vector is used to initialize the decoder hidden state.

During training, for both modalities, the decoder iterates until the end of the sentence is reached. For generation, we approximate $N^{\star}$ by performing a beam search on the space of all possible summaries using the model output, with a beam size of 10 and a maximum summary length of equal to the maximum length of the input. For inference, we let the decoder run for this number of steps or until the *EOS* token is generated.

4 Empirical Study

4.1 Wikipedia

We collected historical data dumps from Wikipedia, choosing some of the most edited articles in English and German, in a way analog to the language choice in GitHub. For English, we worked with the articles for *United States* and *World War II*, while for German we chose *Deutschland* (Germany) and *Zweiter Weltkrieg* (World War II). To our eyes, one of the critical differences between our studied modalities is the amount of control users have over the editing activity, which is practically non-existent in the case of Wikipedia. To study this, we also collected the editing history of *Donald Trump*'s article, which exhibited a very dynamic and polarizing editing activity record.

Wikipedia dumps contain every version of a given page in *wikitext*, the official markup-like language, along with metadata for every edit. To obtain the content associated to each $\Delta_{t-1}^{t}(v)$, we sorted the extracted edits chronologically and computed the *diff* of each pair of consecutive versions using the Unix diff tool. Due to the line-based approach of the Unix *diff* tool, small changes in *wikitext* led to big chunks of differences in the resulting *diff* file. To alleviate this problem, we extracted the unique set of sentences that was either added or removed, which gave us a much fine-grained characterization of the edits. For English sentence splitting we used the automatic approach by Kiss et al. (2006), and Somajo (Proisl and Uhrig, 2016), for German.

We found that articles related to controversial topics —such as Donald Trump— exhibited a high proportion of reverting edits, as well as extreme vandalization cases. Since these edits provide no additional information to our model, we filtered them out.

4.2 GitHub

We rely on the concept of code commit, the standard contribution procedure implemented in modern subversion systems (Gousios et al., 2014), which provides both the actual change and a short explanatory paragraph. To model both as a sequence of source code tokens $SC_k = x_1, ..., x_{L_{SC}}$, and a sequence of documentation tokens $SD_k = z_1, ..., z_{L_{SD}}$ we use *diff* files associated to each commit for a given project in GitHub. These *diff* files encode per-line differences between two files or sets of files in a standard format, allowing us to recover source code changes at the line level.

We obtain all the *diff* files for a given project using the GitHub API. However, given the flat structure of the diff file, source code in contiguous lines might not necessarily correspond to originally neighboring code lines. Moreover, they might come from different files in the project. To deal with this issue, we followed Loyola et al. (2017) and only considered the *diff* files of those commits that modify a single file in the project.

To obtain the messages associated to each introduced change, we use the API to download the metadata associated to each commit, which allows us to recover information such as the author and message of each commit.

For this paper, we chose projects for Python and Javascript, as they are among the most widely adopted programming languages. We selected two of the historically most popular projects for each language on GitHub as data sources. For Python, we worked with *Theano* and *youtube-dl*, whereas for Javascript we worked with *angular* and *react*. We parsed the *diff* files using a lexer (Brandl, 2016) to tokenize their contents in a per-line fashion.

Modality	Dataset	Max. Length	Our Model		MOSES
			METEOR	BLEU	BLEU
GitHub	Theano	100	0.319	27.3	5.9
		300	0.220	27.4	5.5
	youtube-dl	100	0.132	18.3	17.6
		300	0.325	12.7	13.0
	angular	100	0.254	21.6	12.7
		300	0.412	20.2	9.7
	react	100	0.330	27.9	10.5
		300	0.263	22.6	7.3
Wikipedia	World War II	100	0.399	14.3	11.8
		300	0.244	14.5	5.2
	Zweiter Weltkrieg	100	0.330	17.5	16.3
		300	0.312	12.1	9.8
	United States	100	0.241	12.6	11.3
		300	0.325	12.8	9.0
	Deutschland	100	0.352	14.2	14.8
		300	0.352	13.9	10.4
	Donald Trump	100	0.610	14.7	10.5
		300	0.581	12.5	7.8

Table 1: Summary of our results on both modalities.

Modality	Max. Length	Mean Ours	Mean MOSES
GitHub	100	23.8	11.7
	300	20.7	8.9
Wikipedia	100	14.7	12.9
	300	13.2	8.4

Table 2: Summary, in terms of BLEU scores, of the impact of increasing the maximum sequence length across modalities.

The extracted commit end edit messages were processed using the Penn Treebank tokenizer (Marcus et al., 1993), which nicely deals with punctuation and other text marks typical of natural language. During experimentation, we found that some excessively repeating patterns on the NL descriptions, such as the phrase *merge pull request*, were misguiding for the learning process so we deleted them from the data, keeping the rest of the content of each sequence, if any. Sequences that solely contained these sequences were discarded.

To evaluate the quality of our generated descriptions we use METEOR (Lavie and Agarwal, 2007) and sentence level BLEU-4 (Papineni et al., 2002). These metrics, popular from automatic machine translation evaluation, are scores calculated for individual translated segments by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality. We compute them on our validation set after every epoch and save the intermediate model that maximizes each.

Following previous work on mapping source code to natural language (Loyola et al., 2017; Iyer et al., 2016), we used MOSES (Koehn et al., 2007) as a baseline, which although is designed as a phrase-based machine translation system, was previously used by Iyer et al. (2016) to generate text from source code. Concretely, we treated the tokenized input (only the source code for the case of GitHub) as the source language and the NL description as the target. We trained a 3-gram language model using KenLM (Heafield et al., 2013) and used mGiza to obtain alignments. For validation, we use minimum error rate training (Bertoldi et al., 2009; Och, 2003) in our validation set. To evaluate model capabilities, we generated two versions of each dataset for a maximum input/output sequence length of 100 and 300 tokens.

Data		Reference	Generated
GitHub	Theano	better test error UNK	better error message
		allow to disable the gpu when UNK and UNK	disable the gpu back-end .
		add test case .	added test message for UNK
	youtube-dl	[cbc] skip geo-restricted test case	[generic] add test
		[extractor/generic] add support for onionstudios embeds (closes # NUMBER)	[extractor/generic] handle UNK embeds (closes # NUMBER)
	angular	refactor (UNK) : remove UNK facade (# NUMBER)	refactor (changelog) : add UNK (# NUMBER)
		fix (core) : export dev mode api in UNK closes # NUMBER	fix (UNK) : add UNK UNK closes # NUMBER
	react	clarify tutorial UNK fixes # NUMBER .	clarify tutorial
		add shirtstarter to examples of UNK UNK .	update shirtstarter UNK
Wikipedia	D. Trump	/* Foreign policy */ wiki link	/* Foreign policy */ cite cleanup
		UNK not graduate from Fordham	He did not graduate from Fordham University
	U.S.	/* Economy */ Updated unemployment rate	/* Economy */ Its the US
		/* Economy */ update CPI	/* Economy */ update inflation data
	Deutschland	Änderungen von Benutzer : UNK rückgängig gemacht und letzte Version von Benutzer : UNK wiederhergestellt	Änderungen von Benutzer : UNK rückgängig gemacht und letzte Version von Benutzer : Aka wiederhergestellt
		/ * Von der Bonner zur Berliner Republik (1990 –Gegenwart) * / kor .	/ * Von der Bonner zur Berliner Republik (1990 –Gegenwart) * /

Table 3: Generated v/s original NL descriptions.

5 Results and Discussion

We summarize our results in terms of both METEOR and BLEU metrics on Table 1. Although we think these metrics may not be completely compatible with our task, since it is not exactly translation, results show that they indeed provide a notion of the degree of alignment between the modalities we are mapping. To gain insight into this we analyzed the cross-run correlation between each metric and the validation cross-entropy loss. We found that METEOR is generally more negatively correlated with the loss. Given that this metric uses language-specific resources, we think it may be over-estimating the quality of the generated passages, as in our case they are not regular English phrases. Based on these results, we relied on BLEU to choose the best model each time.

As shown in Table 2, our approach consistently outperforms the baseline. This is even clearer when increasing the maximum length from 100 to 300, which always considerably hinders the baseline's performance but has a comparatively smaller effect on our model. For the particular case of *Theano*, where the increment in length size affected BLEU positively but METEOR negatively for our model, we found that the sizes of both the source vocabulary and the number of training instances increased more compared to other cases —3% and 28% respectively— which could explain the abnormal behavior.

In the case of *youtube-dl*, where MOSES performed better than our approach, we found that the change in maximum length produced a considerable imbalance between the mean lengths of the source and target sequences. Further work is needed to devise a more effective learning strategy in such cases.

In terms of the modalities studied, we see that for GitHub, while the gains of the proposed model against MOSES for both Javascript and Python projects are similar for both sequence length settings –average of 13% and 12% for Javascript, and 11% and 10 % for Python– Python presents higher variance, which is caused by the disparity in performance between *Theano* and *youtube-dl*. In the case of Wikipedia, the model performs consistentl well across articles, always outperforming the baseline.

A more qualitative result is presented in Table 3, where we compare the generated descriptions against the ground truth messages from the test set. In general, we see that the model is able to consistently generate semantically sound descriptions, which are also semantically well correlated to the reference messages. Our results also suggest the emergence of rephrasing capabilities, as the models tend to choose general terms over more specific ones, while also dropping parts of the messages that may seem irrelevant.

An important note is that the model suffers from hallucination, a common problem in sequence-to-sequence models. Specifically, in the case of GitHub, we see that for those projects whose NL messages exhibit a fixed pattern in their structure, such as in the case of *youtube-dl* where users add a header denoting the file that was edited in the commit, the model tends to more frequently hallucinate the content of the message. In this case, as the content of the "header" section may be too specific for the model to leverage on, we believe this restrains the generation capabilities of the decoder, making it more prone to memorization and therefore less able to correctly generalize.

In the case of Wikipedia, we observed that in most of the cases the model was able to correctly generate the portion of the edit messages that lies between the "/*" symbols, which again can be regarded as a message "header". Compared to the case of GitHub, the nature of the header seems to be different, however. We manually checked the messages and discovered that most of the headers correspond to section titles of the Wikipedia articles. For most of the Wikpedia articles that we worked with, we found that wiki editors tend to add this information as a "header" as a way to more directly communicate with other editors, in a way akin to what we observed in the case of some GitHub projects. In this case, this behavior was more consistent across datasets. As the "header" will probably be highly correlated to the nature of the change introduced, we think in this case the model is indeed able to leverage on this content to correctly generate the message. However, despite the model capabilities in terms of "header" generation, we also observe cases of hallucination in the parts of the messages that lie outside "headers". This is specially apparent in some of the examples for *Donald Trump* and *United States*, as Table 3 shows.

6 Conclusions and Future work

In this paper we proposed to study the automatic generation of descriptions for editing behavior in online content. Concretely, we introduced models based on the encoder-decoder architecture that are able to generate natural language descriptions for editing activities in Wikipedia and GitHub.

We think our results could represent a concrete contribution in improving our understanding of the evolution knowledge bases, in terms of both software and scientific documentation, from a linguistic perspective. We envision this as a tool that could be useful for supporting documentation and quality-related tasks in collaborative environments, where human supervision is insufficient or not always available.

In terms of future work, one of the main lines we intend to explore is the the design of an ad-hoc metric for automatic evaluation of the generated messages. Alongside that, we also intend to do an indepth human study for a more comprehensive validation and assessing the usefulness of the descriptions we generate. On the other hand, we also intend to improve our models by allowing feature learning from richer inputs, such as abstract syntax trees and also functional such as execution traces in the case of GitHub.

Finally, in this work we have resorted to *diff* files as a primary source of input information, which means our representation contains redundant information and may therefore be inefficient. Although our results showed that this representation works fairly well for the proposed setting, at the same time providing us a model that is language agnostic, we would like to explore other alternatives to model the input. In particular, we are interested in models that directly take a pair of versions of a given document, for example the version before and after a certain introduced change, allowing us to generalize our proposal to different time scales.

Acknowledgments

We are grateful for the support provided by the NVIDIA Corporation, donating two of the GPUs used for this research.

References

[Allamanis et al.2016] Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International Conference on Machine Learning (ICML)*.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations*, San Diego, California.

[Batista et al.2017] Natércia A. Batista, Michele A. Brandão, Gabriela B. Alves, Ana Paula Couto da Silva, and Mirella M. Moro. 2017. Collaboration strength metrics and analyses on github. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 170–178, New York, NY, USA. ACM.

[Bertoldi et al.2009] Nicola Bertoldi, Haddow Barry, and Jean-Baptiste Fouet. 2009. Improved minimum error rate training in moses. *The Prague Bulletin of Mathematical Linguistics*, pages 1–11.

[Brandl2016] Georg Brandl. 2016. Pygments: Python syntax highlighter. `http://pygments.org`.

[Buse and Weimer2010] Raymond P.L. Buse and Westley R. Weimer. 2010. Automatically documenting program changes. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 33–42, New York, NY, USA. ACM.

[Chisholm et al.2017] Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642, Valencia, Spain, April. Association for Computational Linguistics.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

[Cortés-Coy et al.2014] Luis Fernando Cortés-Coy, Mario Linares Vásquez, Jairo Aponte, and Denys Poshyvanyk. 2014. On automatically generating commit messages via summarization of source code changes. In *SCAM*, volume 14, pages 275–284.

[Crowston et al.2007] Kevin Crowston, Qing Li, Kangning Wei, U Yeliz Eseryel, and James Howison. 2007. Self-organization of teams for free/libre open source software development. *Information and software technology*, 49(6):564–575.

[Dabbish et al.2012] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM.

[Gandon et al.2016] F. Gandon, R. Boyer, O. Corby, and A. Monnin. 2016. Wikipedia editing history in dbpedia: Extracting and publishing the encyclopedia editing activity as linked data. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 479–482, Oct.

[Gousios et al.2014] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355. ACM.

[Guzman et al.2014] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in github: An empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 352–355, New York, NY, USA. ACM.

[Heafield et al.2013] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

[Hindle et al.2012] Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. 2012. On the naturalness of software. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 837–847. IEEE.

[Hinton1984] Geoffrey E Hinton. 1984. Distributed representations.

[Iyer et al.2016] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany, August. Association for Computational Linguistics.

[Jiang and McMillan2017] Siyuan Jiang and Collin McMillan. 2017. Towards Automatic Generation of Short Summaries of Commits. pages 320–323. IEEE, May.

[Jiang et al.2017] Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. Automatically generating commit messages from diffs using neural machine translation. pages 135–146. IEEE, October.

[Kiesel et al.2017] Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2017. Spatio-temporal analysis of reverted wikipedia edits.

[Kiss and Strunk2006] Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Comput. Linguist.*, 32(4):485–525, December.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

[Lavie and Agarwal2007] Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Linares-Vásquez et al.2015] Mario Linares-Vásquez, Luis Fernando Cortés-Coy, Jairo Aponte, and Denys Poshyvanyk. 2015. Changescribe: A tool for automatically generating commit messages. In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, pages 709–712. IEEE Press.

[Liu et al.2018] Zhongxin Liu, Xin Xia, Ahmed E. Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: How far are we? In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ASE 2018, pages 373–384, Montpellier, France, September. Association for Computing Machinery.

[Liu et al.2019a] Qin Liu, Zihe Liu, Hongming Zhu, Hongfei Fan, Bowen Du, and Yu Qian. 2019a. Generating Commit Messages from Diffs Using Pointer-generator Network. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, pages 299–309, Piscataway, NJ, USA. IEEE Press.

[Liu et al.2019b] Shangqing Liu, Cuiyun Gao, Sen Chen, Lun Yiu Nie, and Yang Liu. 2019b. ATOM: Commit Message Generation Based on Abstract Syntax Tree and Hybrid Ranking. *arXiv:1912.02972 [cs]*, December.

[Loyola et al.2017] Pablo Loyola, Edison Marrese-Taylor, and Yutaka Matsuo. 2017. A neural architecture for generating natural language descriptions from source code changes. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 287–292, Vancouver, Canada, July. Association for Computational Linguistics.

[Loyola et al.2018] Pablo Loyola, Edison Marrese-Taylor, Jorge Balazs, Yutaka Matsuo, and Fumiko Satoh. 2018. Content Aware Source Code Change Description Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 119–128, Tilburg University, The Netherlands. Association for Computational Linguistics.

[Luong et al.2015] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

[Marcus et al.1993] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

[Marrese-Taylor et al.2019] Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2019. An Edit-centric Approach for Wikipedia Article Quality Assessment. In *Proceedings of the 5th Workshop on Noisy User-Generated Text (W-NUT 2019)*, pages 381–386, Hong Kong, China, November. Association for Computational Linguistics.

[McIntosh and Kamei2017] Shane McIntosh and Yasutaka Kamei. 2017. Are fix-inducing changes a moving target? a longitudinal case study of just-in-time defect prediction. *IEEE Transactions on Software Engineering*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

[Nielek et al.2016] R. Nielek, O. Jarczyk, K. Pawlak, L. Bukowski, R. Bartusiak, and A. Wierzbicki. 2016. Choose a job you love: Predicting choices of github developers. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 200–207, Oct.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

[Proisl and Uhrig2016] Thomas Proisl and Peter Uhrig. 2016. Somajo: State-of-the-art tokenization for german web and social media texts. In *Proceedings of the*

9th Web as Corpus Workshop (WaC-X) and the EmpiriST Shared Task, pages 57–62, Berlin, Germany. Association for Computational Linguistics.

[Sarkar et al.2019] Soumya Sarkar, Bhanu Prakash Reddy, Sandipan Sikdar, and Animesh Mukherjee. 2019. StRE: Self Attentive Edit Quality Prediction in Wikipedia. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3962–3972, Florence, Italy, July. Association for Computational Linguistics.

[Su and Liu2015] Q. Su and P. Liu. 2015. A psycholexical approach to the assessment of information quality on wikipedia. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 184–187, Dec.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Von Krogh et al.2003] Georg Von Krogh, Sebastian Spaeth, and Karim R Lakhani. 2003. Community, joining, and specialization in open source software innovation: a case study. *Research Policy*, 32(7):1217–1241.

[Yang et al.2017] Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2017. Identifying Semantic Edit Intentions from Revisions in Wikipedia. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2000–2010, Copenhagen, Denmark. Association for Computational Linguistics.

[Yin et al.2019] Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. 2019. Learning to Represent Edits. In *Proceedings of the 7th International Conference on Learning Representations*.

[Zhang et al.2017] Amy X. Zhang, Lea Verou, and David Karger. 2017. Wikum: Bridging discussion forums and wikis using recursive summarization. pages 2082–2096.